applied sciences

MDPI

Article

An Efficient and Fast Hybrid GWO-JAYA Algorithm for Design Optimization

Chiara Furio [1,*], Luciano Lamberti [1] and Catalin I. Pruncu [2]

[1] Department of Mechanics, Mathematics and Management, Polytechnic University of Bari, Via Edoardo Orabona, 4, 70125 Bari, Italy; luciano.lamberti@poliba.it
[2] School of Engineering and the Built Environment, Buckinghamshire New University, 59 Walton Street, Aylesbury HP21 7OG, UK; catalin.pruncu@gmail.com
* Correspondence: c.furio1@phd.poliba.it

Abstract: Metaheuristic algorithms (MHAs) are widely used in engineering applications in view of their global optimization capability. Researchers continuously develop new MHAs trying to improve the computational efficiency of optimization search. However, most of the newly proposed algorithms rapidly lost their attractiveness right after their release. In the present study, two classical and powerful MHAs, namely the grey wolf optimizer (GWO) and the JAYA algorithm, which still attract the attention of optimization experts, were combined into a new hybrid algorithm called FHGWJA (Fast Hybrid Grey Wolf JAYA). FHGWJA utilized elitist strategies and repairing schemes to generate high-quality new trial solutions that may always improve the current best record or at least the old population. The proposed FHGWJA algorithm was successfully tested in seven engineering optimization problems formulated in the fields of robotics, hydraulics, and mechanical and civil engineering. Design examples included up to 29 optimization variables and 1200 nonlinear constraints. The optimization results proved that FHGWJA always was superior or very competitive with the other state-of-the-art MHAs including other GWO and JAYA variants. In fact, FHGWJA always converged to the global optimum and very often achieved 0 or nearly 0 standard deviation, with all optimization runs practically converging to the target design. Furthermore, FHGWJA always ranked 1st or 2nd in terms of average computational speed, and its fastest optimization runs were better or highly competitive with those of the best MHA taken for comparison.

Keywords: metaheuristic optimization algorithms; fast hybrid optimization algorithms; GWO; JAYA; elitist strategies; engineering problems

Citation: Furio, C.; Lamberti, L.; Pruncu, C.I. An Efficient and Fast Hybrid GWO-JAYA Algorithm for Design Optimization. *Appl. Sci.* **2024**, *14*, 9610. https://doi.org/10.3390/app14209610

Academic Editor: Grigorios Beligiannis

Received: 5 September 2024
Revised: 14 October 2024
Accepted: 16 October 2024
Published: 21 October 2024

1. Introduction

Optimization searches for the minimum or the maximum of an *NDV*–variables function $W(\vec{X})$ subject to a set of N_{CON} inequality/equality constraint functions $G(\vec{X})$ or $H(\vec{X})$. Optimization problems may be iteratively solved with gradient based algorithms (GBAs) or metaheuristic algorithms (MHAs). GBAs formulate and solve a series of approximate sub-problems until the search process converges to the optimal solution; sub-problems are built by evaluating first-order and second-order derivatives of cost functions and constraints at the solution point found in the previous iteration. MHAs do not require gradients: local information provided by gradients in GBAs are replaced in MHAs by global information gathered from a population of candidate designs. Trial solutions of MHAs are randomly generated using a mathematical model inspired by evolutionary theory/processes, biology, physics, chemistry, mathematics, astronomy, astrophysics, herd behavior of animals, human behavior and activities, social sciences, etc. MHAs evaluate the new trial solutions randomly generated and then attempt to improve the design with respect to the previous iterations.

Metaheuristic algorithms can inherently deal with highly nonlinear and non-convex optimization problems. The random search allows MHAs to explore larger portions of search space than GBAs without being trapped in local optima. This ability together with their straightforward software implementation made MHAs become the standard approach to the solution of complex engineering optimization problems. Exploration and exploitation are the two typical phases carried out in metaheuristic optimization. Exploration is the leading search mechanism in the early optimization stages where large perturbations are given to optimization variables to quickly find the best regions of the search space. Exploitation governs the search process as the optimizer converges toward the global optimum: local search is carried out in properly selected neighborhoods of the most promising solutions. Exploration and exploitation must be appropriately combined in metaheuristic optimization to fully search the design space and find the globally optimal solution with low computational effort.

MHAs can be categorized in four groups: (i) evolutionary algorithms; (ii) science-based algorithms; (iii) human-based algorithms; (iv) swarm intelligence-based algorithms. Evolutionary algorithms such as genetic algorithms (GAs) [1,2], differential evolution (DE) [3,4], evolutionary programming (EP) [5], evolution strategies (ESs) [6], and biogeography-based optimization (BBO) [7] reproduce evolution theory and evolutionary processes.

Science-based MHAs mimic laws of physics, chemistry, astronomy and astrophysics, and mathematics. Simulated annealing (SA) [8,9], charged system search (CSS) [10], magnetic charged system search (MCSS) [11], colliding bodies optimization (CBO) [12], water evaporation optimization (WAO) [13], thermal exchange optimization (TEO) [14], equilibrium optimizer (EO) [15], gases Brownian motion optimization (GBMO) [16], and Henry gas solubility optimization (HGSO) [17] are physics/chemistry-based MHAs that reproduce equilibrium conditions of mechanical, electro-magnetic, physical/chemical, or thermal systems subject to external perturbations. Ray optimization (RO) [18] and light spectrum optimizer (LSO) [19] reproduce optics laws to define search directions containing high-quality trial solutions. Big bang–big crunch optimization (BB-BC) [20] and the gravitational search algorithm (GSA) [21] are inspired by astronomy/astrophysics phenomena such as the expansion (big bang)—contraction (big crunch) cycles leading to the formation of new star–planetary systems and gravitational interactions between masses. Mathematics inspired the sine cosine algorithm (SCA) [22], the Runge–Kutta optimizer (RUN) [23], and the arithmetic optimization algorithm (AOA) [24].

Tabu search (TS) [25], harmony search optimization (HS) [26], teaching–learning-based optimization (TLBO) [27], JAYA [28], the group teaching optimization algorithm (GTOA) [29], the mother optimization algorithm (MOA) [30], the preschool education optimization algorithm (PEOA) [31], the learning cooking algorithm (LCA) [32], the imperialist competitive algorithm (ICA) [33], and the political optimizer (PO) [34] are representative algorithms reproducing human activities, behaviors, learning/education processes, social sciences, international strategies, and politics.

Swarm intelligence-based algorithms represent the largest category in metaheuristic optimization. These algorithms mimic the social/individual behavior of animals (insects, terrestrial animals, birds and other flying animals, and aquatic animals) in reproduction, food search, hunting, migration, etc. Particle swarm optimization (PSO) [35], reproducing interactions between individuals of bird/fish swarms, is the most cited metaheuristic algorithm according to the Scopus database; in PSO, a population of candidate designs (the particles) are generated, and their positions and velocities are updated referring to the position of the leader(s) and the best positions of individual particles in each iteration. Insect behavior-inspired algorithms, for example, include ant colony optimization (ACO) [36], artificial bee colony (ABC) [37], the firefly algorithm (FFA) [38], and the ant lion optimizer (ALO) [39].

The grey wolf optimizer (GWO) [40], coyote optimization algorithm (COA) [41], snake optimizer (SO) [42], and snow leopard optimization algorithm (SLOA) [43] simulate the behavior of terrestrial animals. The GWO, reproducing the hunting behavior of grey

wolves, is the 2nd most cited algorithm in terms of citations/year after PSO. However, the GWO is preferred to PSO in view of its simpler formulation that does not include internal parameters except population size and a limit to the number of iterations.

The bat algorithm (BA) [44], cuckoo search (CS) [45], crow search algorithm (CSA) [46], Harris hawks optimization (HHO) [47], and starling murmuration optimizer (SMO) [48] are inspired by flying animals like bats and birds. CS (reproducing the parasitic behavior of some cuckoo species that mix their eggs with those of other birds to guarantee the survival of their chicks) and HHO (reproducing the cooperative behavior of Harris hawks in nature, specifically their surprise attacks during the chase) are the most cited MHAs of this sub-category.

Last, the dolphin echolocation algorithm (DEA) [49], whale optimization algorithm (WOA) [50], salp swarm algorithm (SSA) [51], marine predators algorithm (MPA) [52], and giant trevally optimizer (GTO) [53] reproduce social behavior, hunting strategy, swarming, and foraging of aquatic animals; WOA, SSA, and MPA are the most cited MHAs in this sub-category.

Hybrid/improved/enhanced methods also were developed in the optimization literature by adding new equations in the original formulation or merging two or more MHAs. The goal always was finding the best balance between exploration and exploitation to minimize computational cost (i.e., the number of structural analyses or function evaluations required by the optimizer), improve robustness, and limit the number of internal parameters of the metaheuristic formulation. Hybrid metaheuristic algorithms may have a parallel or a serial architecture [54]. In the former case, the component algorithms are independently run on parallel computers, while in the latter case, they are sequentially executed on the same machine.

Metaheuristic optimization algorithms are widely employed in various engineering fields such as, for example, the mechanical characterization of materials and structural identification [55], static and dynamic structural optimization [56,57], damage identification [58], vehicle routing optimization [59], 3D printing process optimization [60], cancer classification [61], and image processing [62]. However, in spite of the large diffusion of MHAs in engineering practice, some issues remain open in metaheuristic optimization: (i) no metaheuristic algorithm can always outperform all other MHAs in all optimization problems; (ii) MHAs may require a very large number of function evaluations (analyses) for completing optimization process; (iii) sophisticated algorithmic variants and hybrid MHAs combining multiple methods often increase the computational complexity of the optimizer also because of the presence of additional internal parameters that are difficult to be tuned. As a matter of fact, newly developed MHAs often added very little to the optimization field and stopped attracting potential users even after a rather short time from their release.

