buildings

MDPI

Article

Use of Real Coded Genetic Algorithm as a Pre-Dimensioning Tool for Prestressed Concrete Beams

Tarniê Vilela Nunes Narques [1,*], Roberto Chust Carvalho [1], André Luis Christoforo [1], Fernando Júnior Resende Mascarenhas [1], Felipe Nascimento Arroyo [1], Florisvaldo Cardozo Bomfim Junior [2] and Herisson Ferreira dos Santos [3]

[1] Department of Civil Engineering, Federal University of São Carlos, São Carlos 13565-905, SP, Brazil
[2] Department of Electrical Engineering, Federal University of Uberlândia, Uberlândia 38400-902, MG, Brazil
[3] Research Department of the Federal Institute of Rondônia, Federal Institute of Education Science and Technology of Rondônia, Vilhena 76980-000, RO, Brazil
* Correspondence: tarnienarques@outlook.com; Tel.: +55-(34)-99677-8591

Abstract: In project practice, the search for optimal solutions is based on the traditional process of trial and error, which consumes much time and does not guarantee that solutions found are the optimal solutions for the problem. Many studies have been developed in recent years with the aim of solving problems in various fields of structural engineering with the aid of intelligent algorithms; however, when it comes to the optimization of structural designs, the approaches considered by the authors involve a large number of variables and constraints, making the implementation of optimization techniques difficult and consuming significant processing time. This research aims to evaluate the efficiency of intelligent algorithms when associated with structural optimization approaches that are simpler to implement. Therefore, a Genetic Algorithm in Real Coding was built to serve as an auxiliary tool for pre-dimensioning prestressed concrete beams. With this, the problem becomes simpler to implement, as it depends on a smaller number of variables, leading to less processing time consumption. Simulations were performed to calibrate the Genetic Algorithm and find the optimal solution later. The solution found by the algorithm was compared with the real solution of a project that had already gone through a traditional optimization process. Even in these circumstances, the proposed Genetic Algorithm was able to find, in 210 s, a more economical solution. Our studies found that even with more straightforward approaches, intelligent algorithms can help in the search for optimal solutions to structural engineering problems; in addition, using real coding in fact proved to be a great strategy due to the nature of the problem, making the implementation of the algorithm simpler and ensuring answers with little processing time.

Keywords: optimization; prestressed concrete; I-girger; pre-dimensioning; Genetic Algorithm; real coded; Simulated Binary Crossover (SBX)

Citation: Narques, T.V.N.; Carvalho, R.C.; Christoforo, A.L.; Mascarenhas, F.J.R.; Arroyo, F.N.; Bomfim Junior, F.C.; Santos, H.F.d. Use of Real Coded Genetic Algorithm as a Pre-Dimensioning Tool for Prestressed Concrete Beams. *Buildings* **2023**, *13*, 819. https://doi.org/10.3390/buildings13030819

Academic Editor: Andreas Lampropoulos

Received: 17 February 2023
Revised: 13 March 2023
Accepted: 16 March 2023
Published: 21 March 2023

1. Introduction

In recent years, there has been significant growth in the number of research works that apply intelligent algorithms to solve problems in the most diverse areas of the field of structural engineering. The authors in [1], for example, combining Support Vector Machine (SVM) and Moth-flame Optimization (MFO), presented an efficient method to solve the problem of identifying damage in bridge structures considering the ambient temperature variation. To validate the research, the authors [1] applied the method to identify damages in a simply supported numerical beam and an experimental example of a bridge. Several other works in the same line of research, i.e., using intelligent algorithms for structural health monitoring, were highlighted in the summary review carried out by [2].

Intelligent algorithms have also been widely used in structural engineering to assist in the search for design solutions at the lowest possible cost [3,4]. Although research

in this area has already advanced a lot, even today, the search for optimal solutions to these problems, in practice, is based on the empirical process of trial and error. This procedure is considered inefficient for this purpose because it consumes much time from the professional and works with a very restricted search universe, not guaranteeing that the adopted solution represents the optimum one for the analyzed problem.

However, this behavior that professionals have is understandable due to the difficulty of equating models to search for optimized solutions for problems in the structural engineering field since many variables and restrictions surround them. The authors of [4–7] pointed out that the ideal would be to seek optimal solutions through metaheuristic techniques in these cases. The authors of [3,4] even demonstrated that the number of studies using these techniques to optimize problems in the civil engineering field has increased in recent decades. However, there is a certain lack of research focused on practical ways to optimize prestressed concrete elements.

Optimizing prestressed concrete elements, mainly precast ones, becomes particularly interesting since they are generally produced in series to meet project and market demands. In addition, they are used to solve problems of highly costly works, such as industrial sheds and bridges, where any cost reduction becomes significant. Furthermore, this type of structural project and construction is surrounded by many variables and restrictions, making the optimization process very difficult, using the traditional trial and error methodology. Hence, these facts were the reasons that guided the choice of the element to be optimized in the research.

Among the works published in the literature on optimizing prestressed elements using metaheuristic techniques, we can highlight [8], which modeled a Genetic Algorithm (GA) to propose a new cross-section of prestressed concrete wind turbine towers. The solution presented in the paper guaranteed savings of 15% compared with the traditionally used cross-sections. In [9], a procedure to optimize prestressed concrete flat slabs was presented, where initially, the authors tested two direct research methods, Nelder–Mead and Sequential Quadratic Programming. However, both methods did not provide satisfactory results due to the nature of the problem, which led the authors to work with a combination of Multi-Objective Evolutionary Algorithms with GA to find the optimal solution. The authors of [10] used a traditional Genetic Algorithm to optimize prestressed unidirectional ribbed slabs.

In [11], the authors modeled a GA to optimize the design of the structural elements of a bridge composed of multiple precast beams of I cross-section, reinforced concrete slabs, and columns with H cross-section. Based on that, 33 constraints were imposed on the problem, and the final result was 12.6% more economical than the actual solution implemented. The authors of [12] presented an automated approach associating the model of a prestressed slab simulated in SAP2000 with two metaheuristic algorithms, Particle Swarm Optimization (PSO) and its improved version PSOHS, the latter being the one that presented the best performance. A new method for optimizing prestressed slabs using the Method of Moving Asymptotes (MMAs) was presented in [13]. This methodology led to up to 50% savings compared with the traditional design methodology; however, the authors pointed out that a lot of processing time was consumed to find the optimal solutions to the problems.

It is evident, analyzing the results of [8–13], that the metaheuristic techniques, particularly the GA, have presented themselves as a good alternative for optimizing prestressed elements, corroborating with the reports of [4–7]. However, as shown in [13], the way the problem is approached can significantly influence the processing time of the optimization algorithm.