The main objective of this study was to develop an efficient and robust hybrid metaheuristic algorithm for engineering optimization. Generally speaking, a hybrid optimizer should combine high-performance MHAs that complement each other in terms of exploration and exploitation. Furthermore, component algorithms should be versatile so as to successfully deal with as many different optimization problems as possible. Lastly, component algorithms should be simple enough in order to simplify the formulation of the hybrid optimizer and to limit the number of additional internal parameters governing the switch from one component algorithm to another. In view of this, the grey wolf optimizer (GWO) and the JAYA algorithms were selected in the present study and combined into the novel FHGWJA hybrid algorithm (the acronym stands for Fast Hybrid Grey Wolf JAYA) because they certainly satisfy the above-mentioned requisites.

The GWO mimics the leadership hierarchy and hunting mechanism of grey wolves [40]. The leadership hierarchy is simulated by defining four types of grey wolves (α, β, δ, and ω). The hunting mechanism is simulated with different algorithmic operators that represent searching for prey, encircling prey, and attacking prey. The α wolf is the group leader. The β wolf helps the α in decision-making or other pack activities. The lowest ranking is

represented by ω wolves. The δ wolves rank between the α-β wolves and the ω wolves. Alpha, beta, and delta wolves estimate the position of the prey, and other wolves randomly update their positions around the prey.

As previously mentioned, the GWO is the 2nd most cited metaheuristic algorithm after PSO according to the Scopus database. It was successfully applied to many fields because of its simple formulation and computational efficiency (see, for example, the surveys on GWO applications presented in [63–65]). However, the GWO may suffer from lack of exploitation, risk of stagnation especially in complex problems, and limited adaptability to variations in the problem landscape during the optimization process. Enhanced GWO formulations including (i) new operators to capture other characteristic behaviors of wolves (i.e., gaze cues learning [64] and dimension learning-based hunting [66]), (ii) Cauchy-Gaussian mutation (increasing the search range of leader wolves when they tend to the local optimal solution) along with greedy selection (to maintain population diversity) and improved search strategy considering average position of all individuals [67], (iii) chaotic grouping and dynamic regrouping of individuals to increase population diversity [68], (iv) optimization of initial population along with a nonlinear control parameter convergence strategy and nonlinear tuning strategy of parameters [69], and (v) update of wolves' positions with spiral movements [70], as well as (vi) hybrid algorithms combining GWO with other powerful MHAs (i.e., particle swarm optimization, biogeography-based optimization, differential evolution, Harris hawks optimization, and the whale optimization algorithm [71–75]) were proposed in order to overcome the above-mentioned limitations.

JAYA [28] utilizes the most straightforward search scheme amongst all MHAs consisting of only one equation to perturb optimization variables: to approach the population's best solution and move away from the worst solution, thus achieving a significant convergence capability. JAYA is a very versatile algorithm with a large number of applications documented in the literature [76,77]. JAYA has an inherently hybrid nature combining basic features of evolutionary algorithms (the survival of fittest individual) and swarm-based algorithms where the swarm normally follows the leader in the search of the optimal solution. These characteristics make JAYA a very good candidate component of new hybrid metaheuristic algorithms.

Similar to the GWO, JAYA may present a rather weak exploitation phase. Furthermore, JAYA uses only one equation to perturb optimization variables involving only the best and the worst individuals of the population: this may produce stagnation and also limit the exploration phase. To overcome these issues, improved JAYA formulations using subpopulations [78], or involving also the current average solution and the historical solutions (a population of candidate solutions initially generated besides the standard population and permuted in the optimization process according to a probabilistic criterion) [79], were developed. Fuzzy clustering competitive learning, experience learning, and Cauchy mutation mechanisms were also implemented [80] to effectively utilize population information, speed up the convergence rate, improve the balance between exploiting the previously visited regions and exploring new regions of search space in the search process, and reduce the risk of being trapped into local optimum by fine-tuning the quality of the so-far best solution. The high computational cost is another issue in JAYA optimization. In [81,82], it was attempted to reduce the number of analyses and increase convergence speed by directly rejecting heavier designs than population individuals, but this strategy missed the global optimum in many structural optimization problems. Similarly, in [83], the population was updated not only if the new trial solutions improve the individuals currently stored in the population but also if they have the same values for the cost function or penalized cost function. In [84], generation of trial designs also relied on two randomly selected individuals $\vec{X_m}$ and $\vec{X_n}$, and the perturbation equation varied if $m > n$ or $m < n$ as well as if $W(\vec{X_m}) < W(\vec{X_n})$ or $W(\vec{X_m}) > W(\vec{X_n})$. JAYA was also hybridized with other efficient MHAs such as, for example, harmony search optimization [56], genetic algorithms [85], crow search [86], the Rao-1 optimizer [87], teaching–learning-based optimization [88], and differential evolution [89].

The main features of the novel FHGWJA algorithm developed in this study and the advancements with respect to the state-of-the-art can be summarized as follows:

(1) GWO and JAYA were simply merged into FHGWJA without complicating the algorithmic structure resulting from the hybridization process: this is a significant step further with respect to the above-mentioned studies where performance improvements achieved with the GWO and JAYA variants often were problem-sensitive and entailed complicated optimization formulations that were not easy to reproduce;
(2) FHGWJA utilizes elitist and repair strategies to generate high-quality candidate solutions that always have a significantly high probability of improving the current best record. The movements assigned to optimization variables in the perturbation process may be adjusted according to a rank-proportional variation strategy. This approach is more effective than those adopted in the GWO/JAYA variants proposed in the literature that do not guarantee each new trial solution be better than the current best record. Since elitist/repair strategies are utilized regardless of performing exploration or exploitation, FHGWJA actually optimizes the balance between these two phases;
(3) The positions of leading wolves are dynamically updated to avoid stagnation and convergence to local optima. For that purpose, FHGWJA utilizes a mirroring strategy based on the concept of descent direction, which is much simpler than the strategies documented in the literature;
(4) FHGWJA evaluates constraints only after having verified that the new trial design can effectively improve the current best record in the current or subsequent iterations. Should this not occur, repair strategies are activated. This allows for reduced computational costs of optimization to a great extent;
(5) FHGWJA does not require any internal parameters aside from population size and a limit to the number of iterations. The elitist strategies and convergence criterion implemented via FHGWJA actually make it unnecessary to specify the limit to the number of iterations.

The proposed FHGWJA algorithm was successfully tested in seven "real world" engineering problems. The selected test cases, including up to 29 optimization variables and 1200 nonlinear constraints, regarded, in particular, the following: (i–ii) 2D path planning (minimization of trajectory lengths, respectively, with 7 or 10 obstacles); (iii) shape optimization of a concrete gravity dam with additional earthquake load (volume minimization); (iv–v) calibration of nonlinear hydrologic models (the Muskingum problem solved with 3 or 25 unknown model parameters to be identified); (vi) optimal crashworthiness design of a vehicle subject to side impact; (vii) weight minimization of a planar 200-bar truss structure subject to three independent loading conditions. The present algorithm was compared with the best performing algorithms indicated in the literature for each test problem and advanced variants of state-of-the-art MHAs.

The rest of this manuscript is structured as follows. Section 2 describes the new hybrid optimization algorithm FHGWJA developed in this study. Test problems and optimization results are presented and discussed in Section 3. Section 4 summarizes the main findings of this study and outlines directions of future research.

2. The FHGWJA Algorithm

The new hybrid metaheuristic algorithm FHGWJA developed in this study is now described in detail. The algorithm is composed of seven steps: (i) initialization; (ii) preliminary definition of trial solutions with the classical GWO; (iii) rank-based refinement of preliminary trial solutions defined with the classical GWO; (iv) evaluation/repair of trial solutions with elitist strategies and JAYA-based schemes; (v) population reordering to define the new best and worst individuals and update of the α-β-δ wolves trying to avoid stagnation; (vi) convergence check; (vii) end of optimization search and output.

Step 1. Initialization

FHGWJA randomly generates a population of N_{POP} candidate designs (i.e., wolves) as:

$$x_j^i = x_j^L + \rho_j{}^i \ (x_j^U - x_j^L) \quad \begin{cases} j = 1, \ldots, NDV \\ i = 1, \ldots, N_{POP} \end{cases} \tag{1}$$

where NDV is the number of optimization variables; x_j^L and x_j^U, respectively, are the lower and upper bounds of the jth optimization variable; $\rho_j{}^i$ is a random number in the (0,1) interval.

In the minimization problem of the $W(\vec{X})$ function (depending on NDV variables stored in the solution vector $(\vec{X})$ subjected to N_{CON} inequality constraint functions $G_k(\vec{X}) \leq 0$, the penalized cost function $W_p(\vec{X})$ is defined as follows:

$$W_p(\vec{X}) = W(\vec{X}) + p \cdot \psi \tag{2}$$

where p is a penalty coefficient. The penalty function ψ is defined as follows:

$$\psi = \sum_{k=1}^{N_{con}} \left(\max(0, g_j)\right)^2 \tag{3}$$

The $W_p(\vec{X})$ penalized cost function coincides with the $W(\vec{X})$ cost function if the trial solution $\vec{X}$ satisfies optimization constraints. No penalty function is defined in the case of unconstrained optimization problems. Any equality constraint $H(\vec{X}) = 0$ included in the optimization problem is transformed into two inequality constraints $G(\vec{X}) \leq 0$ and $-G_k(\vec{X}) \leq 0$. Candidate solutions are sorted in terms of penalized cost function: the current best solution $\vec{X}_{best}$ and the worst solution $\vec{X}_{worst}$, respectively, correspond to the lowest and highest values of ψ.