In [5], for example, the authors considered 59 discrete variables in the optimization problem. The objective was to find the final design solution for a road bridge composed of a solid concrete slab and U-section beams. Seven geometric variables were considered: beam height, slab thickness, beam soffit width, beam soffit thickness, width and thickness of flanges, and thickness of webs, and two variables to consider different concrete grading

used in beams and slab. The other 50 variables were used to define the pattern and positioning of the transverse and longitudinal reinforcement of the beams and slab. Due to the complexity of the problem, it took 36,233 s for the algorithm to find the most economical solution.

Approaches such as these, which involve a large number of variables and constraints, require a lot of programming effort and consume a lot of processing time, distancing the applicability of metaheuristic techniques from the leading agents they would serve, the professionals in the field of structural engineering [14].

An alternative to these issues would be applying approaches that involve fewer variables and constraints, making the coding of metaheuristics more straightforward to implement and delivering results with less processing time. In line with this proposal, this research aimed to evaluate the GA efficiency when applied as an auxiliary pre-dimensioning tool since, in this context, the problems depend on fewer variables and restrictions.

To validate the research, a GA was elaborated to assist in pre-dimensioning prestressed beams for the roof of sheds. Its optimal solution was compared with the real solution of an already executed project. It is noteworthy that the real solution, according to the engineers' report, was the product of a succession of optimization studies using the traditional trial and error methodology, which aimed to find the lowest possible cost solution. Even so, the GA proposed in the research, composed of only six variables, could find a more economical solution in 210 s, showing that algorithms such as the one proposed in this research, designed for the pre-dimensioning level, can also lead to optimal results.

In order to maintain the proposal for more straightforward coding to be implemented, the entire GA was structured in real coding, eliminating the need for conversion routines between the real and binary systems, thereby significantly reducing the processing time of the algorithm. In addition, real codes guarantee a greater approximation of the optimal solutions to the problems. They are more precise, mathematically speaking, than binary codes, which have their precision directly related to the number of bits adopted in the programming. Another point worth mentioning in this research is the use of the Simulated Binary Crossover (SBX), a Crossover Operator suitable for real coding, which had not yet been considered in the optimization of structural engineering problems.

2. Materials and Methods

This research evaluated the efficiency of applying a GA, designed simpler, to work as an auxiliary tool for pre-dimensioning a prestressed concrete beam. The prestressed element sizing routine and the Genetic Algorithm routine were coded in the MATLAB computational environment [15] on a laptop with an Intel® Core™ i5-3210M CPU @ 2.5 GHz and 4.00 GB of installed RAM. It is noteworthy that, although there is already a toolbox formatted for the Genetic Algorithm for MATLAB (Genetic Algorithm Optimization Toolbox—GAOT), the entire GA coding of this research was implemented by the authors. It was performed to assess its complexity, obtain greater control over the algorithm's behavior, and provide the flexibility to choose the operators that best fit the problem and research proposal. The entire design routine of the prestressed element followed the criteria established by Brazilian codes [16–19].

The optimization problem was detailed in the following sections, and the particularities of the beam design routines and the implemented Genetic Algorithm were later detailed.

2.1. Problem to Be Optimized

The object optimized in this research was a prestressed concrete beam on the roof of an industrial shed built using the precast system, a widespread type of structure in Brazil [20]. To evaluate the efficiency of the GA, its result was compared with the actual project of one of the roof beams of a shed, already executed, of 5280 m^2, whose schematic perspective is presented in Figure 1.

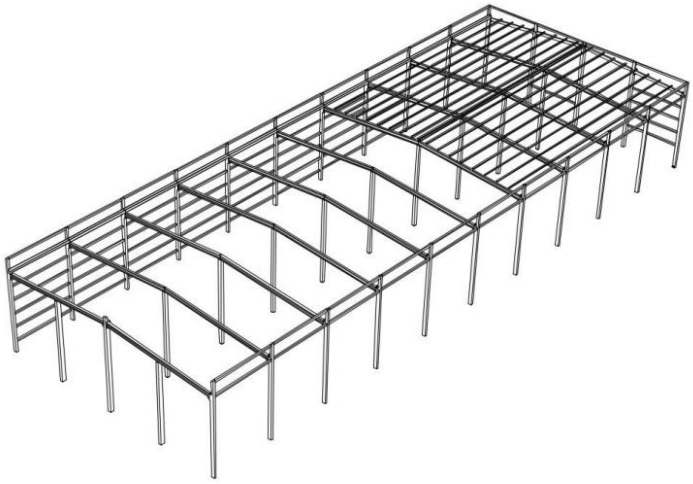

Figure 1. Schematic view of the industrial shed.

The studied industrial shed in question is formed by a set of 11 frames spaced every 12 m, with internal frames composed of three rectangular columns and two beams with an I cross-section spanning 22 m, as shown in Figure 2.

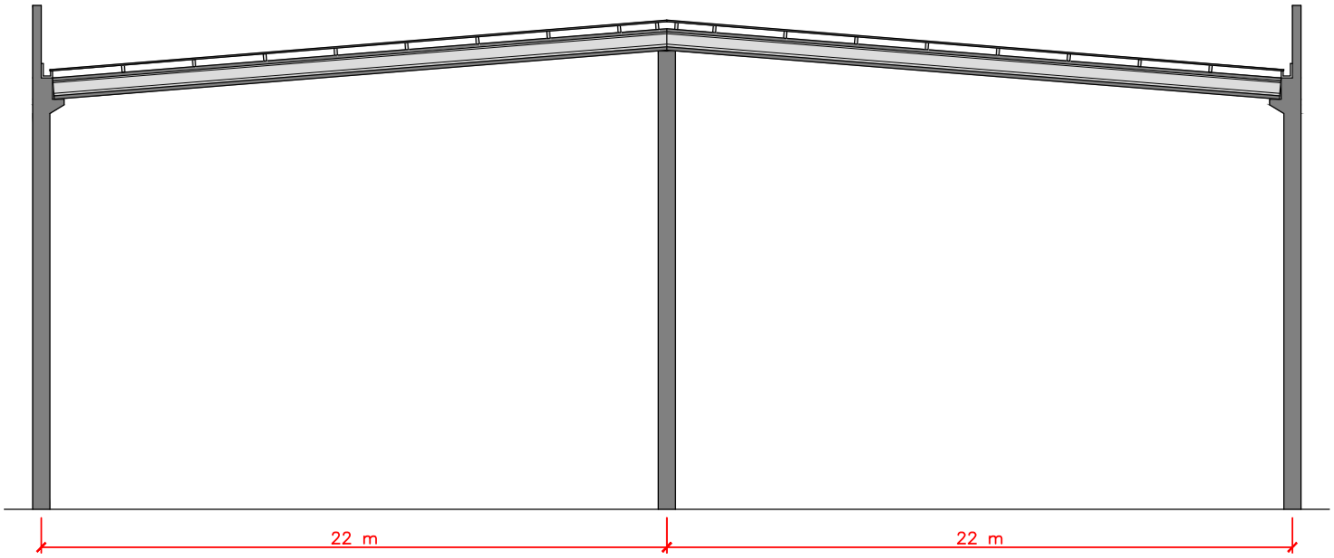

Figure 2. The internal frame of the industrial shed.