Step 2. GWO phase: preliminary generation of new trial designs

FHGWJA utilizes the classical GWO scheme to preliminarily generate the new trial designs. The best three individuals stored in the population are ranked as wolves α, β, and δ. The other individuals of the population are ranked as wolves ω. Wolves encircle the prey during the hunt. Such a behavior is mathematically described as follows:

$$\vec{D} = \left| \vec{C} \cdot \vec{X}_p \right| - \vec{X}_{it} \tag{4}$$

$$\vec{X}_{it+1} = \vec{X}_p - \vec{A} \cdot \vec{D} \tag{5}$$

In Equations (4) and (5), it denotes the current iteration, $\vec{A}$ and $\vec{C}$ are coefficient vectors, $\vec{X}_p$ is the position vector of the prey, and $\vec{X}_{it}$ is the generic grey wolf (i.e., search agent) of the population updated in the current iteration to the new solution $\vec{X}_{it+1}$. The "·" notation denotes the term-by-term multiplication between vectors that leads to defining another vector.

Vectors $\vec{A}$ and $\vec{C}$ are defined as follows:

$$\vec{A} = 2\vec{a} \cdot \vec{r_1} - \vec{a} \tag{6}$$

$$\vec{C} = 2\vec{r_2} \tag{7}$$

In Equations (6) and (7), the components of the $\vec{a}$ vector decrease linearly from 2 to 0 as the optimization progresses; $\vec{r_1}$ and $\vec{r_2}$ are random vectors in the [0,1] interval.

While grey wolves actually recognize the prey's location in the hunting phase, the optimum position (prey) usually is not known a priori in the optimization search. In order to reproduce wolves' behavior in the GWO algorithm, it is assumed that the three best solutions of the population (i.e., α, β, and δ wolves) have a better knowledge of the potential prey's location (i.e., the target optimal solution). The rest of population must be updated using the $\vec{X_\alpha}$, $\vec{X_\beta}$, and $\vec{X_\delta}$ positions of the best three search agents. The generic solution $\vec{X_i}$ of population is updated to $\vec{X_{i,tr}}$ as follows:

$$\begin{cases} \vec{D_\alpha} = \left| \vec{C_1} \cdot \vec{X_\alpha} \right| - \vec{X_i} \\ \vec{D_\beta} = \left| \vec{C_2} \cdot \vec{X_\beta} \right| - \vec{X_i} \\ \vec{D_\delta} = \left| \vec{C_3} \cdot \vec{X_\delta} \right| - \vec{X_i} \end{cases} \tag{8}$$

$$\begin{cases} \vec{X_1} = \vec{X_\alpha} - \vec{A_1} \cdot \vec{D_\alpha} \\ \vec{X_2} = \vec{X_\beta} - \vec{A_2} \cdot \vec{D_\beta} \\ \vec{X_3} = \vec{X_\delta} - \vec{A_3} \cdot \vec{D_\delta} \end{cases} \tag{9}$$

$$\left(\vec{X_{i,tr}} \right)^{prel} = \frac{\vec{X_1} + \vec{X_2} + \vec{X_3}}{3} \tag{10}$$

In Equation (9), vectors $\vec{X_1}$, $\vec{X_2}$, and $\vec{X_3}$ are defined for each candidate design $\vec{X_i}$ of the population. Equations (8)–(10) update the positions of all search agents.

In the optimization process, the wolves α, β, and δ estimate the prey's position (i.e., the position of the optimal solution) and each candidate solution in the population updates its distance from the prey. The parameter a is reduced from 2 to 0 to emphasize exploration and exploitation, respectively. In the exploration phase, candidate solutions tend to search for another prey when $\|\vec{A}\| > 1$. Conversely, in the exploitation phase, candidate solutions converge towards the prey when $\|\vec{A}\| < 1$.

Step 3. Rank-based refinement of preliminary trial designs

In the classical GWO, the new population formed by the trial solutions $\vec{X_{i,tr}}$ obtained by perturbing the individuals $\vec{X_i}$ stored in the population in the previous iteration are sorted in terms of penalized cost function values to update positions $\vec{X_\alpha}$, $\vec{X_\beta}$, and $\vec{X_\delta}$. The process ends upon reaching the limit to the number of iterations or function evaluations defined by the user. However, the classical GWO search scheme has two inherent drawbacks: (i) since the position of the prey (optimal solution) is not known a priori, the search is directly biased toward the α wolf that has a null distance from the current best record X_{best} as it corresponds to the current best record itself in view of population sorting; (ii) there is no guarantee that each new trial solutions $\vec{X_{i,tr}}$ may improve the current best record $\vec{X_{best}}$ or at least the corresponding individual $\vec{X_i}$ stored in the previous population.

In order to solve these issues, FHGWJA refines the preliminary trial solutions $(\vec{X_{i,tr}})^{prel}$ obtained in Step 1 by means of a rank-based criterion. The population defined in the previous iteration is sorted with respect to the penalized cost function values $W_p(\vec{X_i})$. The best solution is assigned the rank 1 while the worst solution is assigned the rank $1/N_{\mathrm{POP}}$: let $rank^{(\mathrm{i})} = 1/i$ be the rank of the generic individual $\vec{X_i}$. The average rate of variation for the penalized cost function with respect to the current best record $\gamma^{(\mathrm{i})} = \Delta W_p{}^{(\mathrm{i})} / \Delta S^{(\mathrm{i})}$ is evaluated for each solution $\vec{X_i}$. The $\Delta W_p{}^{(\mathrm{i})} = [W_p(\vec{X_i}) - W_p(\vec{X_{best}})]$ numerator represents

the increase in the penalized cost function that occurs when the design is perturbed by $\Delta S^{(\mathrm{i})} = \|\vec{X}_i - \vec{X}_{best}\|$ from the current best record to the generic candidate solution $\vec{X}_i$. The average rate of variation $\gamma^{(\mathrm{aver})} = \sum_{\mathrm{i}=1}^{\mathrm{N_{POP}}} \gamma^{(\mathrm{i})} / \mathrm{N_{POP}}$ is hence defined. The refinement step size $\mu^{(\mathrm{i})}$ is defined for each individual $\vec{X}_i$ as follows:

$$\mu^{(\mathrm{i})} = Min\left\{\left(1 - rank^{(\mathrm{i})}\right); \left(\frac{\gamma^{(\mathrm{i})}}{\gamma^{(\mathrm{aver})}}\right)\right\} \tag{11}$$

The preliminary trial solution $\left(\vec{X}_{i,tr}\right)^{prel}$ is adjusted to the trial solution $\vec{X}_{i,tr}$ using the following equation:

$$\vec{X}_{i,tr} = (\vec{X}_{i,tr})^{prel} + \frac{(\vec{X}_{best} - \vec{X}_i)}{\|\vec{X}_{best} - \vec{X}_i\|}\mu^{(\mathrm{i})} \tag{12}$$

The rationale behind Equations (11) and (12) is the following. All trial solutions generated via the GWO must be evaluated to see if they improve design. Individuals ranked as high-quality solutions in the previous iteration are very likely to preserve their rank in the current iteration. However, rank may decrease if the cost function changes sharply in the neighborhood of the currently perturbed solution: for example, this may occur if the perturbed solution turns infeasible, and the penalty term increases. For this reason, Equation (11) tends to preserve the overall ranks of population individuals by setting small refinement step sizes $\mu^{(\mathrm{i})}$. Since the design improves by moving from $\vec{X}_i$ to $\vec{X}_{best}$, the direction $\vec{S}_i = (\vec{X}_{best} - \vec{X}_i)$ is a descent direction with respect to the $\vec{X}_i$ individual. Hence, FHGWJA attempts to perturb design along the $\vec{S}_i$ direction to further reduce the cost function at least with respect to $\vec{X}_i$.

Step 4. Evaluation and modification/repair of new trial designs: elitist strategies and JAYA scheme

The trial designs $\vec{X}_{i,tr}$ generated via FHGWJA in Steps 1 and 2 are then evaluated. In the classical GWO, if the new design $\vec{X}_{i,tr}$ improves the old design $\vec{X}_i$, it replaces $\vec{X}_i$ in the new population. However, this task entails a new constraint evaluation to compute the penalized cost function value for each new trial design.

In order to reduce the number of constraint evaluations required in the optimization search, FHGWJA utilizes an elitist strategy retaining only the trial solutions that are very likely to improve design. Hence, FHGWJA initially compares only the cost function $W(\vec{X}_{i,tr})$ computed for the new trial design $\vec{X}_{i,tr}$ with the penalized cost function $W_p(\vec{X}_{best})$ computed for the current best record. If $W(\vec{X}_{i,tr}) > W_p(\vec{X}_{best})$, the new trial design $\vec{X}_{i,tr}$ is directly rejected because it certainly cannot improve the current best record as the cost function value $\vec{X}_{i,tr}$ is by itself larger than the penalized cost of $\vec{X}_{best}$ which also accounted for any constraint violation.

If $W(\vec{X}_{i,tr}) < W_p(\vec{X}_{best})$, the cost function value $W(\vec{X}_{i,tr})$ computed for the new trial solution $\vec{X}_{i,tr}$ is compared with 1.1 times the cost function value $W(\vec{X}_{best})$ computed for the current best record. Ifit holds $W(\vec{X}_{i,tr}) \leq 1.1W(\vec{X}_{best})$, the new trial solution $\vec{X}_{i,tr}$ is provisionally included in the new population. The $1.1W(\vec{X}_{best})$ threshold was selected upon the following rationale. Exploration is characterized by a high probability of improving design: hence, the $W(\vec{X}_{i,tr}) > 1.1W(\vec{X}_{best})$ condition is not likely to occur. In exploitation, local minima should be bypassed by the optimizer to converge to the global optimum. Similar to SA, FHGWJA is allowed to provisionally accept slightly worse candidate solutions

than $\vec{X}_{best}$. The threshold level of acceptance probability in state-of-the-art SA formulations such as [81,90] is 0.9 if the ratio between the cost function increment $[W(\vec{X}_{i,tr}) - W(\vec{X}_{best})]$ and the annealing temperature T is 0.1. This means that there is a 90% probability to provisionally accept a worse design than $\vec{X}_{best}$ and make it improve in the next iterations. Since T is initially set equal to the expected optimum cost (or roughly corresponds to the cost function value $W(\vec{X}_{best})$), it is reasonable to assume that solutions up to 10% worse than $\vec{X}_{best}$ may improve $\vec{X}_{best}$ itself in the next few iterations.