The roof of the industrial shed was fabricated of metal tiles, supported by prestressed purlins of T-section, which in turn, were supported by the roof beams. The beams with an I cross-section of the internal frame of the industrial shed were chosen to be optimized in this research. The following parameters were considered fixed:

- The distance between the frames: 12 m, according to the project;
- Beam Length: 21.60 m;
- Beam Concrete Strength: 40 MPa;
- Prestressing reinforcement: CP 190 RB;
- Use of pretensioning;
- Distance from the CG of the active reinforcement to the bottom edge of the beam: 0.10 m;
- Cost of concrete and steel reinforcement.

The costs were raised in Tables of Unified Unit Prices for Bridges (in Portuguese *Tabelas de Preços Unitários Unificados Não Desonerados de Obras de Artes Especias*), developed by the

Department of Logistics and Transport of the State of São Paulo (in Portuguese *Secretaria de Logistica e Transporte do Estado de São Paulo*) at the time of the development of the research.

The same loads used in the real project were considered in calculating the forces acting on the beams. The permanent loads due to the own weight of precast elements (beam and purlins); the permanent load due to the own weight of the metallic tile (0.11 kN/m^2); a permanent overload of 0.05 kN/m^2 representing equipment permanently fixed to the roof (lighting, sprinklers, and others.); and an accidental overload of 0.25 kN/m^2. It is noteworthy that the beam's own weight was calculated within the Genetic Algorithm routine after defining the geometric characteristics of the beams.

To determine the forces due to the wind, the program Visual Ventos Version 2.0.2 was used, which follows the prescriptions of [18] and a basic wind speed of 40 m/s. As in the project, it was considered that the building would have industrial installations with a low occupation factor. It would be executed on flat land, covered by numerous and closely spaced obstacles in an industrial zone. It was also considered that the studied building would be effectively watertight to determine the internal pressure coefficient.

2.2. Genetic Algorithm

According to [21], Genetic Algorithms are metaheuristic optimization techniques that are simple to understand and implement, developed in the 1960s by John Holland and a group of researchers from the University of Michigan. The method used consecrated concepts of genetics and was inspired by Charles Darwin's theory of natural selection of individuals and the survival of those most adapted to the environment [21,22].

Genetic Algorithms have already been widely applied, and their efficiency has been proven through studies in different fields of knowledge. Its adaptability and operating mechanics allow it to quickly escape the local optimum points surrounding nonlinear and non-convex problems, such as those in structural engineering [23]. Figure 3 presents the Flowchart of a Genetic Algorithm. As GA has already been widely publicized in the technical field, we suggest reading [21,22,24,25] for further studies on its structure and the particularities of its operators.

2.3. Genetic Algorithm Implemented in the Research

Enabling future reproductions of the algorithm to anyone interested and demonstrating the simplicity of the coding, a brief presentation of each of the steps of the GA was made. The particularities adopted in the code implemented in the research were also demonstrated.

2.3.1. Step 1—Generate Initial Random Population

In order to design an initial space for searching for solutions, the Genetic Algorithm creates a population with a predetermined number of individuals, each of which is a possible solution to the problem. Each individual carries a set of variables with values randomly assigned at this stage to guarantee that they assume different positions in the search space for the optimal solution to the problem. These are the variables manipulated by GA during the optimization process. After some preliminary tests, it was decided to structure the research's GA with a population of 150 individuals, i.e., 150 possible beams.

Six variables were considered in the optimization of this problem: the width and height of the bottom flange of the beam (b_w e h_5); the width and height of the top flange of the beam (b_f *e* h_1); and the thickness and height of the web (b_a e h_3). All these variables were schematically detailed in Figure 4. Dimensions h_2 and h_4, which complete the section, were determined as a function of angles of 20° and 45°, respectively, respecting the same specifications as the reference project. As mentioned, structuring the GA in real coding eliminated the need to convert all the variables of all individuals to the binary system.

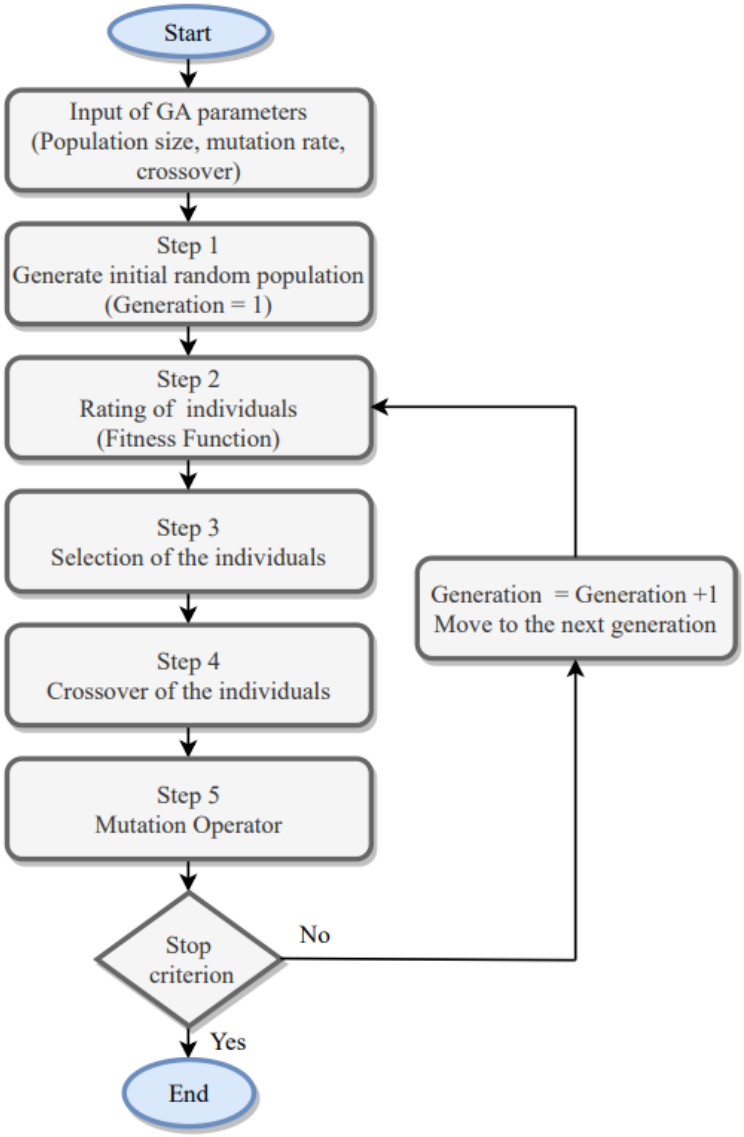

Figure 3. Flowchart of a Genetic Algorithm.