Since each new trial solution $\vec{X}_{i,tr}$ provisionally included in the new population was generated by FHGWJA using the classical GWO scheme based on the positions of wolves α, β, and δ, FHGWJA adopts a JAYA-based scheme to avoid stagnation. For that purpose, when $W(\vec{X}_{i,tr}) \leq 1.1W(\vec{X}_{best})$, a new trial solution $(\vec{X}_{i,tr})'$ is defined as follows:

$$(\vec{X}_{i,tr})' = \vec{X}_{i,tr} + \vec{\lambda}_1 \cdot(\vec{X}_{best} - \vec{X}_{i,tr}) - \vec{\lambda}_2 \cdot(\vec{X}_{\delta} - \vec{X}_{i,tr}) \quad (13)$$

where $\vec{\lambda}_1$ and $\vec{\lambda}_2$ are two vectors of NDV random numbers generated in the [0,1] interval. Equation (13) is relative to exploitation as it perturbs the good design $\vec{X}_{i,tr}$ that was provisionally included in the new population being close to the current best record or likely better than it. The δ wolf is temporarily selected as the worst individual of the population, and FHGWJA is forced to locally search in a region of design space containing only high-quality solutions.

The new trial solution $(\vec{X}_{i,tr})'$ is compared with $\vec{X}_{i,tr}$ to select the solution that finally updates population. If $W((\vec{X}_{i,tr})') < 1.1W(\vec{X}_{best})$ and $W((\vec{X}_{i,tr})') < W(\vec{X}_{i,tr})$, $(\vec{X}_{i,tr})'$ is finally retained in the new population, and $\vec{X}_{i,tr}$ is rejected. If $W((\vec{X}_{i,tr})') < 1.1W(\vec{X}_{best})$ but it occurs that $W((\vec{X}_{i,tr})') > W(\vec{X}_{i,tr})$, $(\vec{X}_{i,tr})'$ is rejected, and $\vec{X}_{i,tr}$ is finally retained in the new population. If $W((\vec{X}_{i,tr})') > 1.1W(\vec{X}_{best})$, $\vec{X}_{i,tr}$ is finally retained in the new population, and $(\vec{X}_{i,tr})'$ is directly rejected.

FHGWJA adopts a repair strategy if the new trial solution $\vec{X}_{i,tr}$ is such that $W(\vec{X}_{i,tr}) > 1.1W(\vec{X}_{best})$. In this case, the trial solution $\vec{X}_{i,tr}$ was sufficiently good to avoid immediate rejection (i.e., it holds $W(\vec{X}_{i,tr}) < W_p(\vec{X}_{best})$) but it is not good enough to be provisionally included in the new population. Another JAYA-based scheme is adopted in FHGWJA to define the new trial solution $(\vec{X}_{i,tr})'$ as follows:

$$(\vec{X}_{i,tr})' = \vec{X}_i + \vec{\lambda}_1 \cdot(\vec{X}_{best} - \vec{X}_i) - \vec{\lambda}_2 \cdot(\vec{X}_{worst} - \vec{X}_i) \quad (14)$$

where $\vec{\lambda}_1$ and $\vec{\lambda}_2$ are two vectors of NDV random numbers similar to those of Equation (13). Absolute values of optimization variables are taken for each component of vectors $\vec{X}_i$ or $\vec{X}_{i,tr}$, respectively, in Equations (13) and (14).

Equation (14) is related to exploration and resembles the classical JAYA solution updating equation where each individual is perturbed trying to approach the current best solution $\vec{X}_{best}$ and moving away from the worst solution of the population $\vec{X}_{worst}$. FHGWJA involves the whole population of N_{POP} search agents in the generation of new trial design $(\vec{X}_{i,tr})'$ because wolves α, β, and δ that drive search in the GWO phase failed to move other individuals (i.e., $\vec{X}_i$) to good positions of the search space located near the prey (i.e., the current best record). Since population renewal aims to improve each individual $\vec{X}_i$, FHGWJA searches on the descent direction $(\vec{X}_{best} - \vec{X}_i)$ that leads to a better design

than $\vec{X}_i$ and escape from the worst design of the population $\vec{X}_{worst}$, which certainly cannot improve $\vec{X}_i$.

Similar to the process implemented for Equation (13), FHGWJA compares the cost function value $W((\vec{X}_{i,tr})')$ corresponding to the new trial design $(\vec{X}_{i,tr})'$ defined with Equation (14) with 1.1 times the cost function value of the current best record $W(\vec{X}_{best})$. The modified trial design $(\vec{X}_{i,tr})'$ is included in the new population if it holds that $W((\vec{X}_{i,tr})') \leq 1.1W(\vec{X}_{best})$.

However, if the JAYA-based repair strategy fails and it still holds that $W((\vec{X}_{i,tr})') > 1.1W(\vec{X}_{best})$, FHGWJA has to deal with three solutions, $\vec{X}_i$, $\vec{X}_{i,tr}$, and $(\vec{X}_{i,tr})'$, that do not improve the current best record $\vec{X}_{best}$. Figure 1 shows the mutual positions of the cost function gradient $\nabla\vec{W}_{\text{best}}$ evaluated at the current best record $\vec{X}_{best}$ and the non-descent directions $\vec{S}_i = \vec{X}_i - \vec{X}_{best}$, $\vec{S}_{i,tr} = \vec{X}_{i,tr} - \vec{X}_{best}$, and $(\vec{S}_{i,tr})' = (\vec{X}_{i,tr})' - \vec{X}_{best}$; P_{best} is the point of the search space corresponding to the current best record $\vec{X}_{best}$ while the P_{i}, $\text{P}_{\text{i,tr}}$, and $\text{P}_{\text{i,tr}}'$ points, respectively, correspond to trial solutions $\vec{X}_i$, $\vec{X}_{i,tr}$, and $(\vec{X}_{i,tr})'$. It can be seen from Figure 1 that non-descent directions make positive scalar products with the gradient $\nabla\vec{W}_{\text{best}}$. Design may be improved with respect to $\vec{X}_{best}$ by moving on the descent directions $\vec{S}_i^{(mirr)} = -(\vec{X}_i - \vec{X}_{best})$, $\vec{S}_{i,tr}^{(mirr)} = -(\vec{X}_{i,tr} - \vec{X}_{best})$, and $(\vec{S}_{i,tr})'^{(mirr)} = (\vec{X}_{i,tr})' - \vec{X}_{best}$ that are, respectively, opposite to the non-descent directions $\vec{S}_i$, $\vec{S}_{i,tr}$, and $(\vec{S}_{i,tr})'$ and hence make negative scalar products with respect to $\nabla\vec{W}_{\text{best}}$ (see Figure 1). Hence, FHGWJA uses a "mirroring" strategy to define the new trial solution $(\vec{X}_{i,tr})''^{(mirr)}$ as follows:

$$(\vec{X}_{i,tr})''^{,mirr} = \vec{X}_{best} + \eta_{mirr,i}\vec{S}_i^{(mirr)} + \eta_{mirr,i-tr}\vec{S}_{i,tr}^{(mirr)} + \eta_{mirr,(i-tr)'}(\vec{S}_{i,tr})'^{(mirr)} \quad (15)$$

where $\eta_{\text{mirr,i}}$, $\eta_{\text{mirr,i-tr}}$, and $\eta_{\text{mirr,(i-tr)'}}$ are three random numbers in the [0,1] interval. Basically, FHGWJA takes the descent direction $((\vec{X}_{i,tr})''^{(mirr)} - \vec{X}_{best})$ obtained by combining three other descent directions that may improve design. Hence, the $(\vec{X}_{i,tr})''^{(mirr)}$ trial solution (corresponding to the point $\text{P}_{\text{i,tr}}''^{(\text{mirr})}$ of search space shown in Figure 1) is very likely to improve design. The new solution $(\vec{X}_{i,tr})''^{(mirr)}$ is evaluated as outlined above for the other trial solutions generated by FHGWJA.

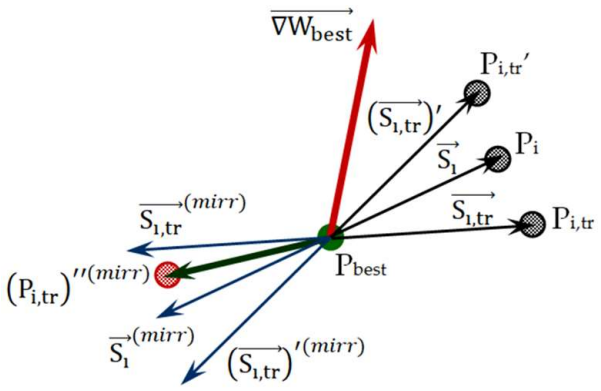

Figure 1. Schematic of the "mirroring" strategy used in FHGWJA if trial solutions $\vec{X}_i$, $\vec{X}_{i,tr}$, and $(\vec{X}_{i,tr})'$ do not improve the current best record.

Step 5. Resort population, update α, β, and δ wolves, $\vec{X}_{best}$, and $\vec{X}_{worst}$

Once FHGWJA updated all candidate solutions $\vec{X}_i$, the population is resorted and the N_{POP} individuals are ranked with′ respect to their penalty function values. Should $W_p(\vec{X}_{i,tr})$ or $W_p((\vec{X}_{i,tr})'$ or $W_p((\vec{X}_{i,tr})')$ be greater than $W_p(\vec{X}_i)$, all new trial solutions generated/refined for $\vec{X}_i$ are rejected and the old design $\vec{X}_i$ is retained in the population. However, the elitist strategies based on the $1.1W(\vec{X}_{best})$ threshold, JAYA-based schemes of Equations (13) and (14), and the "mirroring" strategy of Equation (15) greatly increase the probability of updating the whole population with better designs than those stored in the previous population.