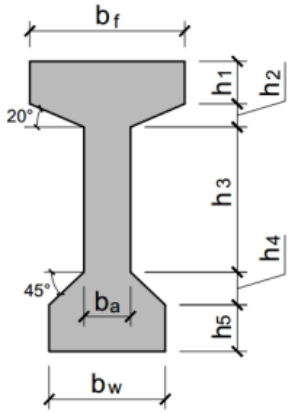

Figure 4. Schematic cross-section of the beam to be optimized.

Aiming to eliminate infeasible solutions and guarantee less processing time until the solution converges, the values attributed to the variables of the individuals of the initial population followed the rules demonstrated by Equations (1)–(6), which were created

based on border technical criteria. In these equations, *rand*(0, X) é was a random value determined by MATLAB between 0 and X.

$$b_{w,i} = 0.30 + rand(0, 0.20), \quad (1)$$

$$b_{a,i} = 0.12 + rand(0, 0.08), \quad (2)$$

$$b_{f,i} = 0.40 + rand(0, 0.20), \quad (3)$$

$$h_{1,i} = 0.12 + rand(0, 0.10), \quad (4)$$

$$h_{3,i} = 0.25 + rand(0, 0.40), \quad (5)$$

$$b_{5,i} = 0.12 + rand(0, 0.10), \quad (6)$$

2.3.2. Step 2—Rating of Individuals

In Step 2, individuals were assessed using the Fitness Function (Equation (7)). For this to be possible, all individuals, prior to this step, went through the sizing routine described in Section 2.6.

$$F_{(x)} = f_{(x)} + pen_{(x),} \quad (7)$$

where $F_{(x)}$ is the Fitness Function; $f_{(x)}$ is the Objective Function of the optimization problem; and $pen_{(x)}$ is the Penalty Function.

There are different types of Fitness Functions found in the related literature, and one of them was used in developing this work research. This variety of functions is because Genetic Algorithms are adaptable to different optimization problems. Despite the different formulations in the literature, their purposes always remain to determine how well each individual solves the problem.

The Objective Function ($f_{(x)}$) quantifies the fitness of individuals according to some measurable parameter, such as the cost of its production, for example. To make this possible, a cost calculation routine for concrete (C_c) and steel reinforcement (C_{ap}) was implemented in the research GA. These values were later added to the Objective Function of the problem in question (Equation (8)).

$$f_{(x)} = C_c + C_{ap}, \quad (8)$$

The Penalty Function ($pen_{(x)}$) is linked to the restrictions of the problem and aims to penalize those individuals who, by chance, did not meet some of these restrictions, purposefully increasing, in the case of this research, their aptitude (Cost), making them less eligible for the next steps. The Penalty Function used in the algorithm was detailed in Equation (9).

$$pen_{(x)} = K\,C, \quad (9)$$

where "*C*" is the number of unfulfilled constraints, and "*K*" is a constant with a value proportional to the Objective Function of the problem; in this case, USD 100.00 was adopted, a value equivalent to approximately 20% of the average cost of beams of this size.

To meet the requirements of [16], 10 restrictions were imposed on this research problem, namely:

1. Stress limit in concrete to meet the decompression limit state (evaluation of the upper edge of the cross-section);
2. Stress limit in concrete to meet the decompression limit state (assessment of the lower edge of the cross-section);
3. Stress limit in concrete to meet the limit state of crack formation (evaluation of the upper edge of the cross-section);
4. Stress limit in concrete to meet the limit state of crack formation (evaluation of the lower edge of the cross-section);
5. Stress limit in concrete to meet the limit state of excessive compression (evaluation of the upper edge of the cross-section);

6. Stress limit in concrete to meet the limit state of excessive compression (evaluation of the lower edge of the cross-section);
7. Stress limit in concrete to meet the decompression limit state (evaluation of the upper edge of the cross-section);
8. Stress limit in concrete to meet the ultimate limit state in the moment of prestressing (assessment of the lower edge of the cross-section);
9. Excessive deformation limit state verification;
10. Verification regarding the arrangement of reinforcements in the cross-section.

2.3.3. Step 3—Selection of Individuals

At this time, individuals with the best aptitudes were selected as progenitors of the following population employing a Selection Operator. It was decided to implement the traditional operator "Selection by Tournament" in the research algorithm, which promotes competition between randomly selected individuals in the population and assigns a position in the ranking (parent population) to the champion. In this research, those with the lowest cost were the champions.

2.3.4. Step 4—Crossover of the Individuals

In Step 4, the construction of the new population of possible solutions to the problem began. With the Crossover Operator, new individuals were created by merging information from pairs of individuals from the parent population to bring them closer, in each generation, to the optimal solution of the problem. The Crossover Operator used in the research algorithm was the Simulated Binary Crossover (SBX), a real coding operator that simulates the operation of the traditional Single Point Binary Crossover Operators [26,27]. It is noteworthy that, although the efficiency of the SBX has already been proven concerning the Binary Crossing Operators [28–30], no structural optimization works that considered its use was found in our research. More detailed information about this operator is presented in Section 2.4.

2.3.5. Step 5—Mutation

Finally, the last step of the Genetic Algorithm is the application of the Mutation Operator. This has the purpose of changing the characteristics of a small portion of individuals of the new population (generated in Step 4), forcing it to remain diversified and providing the algorithm with escape routes from local optimal points at each iteration. The mutation rate used by the operator must be previously defined in the coding.

After completing this entire march, the algorithm counts an iteration and starts its routine again; however, now, adopting as initial population, that of new solutions generated in Step 4 and finalized in Step 5. This routine is repeated countless times until a predefined stopping point is reached or until the predetermined cycle of iterations is completed.

In the GA of this research, the 6 variables of each individual were tested at this stage according to the mutation rate. If any of these were chosen for mutation, a new value, controlled by Equations (1)–(6), was assigned to it. For the first simulations of the research, a mutation rate of 1% was fixed, and a total of 60 iterations was adopted as a stopping criterion.

2.4. Simulated Binary Crossover (SBX)

The SBX is a real variable Crossover Operator designed to simulate the operation of traditional Single Point Binary Crossover Operators. According to [26], the motivation for creating the SBX came from the success of Binary Crossing Operators when applied to optimization problems with discrete search space and their difficulties in operating in a continuous search space, as in the case of this research. The SBX uses a non-uniform bimodal probability distribution, and as it automatically adapts during the GA execution, it is considered a self-adaptive operator [31].

According to [26], several simulations were performed to compare the efficiency of the SBX with the Single Point Binary Operator. According to the authors [26], when the SBX was not shown to be more efficient than the binary operator, it was at least equal to its efficiency.

Similar to most Crossover Operators, SBX generated two new individuals (F_1 e F_2), conventionally treated as children, from two individuals of the parent population (P_1 e P_2), usually treated as parents. Equations (10)–(14) were used to structure the construction of the operator [27].

$$F_1 = 0.5[(1-\beta)P_1 + (1+\beta)P_2] \tag{10}$$

$$F_2 = 0.5[(1-\beta)P_2 + (1+\beta)P_1] \tag{11}$$

$$u = rand(0,1) \tag{12}$$

$$u \leq 0.5 \rightarrow \beta = (2u)^{\frac{1}{\eta+1}} \tag{13}$$

$$u > 0.5 \rightarrow \beta = \left(\frac{1}{2(1-u)}\right)^{\frac{1}{\eta+1}} \tag{14}$$

where u is a random variable with a uniform distribution between 0 and 1; and η any non-negative real number. The authors of [27] pointed out that the greater the value of η, the greater the probability of creating children with characteristics close to the parents.

2.5. GA Solution Validation Criteria

In several stages of the GA, randomization processes are commonly used. For this reason, there is the possibility of not finding the optimal global solution to the problems with only one simulation. Therefore, it is recommended that a series of simulations be carried out to validate the solution. It was established in this research that 10 simulations would be performed to investigate the optimal solution to the problem.

2.6. Beam Design Criteria

According to the Brazilian normative code, the design of prestressed elements must ensure compliance with the ultimate and serviceability limit states and present geometric conditions that favor good conditions for assembling the reinforcement and concreting of the element. The necessary normative specificities regarding each item implemented in the algorithm's sizing routine are presented below.

2.6.1. Ultimate Limit State (ULS)

To ensure compliance with the ULS, the demands imposed on the structural elements must be less than their resistant capacity, as shown by Equation (15):

$$S_d \leq R_d, \tag{15}$$

where S_d represents the acting loads according to the typical load combinations, and R_d is the resistance (load-carrying capacity) of the element.

Following this condition, an active longitudinal reinforcement dimensioning routine was implemented to determine the minimum amount necessary to meet the bending moment acting in the most critical section of the beam (mid-span). Routines for calculating passive transverse and longitudinal reinforcement were not implemented, considering that the final amount of these in the project is usually defined according to the designer's particular detailing criteria, surpassing the designed reinforcement value.

In addition, a routine was implemented to verify compliance with the ULS at the time of prestressing, taking into account the normal stresses caused by the loads acting at the time of prestressing and the compressive strength of the concrete on the date of prestressing, limited to 70% of its value, as recommended in item 17.2.4.3 of [16].

2.6.2. Serviceability Limit State (SLS)

In order to guarantee ideal conditions of use and durability of the structural element, routines were implemented to verify the limit state of excessive deformation and the other verifications associated with the stress limits established for the concrete, namely, verification of crack formation, decompression, and excessive compression. The limit stresses for each case are presented in Equations (16)–(18), respectively:

$$\sigma \geq -0.7 \cdot \alpha \cdot f_{ct,m} \tag{16}$$

$$\sigma \geq 0\,\mathrm{MP}_a \tag{17}$$

$$\sigma \leq 0.7 \cdot f_{ck} \tag{18}$$

where σ represents the normative maximum stresses for each one of the respective cases, $f_{ct,m}$ is the average tensile strength of concrete, and f_{ck} is the characteristic compressive strength of concrete.

To validate the above conditions, the normal stresses acting on the top and bottom edges of the cross-sections of the beam were calculated, considering the load combinations specified by the Brazilian code for each case. In addition, the loading conditions operating in the assembly phase were also considered.

Notably, the due prestressing losses and the maximum value allowed by the Brazilian code for the initial prestressing stress were considered in the calculations, both in the ULS and SLS studies.

2.6.3. Reinforcement Detailing Criteria

In order to guarantee feasible solutions from an executive point of view that did not generate problems during concreting, a routine was implemented, following the recommendations of item 18.6.2.3 of [16] to assess whether the dimensions of the beam were sufficient to accommodate the set of active armor assigned to it.

3. Results and Discussion

The studies developed to optimize the beam proposed in this research were divided into three stages. Initially, simulations were carried out to evaluate the behavior of the GA against the parameter settings assumed in the construction of the algorithm. Subsequently, new simulations were performed to calibrate the GA parameters and improve their performance. Finally, the final simulations were carried out in search of the optimal solution to the problem. Each of these stages of the research is presented in more detail in the following text.

3.1. Initial Simulations to Evaluate GA Behavior

With the algorithm built as described in Section 2, 10 simulations were performed, and the solutions found in each of them are listed in Table 1.

All the solutions found by GA met the 10 restrictions imposed on the problem, showing that the Penalty Function, calibrated with a constant "K" equal to USD 100.00, helped to exclude solutions that did not meet some criteria as established by [16]. Analyzing the results, it was possible to notice that the GA was not able to converge to a single solution and that the price difference between the two extremes was significant, USD 83.98. This possibly occurred due to the strong influence of the variables on the final result; and due to the assumed GA parameter configuration. The graph representing the average value of the Fitness Function of the individuals at each iteration of Simulations 2-1, 4-1, and 6-1 was plotted in Figure 5 to illustrate the discrepancy in the behavior of the GA. Thus, these results highlight the need for a study to calibrate its parameters.

Table 1. Summary of solutions (150 individuals, 60 iterations, 1% mutation rate $\eta = 4$).