The best three individuals of the new population are reset as wolves α, β, and δ with the corresponding design vectors $\vec{X}_\alpha$, $\vec{X}_\beta$, and $\vec{X}_\delta$. The best and worst solutions are set as $\vec{X}_{best}$ and $\vec{X}_{worst}$, respectively.

In order to avoid stagnation, ranking of wolves α, β, and δ is checked with another elitist criterion if FHGWJA did not update their positions in the current iteration. The criterion is again based on the concept of descent direction. The $\vec{S}_\beta = (\vec{X}_{best} - \vec{X}_\beta)$ and $\vec{S}_\delta = (\vec{X}_{best} - \vec{X}_\delta)$ obviously are descent directions with respect to positions $\vec{X}_\beta$ and $\vec{X}_\delta$ of wolves β and δ, and the design improves by moving towards the α wolf. Hence, FHGWJA perturbs $\vec{X}_\beta$ and $\vec{X}_\delta$ along the descent directions $\vec{S}_\beta$ and $\vec{S}_\delta$. The positions of wolves β and δ are "mirrored" with respect to $\vec{X}_{best}$ as follows:

$$\begin{cases} (\vec{X}_\beta)^{mirr} = (1 + \eta_{mirr,\beta})\vec{X}_{best} - \eta_{mirr,\beta}\vec{X}_\beta \\ (\vec{X}_\delta)^{mirr} = (1 + \eta_{mirr,\delta})\vec{X}_{best} - \eta_{mirr,\delta}\vec{X}_\delta \end{cases} \quad (16)$$

where $\eta_{mirr,\beta}$ and $\eta_{mirr,\delta}$ are two random numbers in the interval (0,1) that limit step sizes to reduce the probability of generating infeasible positions. The best three positions amongst $\vec{X}_\alpha$, $\vec{X}_\beta$, $\vec{X}_\delta$, $(\vec{X}_\beta)^{mirr}$, and $(\vec{X}_\delta)^{mirr}$ are taken as wolves α, β, and δ for the next iteration, while the two worst positions are compared with the rest of the population and may replace $\vec{X}_{worst}$ and the 2nd worst design of the old population. The latter operation also covers the case that $(\vec{X}_\beta)^{mirr}$ and $(\vec{X}_\delta)^{mirr}$ did not improve any of the wolves α, β, and δ.

The mirror strategy used in FHGWJA is illustrated in Figure 2. In particular, the figure shows the following: (i) the original positions of wolves α, β, and δ, respectively, correspond to points $P_{opt} \equiv P_\alpha$, P_β, and P_δ of the search space; (ii) the positions of "mirror" wolves β^{mirr} and δ^{mirr} are defined with Equation (16) and, respectively, correspond to points $P_{\beta,mirr}$ and $P_{\delta,mirr}$ of the search space; (iii) the cost function gradient vector $\nabla\vec{W}_{best}$ is evaluated at $\vec{X}_{best}$. It can be seen that the "mirror" wolves β^{mirr} and δ^{mirr} defined by Equation (16) lie on descent directions and may even improve $\vec{X}_{best}$, the position of wolf α. In fact, it holds that $\nabla\vec{W}_{best} \times ((\vec{X}_\beta)^{mirr} - \vec{X}_{opt}) < 0$ and $\nabla\vec{W}_{best} \times ((\vec{X}_\delta)^{mirr} - \vec{X}_{opt}) < 0$, where "×" denotes the scalar product between two vectors. Interestingly, since $((\vec{X}_\delta)^{mirr} - \vec{X}_{opt})$ is in principle a steeper descent direction than $((\vec{X}_\beta)^{mirr} - \vec{X}_{opt})$ as $W_p(\vec{X}_\delta) > W_p(\vec{X}_\beta) > W_p(\vec{X}_{best})$, wolf δ may have a higher probability than wolf β to replace wolf α even though it originally occupied a worse position than wolf β in the search space. This elitist approach forces FHGWJA to carry out a new exploration of the search space rather than trying to exploit solutions that have not improved the design in the last iteration. Furthermore, by eliminating the two worst designs of the population, FHGWJA increases the average quality of candidate solutions and achieves a higher probability of generating high-quality trial solutions in the next iteration.

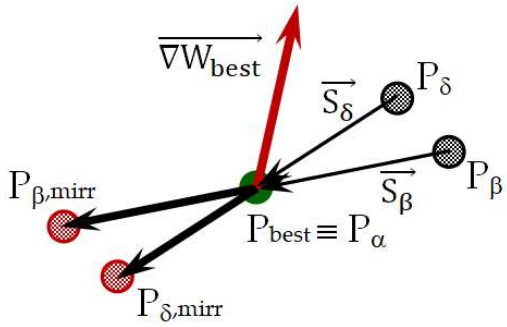

Figure 2. Schematic of the elitist mirror strategy used in FHGWJA to avoid stagnation of wolves α, β, and δ.

Step 6. Check for convergence

Standard deviations on optimization variables and penalty function values of candidate solutions included in the updated population should decrease as the optimization search approaches the global optimum. Hence, FHGWJA normalizes standard deviations with respect to average design $\overrightarrow{X_{aver}} = \left(\sum_{i=1}^{N_{POP}} \overrightarrow{X_i}\right) / N_{POP}$ and average value of penalty function $W_{p,aver} = \left(\sum_{i=1}^{N_{POP}} W_p\left(\overrightarrow{X_i}\right)\right) / N_{POP}$. The termination criterion is as follows:

$$Max\left\{ \begin{array}{l} \dfrac{STD\left\{\left\|\overrightarrow{X_1}-\overrightarrow{X_{aver}}\right\|, \left\|\overrightarrow{X_2}-\overrightarrow{X_{aver}}\right\|, \ldots, \left\|\overrightarrow{X_{NPOP}}-\overrightarrow{X_{aver}}\right\|\right\}}{\left\|\overrightarrow{X_{aver}}\right\|}; \\ \dfrac{STD\left\{W_{p,1}, W_{p,2}, \ldots, W_{p,NPOP}\right\}}{W_{p,aver}} \end{array} \right\} \leq \varepsilon_{conv} \quad (17)$$

where the convergence limit ε_{conv} is equal to 10^{-7}. Since convergence occurs in the last stage of the optimization process when exploitation dominates, Equation (17) requires all search agents be actually located in the best region of the search space hosting the global optimum and contributing to exploitation. Population diversity must decrease to aggregate search agents in the neighborhood of $\overrightarrow{X_{best}}$. Hence, Equation (17) normalizes the standard deviation of the search agents' positions to average the solution to quantify the level of population diversity. The same rationale is adopted for the penalty function values by normalizing their standard deviation with respect to the average penalty function value: hence, competitive solutions correspond to the optimum solution only when they are effectively close to it, that is when the search process is near to end. Penalty function values are considered in Equation (17) to account for the effect of some solution that may turn locally infeasible if the set of active constraints changes as design is perturbed in the neighborhood of the global optimum. Penalty function values are equal to cost function values for feasible solutions or in the case of unconstrained optimization problems.

Steps 2 through 6 are repeated until FHGWJA converges to the global optimum.

Step 7. Terminate optimization process

FHGWJA terminates the optimization process and writes the output data (i.e., optimum design and optimized cost function value) in the results file.

Figure 3 shows the flow chart of the novel FHGWJA algorithm developed in this study. The present algorithm actually is an advanced grey wolf optimizer, which updates population taking care that the α, β, and δ wolves effectively lead to improve the current best record in each iteration. The JAYA schemes and the elitist strategies implemented in FHGWJA enhance exploration and exploitation forcing the present algorithm to increase diversity and select high-quality trial designs without performing too many function evaluations. The definition of all descent directions utilized in FHGWJA in the optimization search does not entail any new constraint evaluation, and all decisions are made in FHGWJA

by comparing only cost function values. Interestingly, FHGWJA does not need any new internal parameters with respect to the classical GWO and JAYA formulations that require users to specify population size N_{POP} and limit number of iterations $N_{itermax}$. This feature of FHGWJA is not very common as simple hybrid metaheuristic algorithms may adopt a set of new heuristic parameters to switch from one component optimizer to the other. However, FHGWJA implements a master–slave algorithmic architecture where the slave algorithm, JAYA, is utilized only to refine or correct the trial designs generated by the master algorithm, GWO.