Simulation	b_w (m)	b_a (m)	b_f (m)	h_1 (m)	h_2 (m)	h_3 (m)	h_4 (m)	h_5 (m)	h_{total} (m)	Cost (USD)
1-1	0.30	0.12	0.45	0.11	0.06	0.31	0.09	0.15	0.72	479.22
2-1	0.30	0.16	0.41	0.10	0.05	0.40	0.07	0.12	0.74	482.72
3-1	0.30	0.12	0.46	0.13	0.06	0.26	0.09	0.12	0.66	502.70
4-1	0.31	0.17	0.42	0.12	0.05	0.30	0.07	0.15	0.69	531.46
5-1	0.30	0.17	0.45	0.10	0.05	0.35	0.06	0.13	0.69	509.08
6-1	0.30	0.12	0.41	0.16	0.05	0.27	0.09	0.13	0.70	496.98
7-1	0.32	0.15	0.43	0.13	0.05	0.29	0.09	0.15	0.71	523.86
8-1	0.31	0.12	0.41	0.10	0.05	0.38	0.09	0.10	0.72	448.48
9-1	0.30	0.13	0.43	0.13	0.05	0.30	0.07	0.16	0.71	490.76
10-1	0.30	0.13	0.40	0.13	0.05	0.40	0.07	0.10	0.75	454.26

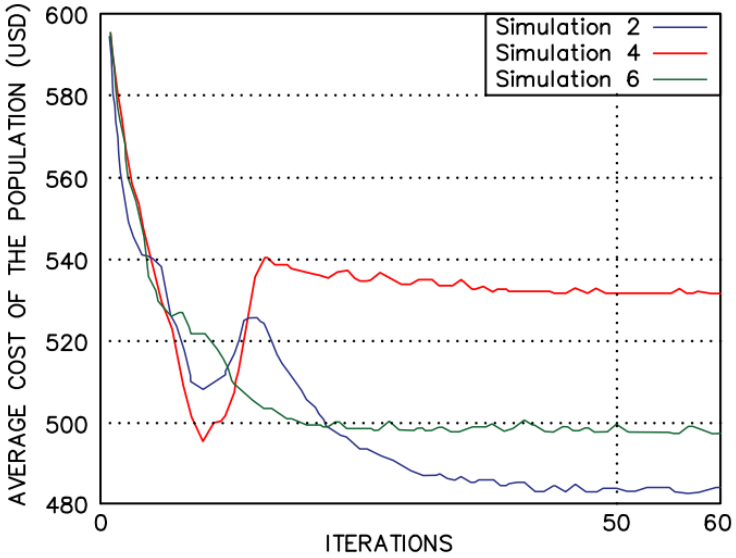

Figure 5. Variation in the average value of the Fitness Function at each iteration of Simulations 2-1, 4-1, and 6-1.

3.2. Calibration of GA Parameters

To improve the performance of the GA, a series of simulations were performed, varying the number of iterations, the SBX operator η coefficient value, and the Mutation Rate value. Altogether, 48 combinations were generated, each tested 10 times.

The best performance was found with the combination considering 500 iterations as a stopping criterion, Mutation Rate of 5%, and the adoption of the coefficient η equal to 3. It was noted in this study that the increase in the number of iterations was crucial to improving the performance of the GA, as it provided more time to escape the local optimum points.

3.3. Final Simulations for the Optimization

After GA calibration, 10 new simulations were performed, and their solutions are listed along with the final beam solution of the reference design in Table 2.

Table 2. Summary of solutions (150 individuals, 500 iterations, 5% mutation rate, and $\eta = 3$).

Simulation	b_w (m)	b_a (m)	b_f (m)	h_1 (m)	h_2 (m)	h_3 (m)	h_4 (m)	h_5 (m)	h_{total} (m)	Cost (USD)
1-2	0.30	0.12	0.41	0.10	0.09	0.45	0.05	0.10	0.79	427.68
2-2	0.30	0.12	0.40	0.10	0.09	0.45	0.05	0.11	0.80	427.78
3-2	0.30	0.12	0.42	0.10	0.09	0.34	0.05	0.12	0.70	460.62
4-2	0.30	0.12	0.42	0.11	0.09	0.42	0.05	0.10	0.77	436.64
5-2	0.30	0.12	0.42	0.10	0.09	0.42	0.05	0.10	0.76	437.32
6-2	0.30	0.12	0.40	0.15	0.09	0.45	0.05	0.10	0.84	444.10
7-2	0.30	0.13	0.43	0.10	0.09	0.36	0.05	0.12	0.72	459.08
8-2	0.30	0.12	0.41	0.10	0.09	0.43	0.05	0.10	0.77	431.28
9-2	0.30	0.13	0.42	0.10	0.09	0.42	0.05	0.10	0.76	442.00
10-2	0.30	0.12	0.40	0.11	0.09	0.41	0.05	0.10	0.76	436.00
Real Project	0.30	0.12	0.40	0.11	0.06	0.37	0.09	0.12	0.75	447.00

Note that even after the calibration, the GA could not find a single solution to the research problem; however, with the new combination of parameters, the difference in cost between the extreme solutions became USD 32.94 (a reduction of 54%). Furthermore, the geometric characteristics of the beams found are more homogeneous than those found in the first set of simulations (Table 1). Thus, it is possible to assume that this new combination of parameters led to acceptable beam solutions for the pre-design level.

The best solution found in the second set of simulations, solution 1-2 with a total cost of USD 427.68, surpassed the results found in the first 10 simulations and proved to be 4.3% more economical than the real design solution. It is noteworthy that the comparison was carried out with a beam that had already gone through a lengthy traditional optimization process, as previously mentioned. Even in these circumstances, the GA proposed in the research found a more economical solution in an average processing time of 210 s.

Figure 6 presents the graph representing the mean value of the Fitness Function of individuals in simulations 1-2 in each interaction. Note that results close to the final solution were found in the optimization after the four hundredth iteration, previously crossing two zones of local optimum points.

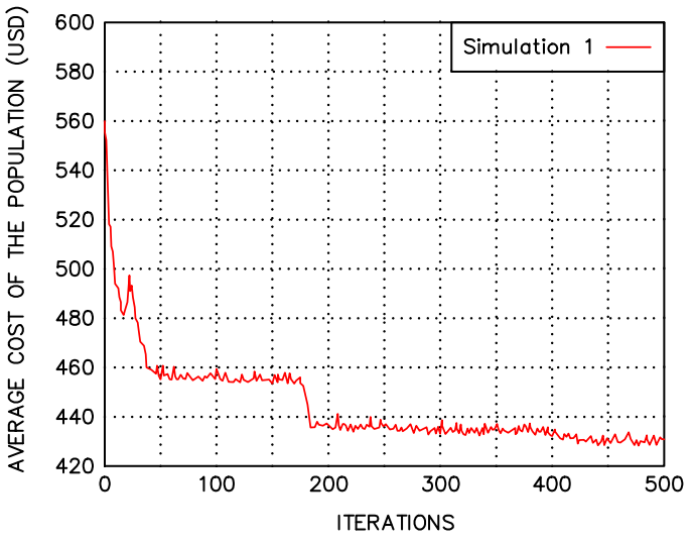

Figure 6. Variation in the mean value of the Fitness Function at each iteration of Simulation 1-2.