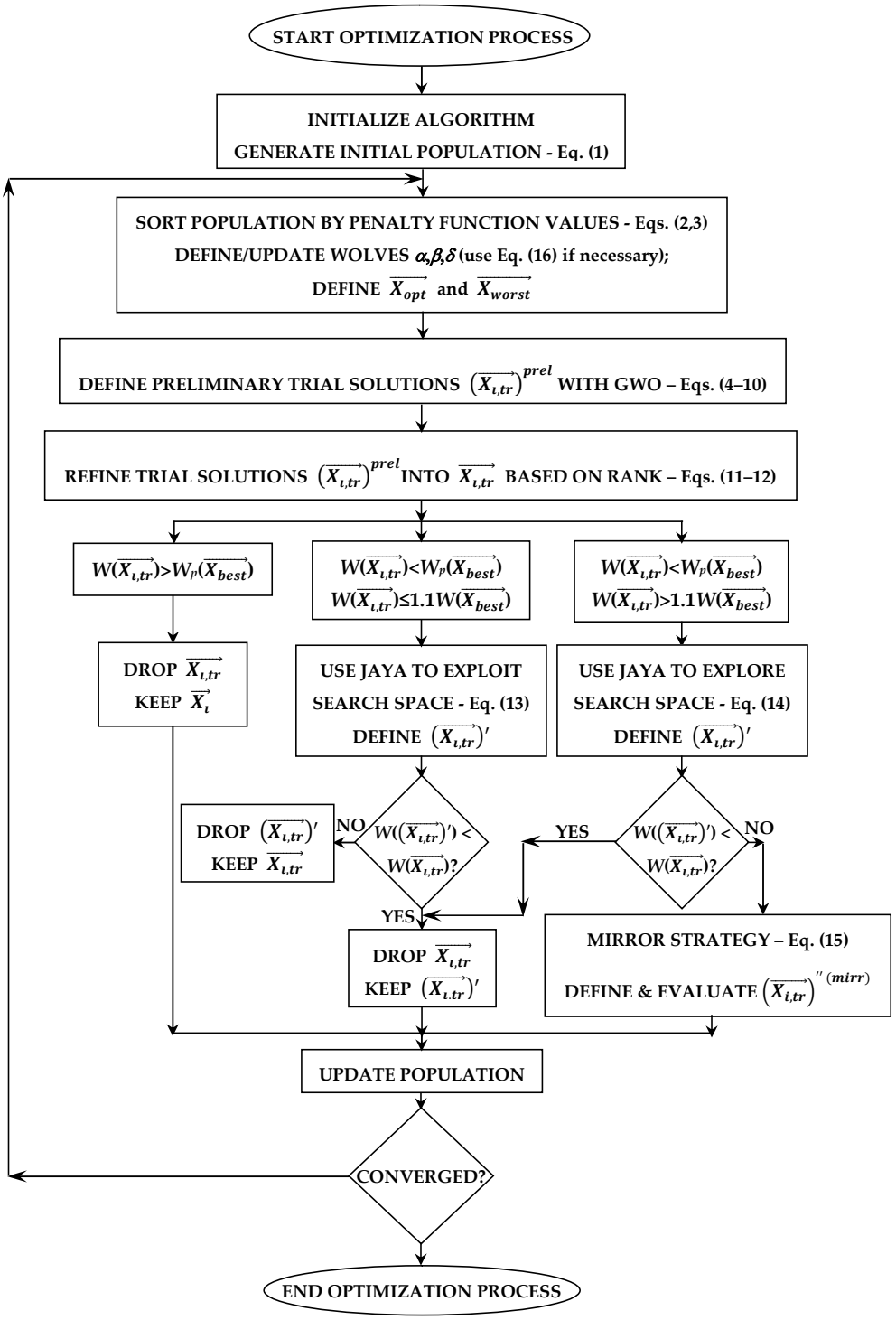

Figure 3. Flowchart of the FHGWJA algorithm developed in this study.

The computational cost of most GWO/JAYA implementations documented in the literature was $N_{POP} \times N_{itermax}$ function evaluations (also indicated as function calls or analyses for general problems or structural analyses for structural optimization problems) including constraint evaluations. The FHGWJA's ability of generating high-quality trial solutions throughout the optimization process significantly reduces the computational cost of the optimization. As far as population size N_{POP}, it is well known that increasing N_{POP} in metaheuristic optimization may enhance exploration capabilities but also results in a very large number of function evaluations. Indeed, it is not necessary to work with a large population if the search engine can always generate high-quality designs that improve the current best record or the currently perturbed search agents such as is accomplished with FHGWJA. It should be noted that grey wolves hunt in nature in groups comprising, at most, 10–20 individuals (the typical family pack is formed by 5–11 animals but in some cases the pack can be formed by 2–3 families). In view of this, FHGWJA optimizations carried out in this study utilized a population of 10 individuals. Sensitivity analysis confirmed the validity of this setting.

3. Test Problems and Optimization Results

The FHGWJA algorithm proposed in this study was tested in seven engineering problems including up to 29 optimization variables and 1200 nonlinear constraints. Test cases regarded the following: (i–ii) 2D path planning (minimization of trajectory lengths with 7 or 10 obstacles); (iii) shape optimization of a concrete gravity dam with additional earthquake load (volume minimization); (iv–v) calibration of nonlinear hydrologic models (the Muskingum problem solved with 3 or 25 unknown model parameters to be identified); (vi) optimal crashworthiness design of a vehicle subject to side impact; (vii) weight minimization of a planar 200-bar truss structure under three independent loading conditions.

FHGWJA was implemented in MATLAB as a standalone optimization code. Optimization runs were carried out on a standard PC equipped with a 3.1 Mhz AMD processor and 16 MB of RAM memory. The Mersenne-Twister (MT19937) MATLAB default algorithm was utilized to generate random numbers uniformly distributed between 0 and 1; this algorithm was selected in view of its ability to generate high-quality numbers with (2^{19937}–1) long period. In order to draw statistically significant conclusions on the computational efficiency of FHGWJA, 20 independent optimization runs were executed for each test problem starting from different initial populations. The population size and limit number of iterations of FHGWJA were, respectively, set equal to 10 and 5000. Initial populations included some designs with up to 1000% constraint violation to check whether FHGWJA can quickly approach the best region of search space containing the optimum solution. Remarkably, FHGWJA always completed all optimization runs within a much lower number of analyses than the theoretical computational budget of $10 \times 5000 = 50{,}000$ analyses of the classical GWO and JAYA. The actual computational cost of FHGWJA was almost entirely due (about 90%) to the number of analyses performed in the optimization search. For example, the optimization runs performed with FHGWJA for the largest test problem solved in this study (weight minimization of a planar 200-bar truss structure including 29 sizing variables) were completed on average within about 25 min of wall clock time.

3.1. 2D Path Planning

Test cases (i–ii) regarded a classical robotics problem: to determine the optimal 2D path connecting two points A and B with a piecewise cubic spline trajectory, defined by a certain number of control points (*NCP*), that passes through points A, B, and a series of base points (*NBP*) while avoiding a set of obstacles (*NOB*). The problem objective is to minimize the total length of the trajectory. In an obstacle-free environment, the shortest path between A and B obviously is a straight line. Cubic splines allow for building a smooth and continuous curved path avoiding obstacles. Path planning complexity increases with the number of obstacles and the number of base points; coordinates of base points are selected as optimization variables.

The 2D path planning problem searching for the minimum length of a 2D trajectory, passing through *NBP* base points so as to avoid *NOB* obstacles, and defined by *NCP* control points connected by (*NCP*−1) spline segments, can be formulated as follows:

$$Minimize\ W(\vec{X}) = \sum_{r=1}^{NCP} \sqrt{(xc_{r+1} - xc_r)^2 + (yc_{r+1} - yc_r)^2} + PENALTY \tag{18}$$

$$Subject\ to \begin{cases} G(\vec{X}) = \frac{rad_{ob}(p)}{\sqrt{(xc_r - xob_p)^2 + (yc_r - yob_p)^2}} - 1 < 0 \\ G(\vec{X}) = \frac{rad_{ob}(p)}{\sqrt{(xb_i - xob_p)^2 + (yb_i - xob_p)^2}} - 1 < 0 \end{cases} \begin{cases} i = 1, NBP \\ p = 1, \ldots, NOB \\ r = 1, \ldots, NCP \end{cases} \tag{19}$$

In Equations (18) and (19), (xb_i,yb_i) are the coordinates of the generic *i*-th base point of the trajectory, (xc_r,yc_r) are the coordinates of the generic *r*-th control point of the trajectory, $rad_{ob}(p)$ is the radius of the circular area limited by the *p*-th obstacle, respectively. The coordinates of base points are taken as optimization variables.

In general, the optimization problem stated above includes 2·*NBP* design variables and (*NCP* + *NBP*) × *NOB* nonlinear constraints. Coordinates of base points must also satisfy the relationships $(xb_i - xb_{i+1}) \leq 0$ and $(yb_i - yb_{i+1}) \leq 0$ (*i* = 1,. . .,*NBP*−1) to preserve the local convexity of the trajectory. The "*PENALTY*" term in Equation (18) aggregates squared violations of constraint inequalities occurring for $G(\vec{X}) > 0$ multiplied by 1000 to amplify the effect of collision with obstacles; it is equal to 0 if the optimizer finds a feasible solution where the designed trajectory does not intersect the *NOB* obstacles. More details on the formulation of path planning problems in robotics can be found in Refs. [91–93].

Figure 4 shows the design space of the two path planning problems (all coordinates are expressed in mm) solved in this study considering the presence of 7 or 10 obstacles and the same number of base points. In the former case, there are the start/end points A(1,1) and B(29,29) mm to be connected and the seven obstacles, O_1(6,6), O_2(7,24), O_3(9,15), O_4(16,23), O_5(20,11), O_6(23,6), and O_7(24,23) mm, limiting circular areas of radii 3, 1, 3, 2, 2, 1, and 2 mm, respectively. The optimization problem hence includes 14 variables corresponding to the coordinates of base points (i.e., 2·*NBP* = 2 × 7) that can vary between 1 and 29 mm, *NOB* = 7 obstacles, *NCP* = 100 control points, and 749 nonlinear constraints (i.e., (*NCP*+*NBP*)× *NOB* = (100 + 7)× 7 = 749).

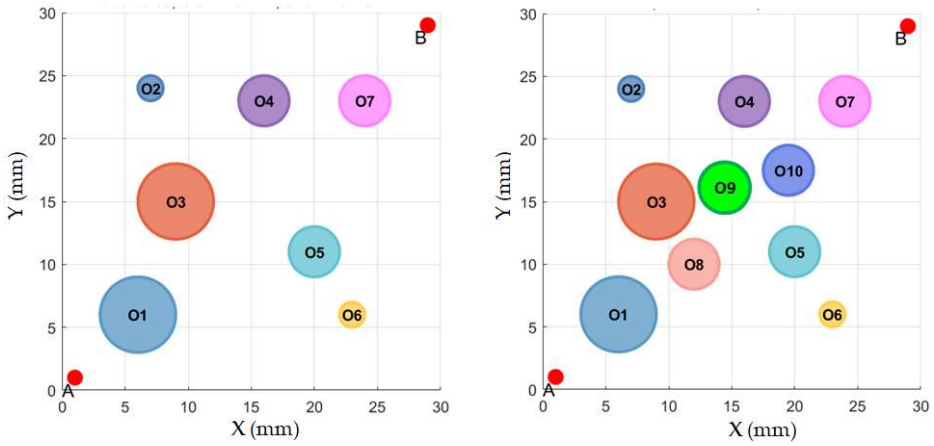

Figure 4. Trajectory spaces of the 2D path planning problem variants including 7 and 10 obstacles.