To finalize and confirm whether it would be possible to find more economical solutions than the one found in simulations 1-2, new simulations were performed with the same parameter settings; however, setting 600 and 700 iterations as a stopping criterion. Even providing the GA with more time to escape local optimal points, the results were similar to those of the simulations with 500 iterations, as seen in Figure 7.

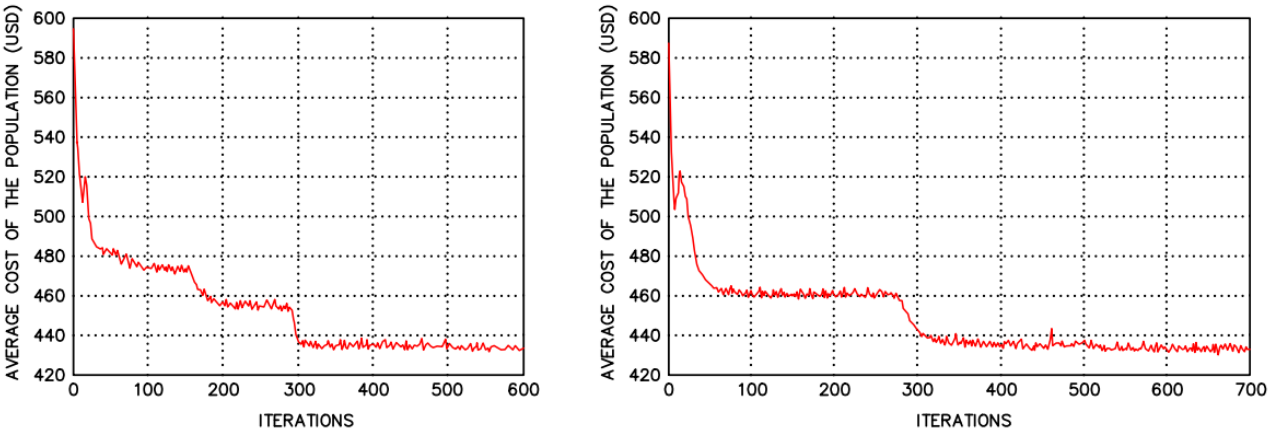

Figure 7. Variation in the mean value of the Fitness Function for simulations with 600 and 700 iterations.

Due to the complexity of the problem, we cannot say that the GA found the optimal global solution to the problem; however, the redundancy of the results found shows that the solution found is adequate for the research problem. Furthermore, this solution would hardly be found as quickly and easily using traditional design routines.

4. Conclusions

In this research, the efficiency of the Genetic Algorithms was evaluated when applied as an auxiliary tool for the pre-dimensioning of prestressed concrete elements. In this context, its coding becomes simpler to implement because it depends on fewer variables and restrictions, saving programming effort and algorithm processing time.

The element chosen to be optimized in the research was a prestressed concrete beam from the roof of a precast shed. The GA efficiency was evaluated by comparing its solution with the real solution of a beam in the roof of an already executed shed. To escape the possibilities of convergence of the solution for points of local optimum, it was established in this research that a series of 10 simulations would be carried out to investigate the optimal solution of the problem.

The results found in the first 10 simulations, considering a population of 150 individuals (150 possible beams cross-sections), 60 iterations, $\eta = 3$, and a mutation rate of 1%, were unsatisfactory, as they delivered very different solutions. Subsequently, a study was conducted, varying the abovementioned parameters to make GA more efficient. The best response was achieved with the following parameters; the population of 150 individuals, 500 iterations, $\eta = 5$, and a mutation rate of 5%.

A second series of 10 simulations were performed, and a solution was found to be 4.3% more economical than the real solution implemented in the project. Despite being a relatively small cost reduction, it is noteworthy that this beam was repeated several times in the shed structure. In addition, the real solution, according to the designers, was the product of a succession of optimization studies using the traditional trial and error methodology, which aimed to find the lowest possible cost solution. Even so, the GA proposed in the research, composed of only six variables, could find a more economical solution in 210 s, showing that algorithms such as the one proposed in the research, designed for the level of pre-dimensioning, can also lead to optimal results.

The result validated the proposed application of GA as an auxiliary pre-dimensioning tool, as it guarantees the designer a good starting point reference for elaborating their

executive projects. It is important to remember that the Genetic Algorithm does not guarantee the finding of the optimal global solution to the problem, but rather better solutions than traditional design routines in a much shorter time, by simulating, in a targeted way, a large number of possibilities.

The proposal to program the entire GA in real coding proved to be simpler and delivered results with little processing time. In addition, it was possible to see that the SBX operator adapted very well to the search optimization problem. Therefore, opportunities for new studies in the line of structural optimization using GAs with similar real coding operators are opened, because there is a lack of studies exploring the implementation of these in the field of civil engineering.

Author Contributions: Conceptualization, T.V.N.N., R.C.C. and A.L.C.; methodology, T.V.N.N., A.L.C., F.C.B.J. and H.F.d.S.; software, T.V.N.N. and F.C.B.J.; validation, F.J.R.M., F.N.A. and H.F.d.S.; formal analysis, T.V.N.N., F.C.B.J. and F.J.R.M.; investigation, F.C.B.J., F.N.A. and A.L.C.; resources, T.V.N.N. and F.N.A.; data curation, R.C.C. and F.J.R.M.; writing—original draft preparation, T.V.N.N., A.L.C. and R.C.C.; writing—review and editing, F.J.R.M., F.N.A. and H.F.d.S.; visualization, F.J.R.M. and F.N.A.; supervision, A.L.C., R.C.C. and H.F.d.S.; project administration, A.L.C., H.F.d.S. and R.C.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Pró-Reitoria de Pesquisa, Inovação e Pós-Graduação of Instituto Federal de Rondônia (PROPESP/IFRO).