In the latter case, there are three additional obstacles O_8(12,10), O_9(14,16), and O_{10}(19.5,17.5) mm all limiting circular areas of radius 2 mm. The optimization problem hence includes 20 variables (i.e., 2·*NBP* = 2 × 10), *NOB* = 10 obstacles, *NCP* = 100 control points, and 1100 nonlinear constraints (i.e., (*NCP* + *NBP*) × *NOB* = (100 + 10) × 10 = 1100).

Besides the 20 independent optimization runs, 1 additional optimization run was carried out by setting the initial positions of base points equal to the coordinates of obstacle

centers shifted by 0.001 mm in order to avoid constraint singularity. This is the worst-case scenario when the trajectory intersects all obstacles. The corresponding path lengths were 231.2 mm for 7 base points and 238.5 mm for 10 base points, while initial constraint violation raised to 159,816% and 204,741%, respectively.

For both problem variants, the FHGWJA hybrid optimizer developed here was compared with the best optimizers quoted in the literature: namely, hybrid harmony search (hybrid HS), hybrid big bang–big crunch (hybrid BBBC), and hybrid fast simulated annealing (HFSA) [81]. Such a comparison should be considered highly indicative because the hybrid optimizers developed in [81] combined efficient metaheuristic search engines with approximate line search strategies and gradient information while FHGWJA repeatedly utilizes elitist strategies based on the concept of descent direction. The classical grey wolf optimizer (GWO) and the classical and improved JAYA formulations [81] also were compared with FHGWJA to prove the advantages of the hybrid optimizer. Finally, modified harmony search optimization with adaptive parameter updating (mAHS) (derived from the AHS algorithm of Ref. [94]), modified big bang–big crunch with upper bound strategy (mBBBC–UBS) (derived from the BBBC–UBS algorithm of Ref. [95]), and modified sinusoidal differential evolution (MsinDE) (derived from the sinDE algorithm of Ref. [96]) were enhanced with the $W(\vec{X}_{i,tr}) \leq 1.1W(\vec{X}_{opt})$ elitist strategy of FHGWJA to gather specific information on the efficiency of this strategy.

The selected population size in Ref. [81] for the hybrid HS, hybrid BBBC, and improved JAYA algorithms was 20, while it was set equal to 10 in this study for the new optimization runs carried out for mAHS, mBBBC–UBS, and MsinDE to have a homogeneous basis of comparison with FHGWJA and its component algorithms, the standard GWO and standard JAYA. All other internal parameters required for SHGWJA's competitors were left the same as those indicated in the original references [81,94–96]. The same approach was adopted for the two problem variants with 14 and 20 optimization variables.

Table 1 presents the optimization results for the 2D path planning problem variant with 14 design variables (seven base points and seven obstacles). All data listed in the table refer to feasible solutions; the number of analyses refers to the number of times that nonlinear constraints were evaluated. FHGWJA was the best optimizer overall and designed the shortest path of 40.811 mm, outperforming the component algorithms, the standard GWO and standard JAYA, that designed longer trajectories of 41.104 mm and 41.050 mm, respectively. The optimal positions of base points found with FHGWJA were (3.6617;1.3944), (6.5927;2.7934), (16.562;11.705), (21.231;17.081), (24.186;19.826), (27.855;26.254), and (28.626;28.196) mm. The other algorithms ranked as follows: hybrid HS [81], MsinDE (derived from [96]), hybrid BBBC [81], HFSA [81], mAHS (derived from [94]), mBBBC–UBS (derived from [95]), and improved JAYA [81].

Table 1. Optimization results for the 2D path planning problem solved with 14 design variables.

Algorithm	Best (mm)	Average (mm)	Worst (mm)	ST Dev (mm)	Number of Analyses
FHGWJA (present)	40.811	40.859	40.890	3.331×10^{-2}	4581 ± 1127 ♣
Standard GWO	41.104	41.112	41.129	1.340×10^{-2}	50,000
Standard JAYA	41.050	41.092	41.162	6.080×10^{-2}	17,204
Improved JAYA [81]	41.082	41.104	41.119	1.850×10^{-2}	6523
Hybrid HS [81]	40.983	41.040	41.105	6.133×10^{-2}	4150
Hybrid BBBC [81]	40.986	41.078	41.101	6.110×10^{-2}	3611
HFSA [81]	40.994	41.048	41.113	6.035×10^{-2}	4127
mAHS (der. from [94])	41.059	41.067	41.095	1.890×10^{-2}	7237
mBBBC–UBS (der. from [95])	41.081	41.099	41.112	1.136×10^{-2}	5724
MsinDE (der. from [96])	40.985	41.002	41.026	2.122×10^{-2}	6291

♣ The fastest optimization run of FHGWJA requires only 3837 analyses.

The rate of success of FHGWJA was 100% because all independent optimization runs designed shorter trajectories than the best solutions obtained with its competitors. In particular, the worst solution of FHGWJA achieved a path length of 40.890 mm while the shortest paths designed via all other optimizers ranged between 40.983 and 41.104 mm. Such a behavior occurred because FHGWJA inherently balances exploration and exploitation phases enhancing the exploration capability of α, β, and δ wolves without risk of stagnation and forcing the optimizer to search very high-quality solutions regardless of performing exploration or exploitation. Interestingly, the $W(\vec{X}_{i,tr}) \leq 1.1W(\vec{X}_{best})$ elitist strategy implemented in FHGWJA was effective also for mAHS, mBBBC–UBS, and MsinDE that improved their optimized solutions with respect to the original AHS [94], BBBC–UBS [95], and sinDE [96] algorithms utilized in [81]: in particular, MsinDE enhanced its overall rank up to the 3rd position.

FHGWJA was the fastest optimizer overall in the 2D path planning problem with 14 variables, requiring on average only 4581 analyses vs. 50,000 and 17,204 analyses, respectively, required for standard GWO and JAYA to converge to their optimized solutions. Furthermore, the present algorithm was about 42.3% faster than the improved JAYA. The computational speed for the fastest optimization run of FHGWJA was practically the same as those of the HS/BBBC/SA hybrid algorithms: only 3837 analyses vs. 4150, 3611, and 4127 analyses, respectively. However, the present algorithm generated a feasible intermediate design with a path length of 40.961 mm (hence, shorter than optimized path lengths of hybrid HS/BBBC/SA ranging between 40.983 and 40.994 mm) within only 2875 analyses.

FHGWJA ranked 6th in terms of standard deviation on optimized path length after mBBBC–UBS, standard GWO, improved JAYA, mAHS, and MsinDE, which, however, completed their optimizations within more analyses than the present algorithm and converged to worse designs. The present algorithm was also robust enough with respect to computational cost showing only 24.6% dispersion on the number of analyses required in the 20 independent optimization runs.

Figure 5 confirms the superiority of FHGWJA over its competitors in terms of convergence behavior. The figure compares the optimization histories of the best runs for the algorithms listed in Table 1; the average FHGWJA convergence curve is also shown in the figure. The plot is limited to the first 4500 analyses of the optimization history and to the 40.5–48.5 mm trajectory length interval for the sake of clarity. It can be seen from the figure that the best and average optimization runs' convergence curves of FHGWJA always lie below those of the other algorithms. FHGWJA started its best optimization run from the very large cost of 225.35 mm (i.e., about 5.5 times the globally optimum length of 40.811 mm found with FHGWJA) while the initial cost for all other optimizers ranged between 59.62 and 184.214 mm. However, the present algorithm immediately recovered the initial gap in cost function with respect to its competitors. The hybrid simulated annealing algorithm (HFSA) of Ref. [81] and the modified big bang–big crunch algorithm (mBBBC–UBS) implementing the $W(\vec{X}_{i,tr}) \leq 1.1W(\vec{X}_{opt})$ elitist strategy of FHGWJA were the only optimizers to compete in convergence speed with FHGWJA, respectively, for the first 150 and 190 analyses.

Table 2 presents the optimization results for the 2D path planning problem variant solved with 20 design variables (10 base points and 10 obstacles). FHGWJA again was the best optimizer overall. In fact, it designed the very short trajectory of length 40.898 mm while the best solutions found with the other algorithms correspond to longer trajectories than 40.898 mm, ranging between 40.997 mm (for hybrid HS) and 41.141 mm (for improved JAYA). Furthermore, the worst solution of FHGWJA corresponds to the path length of 40.924 mm, again shorter than the best designs obtained with all its competitors. The optimal positions of base points found with FHGWJA were (3.9165;1.7729), (7.9192;3.6484), (20.853;16.019), (25.143;21.071), (27.178;24.426), (27.185;24.444), (27.188;24.449), (27.967;26.268), (27.998;26.345), and (28.981;28.578) mm.