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Huang, M.; Lei, Y.; Li, X.; Gu, J. Damage identification of bridge structures considering temperature variations-based SVM and MFO. *J. Aerosp. Eng.* **2021**, *34*, 0402113. [CrossRef]
2. Luo, J.; Huang, M.; Lei, Y. Temperature effect on vibration properties and vibration-based damage identification of bridge structures: A literature review. *Buildings* **2022**, *12*, 1209. [CrossRef]
3. Mei, L.; Wang, Q. Structural Optimization in Civil Engineering: A Literature Review. *Buildings* **2021**, *11*, 66. [CrossRef]
4. Afzal, M.; Liu, Y.; Cheng, J.C.P.; Gan, V.J.L. Reinforced concrete structural design optimization: A critical review. *J. Cleaner Product.* **2020**, *260*, 120623. [CrossRef]
5. Yepes, V.; Martí, J.V.; García-Segura, T.; González-Vidosa, F. Heurístics in optimal detailed design of precast road bridges. *Arch. Cívil Mech. Eng.* **2017**, *17*, 738–749. [CrossRef]
6. Kaveh, A.; Eslamlou, A.D. *Metaheuristic Optimization Algorithms in Civil Engineering: New Applications*, 1st ed.; Springer International Publishing: Cham, Switzerland, 2020; ISBN 978-3-030-45473-9.
7. Saka, M.P.; Hasançebi, O.; Geem, Z.W. Metaheuristics in structural optimization and discussions on harmony search algorithm. *Swarm Evol. Comput.* **2016**, *28*, 88. [CrossRef]
8. Ma, H.W.; Meng, R. Optimization design of prestressed concrete wind-turbine tower. *Sci. China Techol. Sci.* **2014**, *57*, 414–422. [CrossRef]
9. EI Semelawy, M.; Nassef, A.O.; El Damatty, A.A. Design of prestressed concrete flat slab using modern heuristic optimization techniques. *Exp. Sys. Appl.* **2012**, *39*, 5757–5766. [CrossRef]
10. Silva, M.R.L.; Francisco, R.S.; Melo, A.M.; Junior, E.P.; Mota, J.P. Otimização de pavimentos de concreto protendido via Algoritmos Genéticos. In Proceedings of the XXXVIII Iberian Latin American Congress on Computational Methods in Engineering, Florianópolis, SC, Brazil, 4–8 November 2017; ABMEC Brazilian Association of Computational Methods in Engineering: Brasılia, Brazil, 2017.
11. Aydin, Z.; Ayvaz, Y. Overall Cost Optimization of Prestressed Concrete Bridge using Genetic Algorithm. *KSCE J. Civil Eng.* **2013**, *17*, 769–775. [CrossRef]
12. Talaei, A.S.; Nasrollahi, A.; Ghayekhloo, M. An Automated Approach for Optimal Design of Prestressed Concrete Slabs using PSOHS. *KSCE J. Civil Eng.* **2017**, *21*, 782–791. [CrossRef]
13. Zelickman, Y.; Amir, O. Layout optimization of post-tensioned cables in concrete slabs. *Struct. Multidisc. Opt.* **2021**, *63*, 1951–1974. [CrossRef]
14. Waheed, J.; Azam, R.; Riaz, M.R.; Shakeel, M.; Mohamed, A.; Ali, E. Metaheuristic-Based Practical Tool for Optimal Design of Reinforced Concrete Isolated Footings: Development and Application for Parametric Investigation. *Buildings* **2022**, *12*, 471. [CrossRef]

15. Mathworks. MATLAB ®Primer; The MathWorks. Available online: https://www.mathworks.com/help/pdf_doc/matlab/learn_matlab.pdf (accessed on 5 November 2022).
16. Associação Brasileira de Normas Técnicas. *NBR 6118: Projeto de Estruturas de Concreto—Procedimento*; Associação Brasileira de Normas Técnicas: Rio de Janeiro, Brazil, 2014.
17. Associação Brasileira de Normas Técnicas. *NBR 6120: Cargas para o Cálculo de Estruturas de Edifícios*; Associação Brasileira de Normas Técnicas: Rio de Janeiro, Brazil, 2019.
18. Associação Brasileira de Normas Técnicas. *NBR 6123: Forças Devido ao vento em Edificações*; Associação Brasileira de Normas Técnicas: Rio de Janeiro, Brazil, 1988.
19. Associação Brasileira de Normas Técnicas. *NBR 9062: Projeto e Execução de Estruturas de Concreto Pré-Moldado*; Associação Brasileira de Normas Técnicas: Rio de Janeiro, Brazil, 2017.
20. El Debs, M.K. *Concreto Pré-Moldado: Fundamentos e Aplicações*, 2nd ed.; Ofinica de Textos: São Paulo, Brazil, 2017; ISBN 978-85-7975-279-7.
21. Coley, D.A. *An Introduction to Genetic Algorithms for Scientists and Engineers*; World Scientific Publishing Co., Pte. Ltd.: Singapore, 1999; ISBN 978-981-02-3602-1.
22. Goldberg, D.E. *Genetic Algorithms in Search, Optimization, and Machine Learning*; Addison-Wesley Publishing Co., Inc.: Boston, MA, USA, 1989; ISBN 978-02-0115-767-3.
23. Shabbir, F.; Omenzetter, P. Model updating using genetic algorithms with sequential niche technique. *Eng. Struct.* **2016**, *120*, 166–182. [CrossRef]
24. Herrera, F.; Lozano, M.; Verdegay, J.L. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artif. Int. Rev.* **1998**, *12*, 265–319. [CrossRef]
25. Castilho, V.C.; Nicoletti, M.C.; El Debs, M.K. An investigation of the use of three selection-based genetic algorithm families when minimizing the production cost of hollow core slabs. *Comput. Methods Appl. Mech. Eng.* **2005**, *194*, 4651–4667. [CrossRef]
26. Deb, K.; Agrawal, R.B. Simulated Binary Crossover for continuous search space. *Complex Syst.* **1995**, *9*, 115–148.
27. Deb, K.; Kumar, A. Real-coded Genetic Algorithms with Simulated Binary Crossover: Studies on Multimodal and Multiobjective Problems. *Complex Syst.* **1995**, *9*, 431–454.
28. Kumar, S.; Naresh, R. Efficient real coded Genetic Algorithm to solve the non-convex hydrothermal scheduling problem. *Int. J. Electr. Power Energy Syst.* **2007**, *29*, 738–747. [CrossRef]
29. Kumar, S.; Naresh, R. Nonconvex economic load dispatch using an efficient Real-Coded Genetic Algorithm. *Appl. Soft Comput.* **2009**, *9*, 321–329. [CrossRef]
30. Subbaraj, P.; Rajnarayanan, P.N. Optimal reactive power dispatch using self-adaptive Real Coded Genetic Algorithm. *Electr. Power Syst. Res.* **2009**, *79*, 374–381. [CrossRef]
31. Deb, K.; Beyer, H.G. Self-Adaptive Genetic Algorithms with Simulated Binary Crossover. *Evol. Comput.* **2001**, *9*, 197–221. [CrossRef] [PubMed]