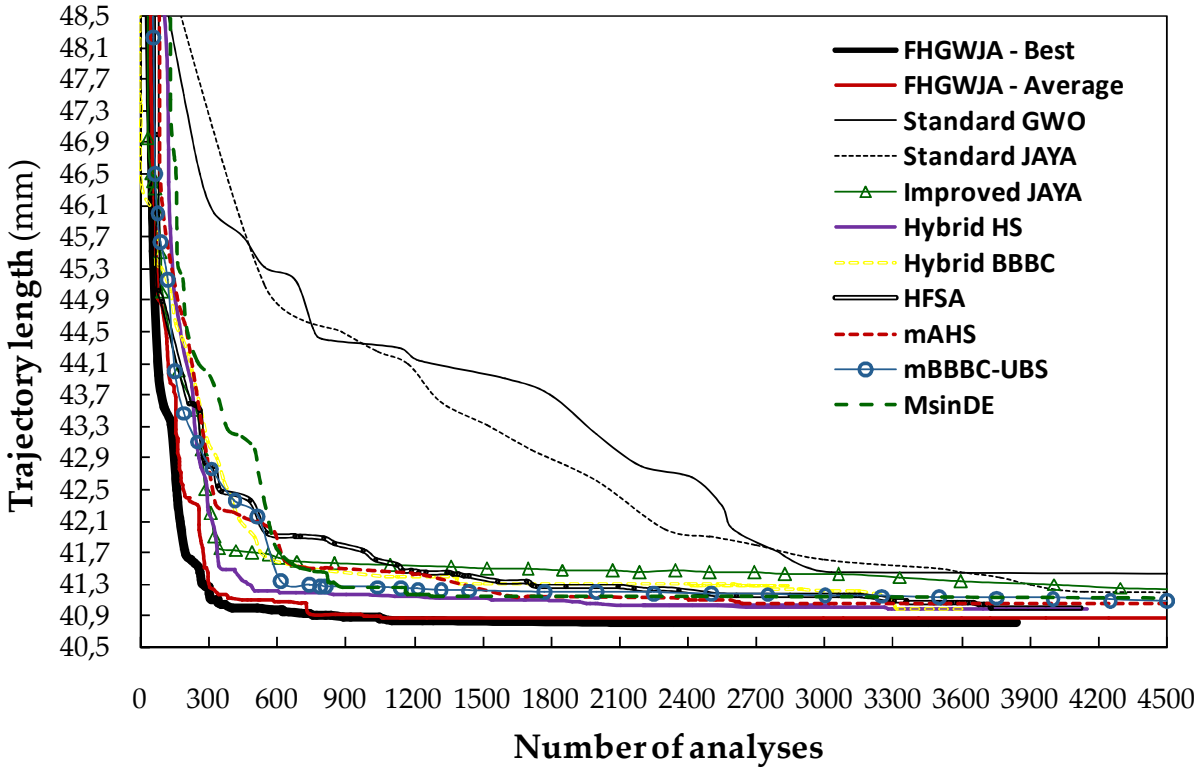

Figure 5. Comparison of convergence curves of FHGWJA and its competitors for the 2D path planning problem solved with 14 design variables. In the vertical axis, the "," notation indicates the decimal signs.

Table 2. Optimization results for the 2D path planning problem solved with 20 design variables.

Algorithm	Best (mm)	Average (mm)	Worst (mm)	ST Dev (mm)	Number of Analyses
FHGWJA (present)	40.898	40.901	40.924	1.161×10^{-2}	5153 ± 248 ♣
Standard GWO	41.134	41.146	41.165	1.276×10^{-2}	50,000
Standard JAYA	41.112	41.160	41.187	3.102×10^{-2}	11,778
Improved JAYA [81]	41.141	41.184	41.209	2.808×10^{-2}	11,393
Hybrid HS [81]	40.997	41.053	41.116	4.861×10^{-2}	7408
Hybrid BBBC [81]	41.034	41.091	41.108	3.165×10^{-2}	7542
HFSA [81]	41.004	41.060	41.127	5.028×10^{-3}	8245
mAHS (der. from [94])	41.075	41.092	41.116	1.682×10^{-2}	8541
mBBBC–UBS (der. from [95])	41.091	41.123	41.150	2.412×10^{-2}	7492
MsinDE (der. from [96])	41.030	41.077	41.092	2.641×10^{-2}	8762

♣ The fastest optimization run of FHGWJA requires only 5065 function evaluations.

FHGWJA was also significantly faster than the other algorithms, completing optimization runs on average within only 5153 analyses while computational cost of its competitors ranged between 7408 (for hybrid HS) and 50,000 (for standard GWO) analyses. Standard JAYA was slightly more efficient than improved JAYA while standard GWO was the worst optimizer overall. The present algorithm was the most robust optimizer, achieving the lowest standard deviation on path length and just 4.8% dispersion on the number of analyses.

The $W(\vec{X}_{i,tr}) \leq 1.1W(\vec{X}_{best})$ elitist strategy implemented in FHGWJA again improved significantly the performance of the mAHS, mBBBC–UBS, and MsinDE algorithms with respect to the original AHS, BBBC–UBS, and sinDE formulations documented in [94–96]:

in particular, computational cost decreased on average by 36%. Interestingly, the high level of nonlinearity of 2D path planning problem variants including 14 and 20 optimization variables was better handled with FHGWJA than with the hybrid HS/BBBC/SA algorithms of Ref. [81]. This occurred in spite of the fact that hybrid HS/BBBC/SA utilized gradient information and multiple line searches to minimize the number of analyses. However, this limited the approximation quality and hence the ability of the optimizers to generate trial solutions effectively located in regions of the search space where $\vec{X}_{best}$ is likely to improve.

Figure 6 compares the optimization histories of the best runs for FHGWJA and its competitors in the 2D path planning problem solved with 20 design variables; the average FHGWJA convergence curve is also shown in the figure. The plot is limited to the first 9000 analyses of optimization history for the sake of clarity. Similar to the 14-variable problem variant, the best and average optimization runs' convergence curves of FHGWJA practically always lie below those of the other algorithms. All algorithms started their best optimization runs from the very large cost of 238.54 mm (i.e., about 5.83 times the optimum trajectory length of 40.898 mm found with FHGWJA). However, the present algorithm recovered the gap in cost function with respect to target optimum in a much faster way than the other algorithms. The hybrid simulated annealing algorithm (HFSA) of Ref. [81] was the only optimizer to compete in convergence speed with FHGWJA but only after 2000 analyses, that is when the distance from target optimum was less than 5%.

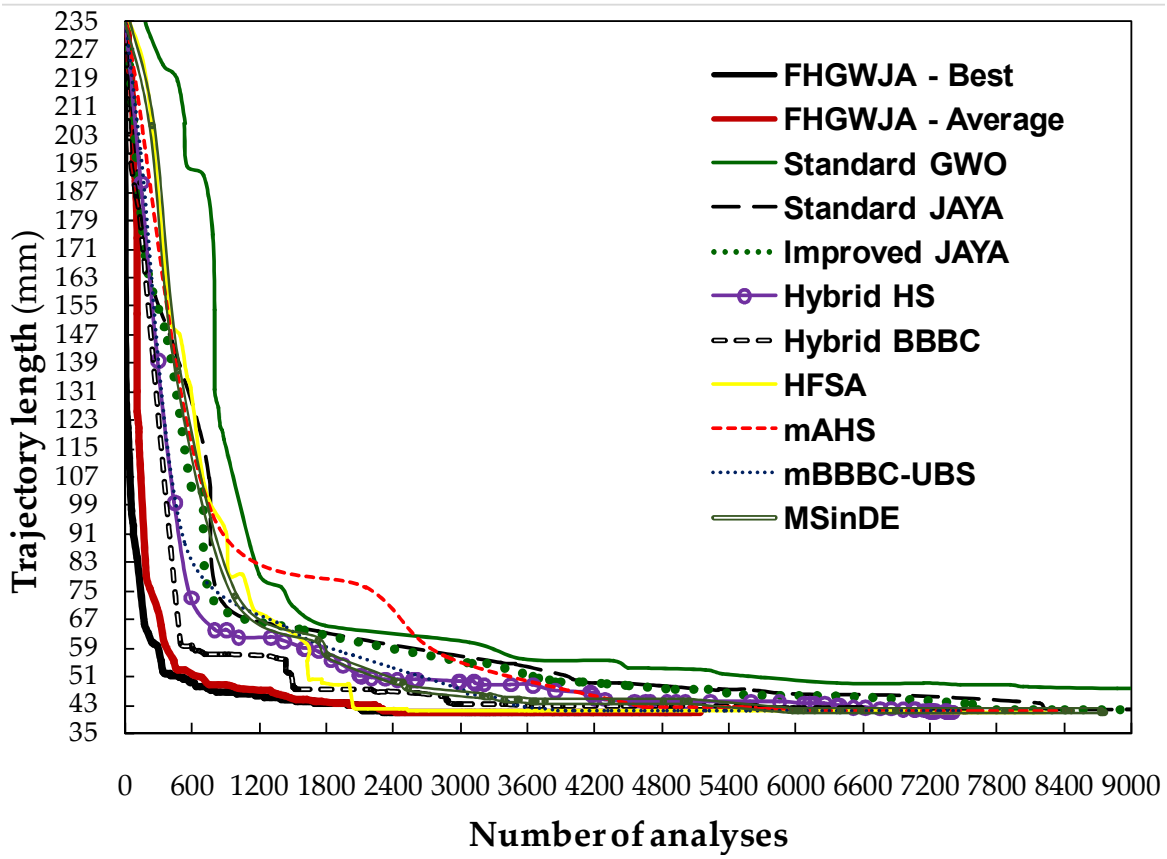

Figure 6. Comparison of convergence curves of FHGWJA and its competitors for the 2D path planning problem solved with 20 design variables.

Figure 7a compares the optimized trajectories obtained with FHGWJA and its competitors in the 2D path planning problem variant including 14 design variables. For the sake of clarity, the figure shows only the best six solutions quoted in Table 1. It can be seen that optimized paths involve only obstacles 1 and 7. Furthermore, the trajectory branch limited by the start point A and obstacle 1 is practically the same for all optimizers. Optimal

trajectories diverge between obstacles 1 and 7: the slope is larger for hybrid HS, HFSA, and standard JAYA than for the present algorithm, hybrid BBBC, and MsinDE. Hybrid HS, hybrid BBBC, HFSA, and MsinDE found very similar path lengths between 40.983 and 40.994 mm; these trajectories are practically symmetric about the line connecting the point near obstacle 1, where trajectories diverge, and the end point B. Since the trajectory designed with standard JAYA between obstacles 1 and 7 is more curved than those designed with hybrid HS and HFSA, the length of standard JAYA's optimal path increased to 41.050 mm.

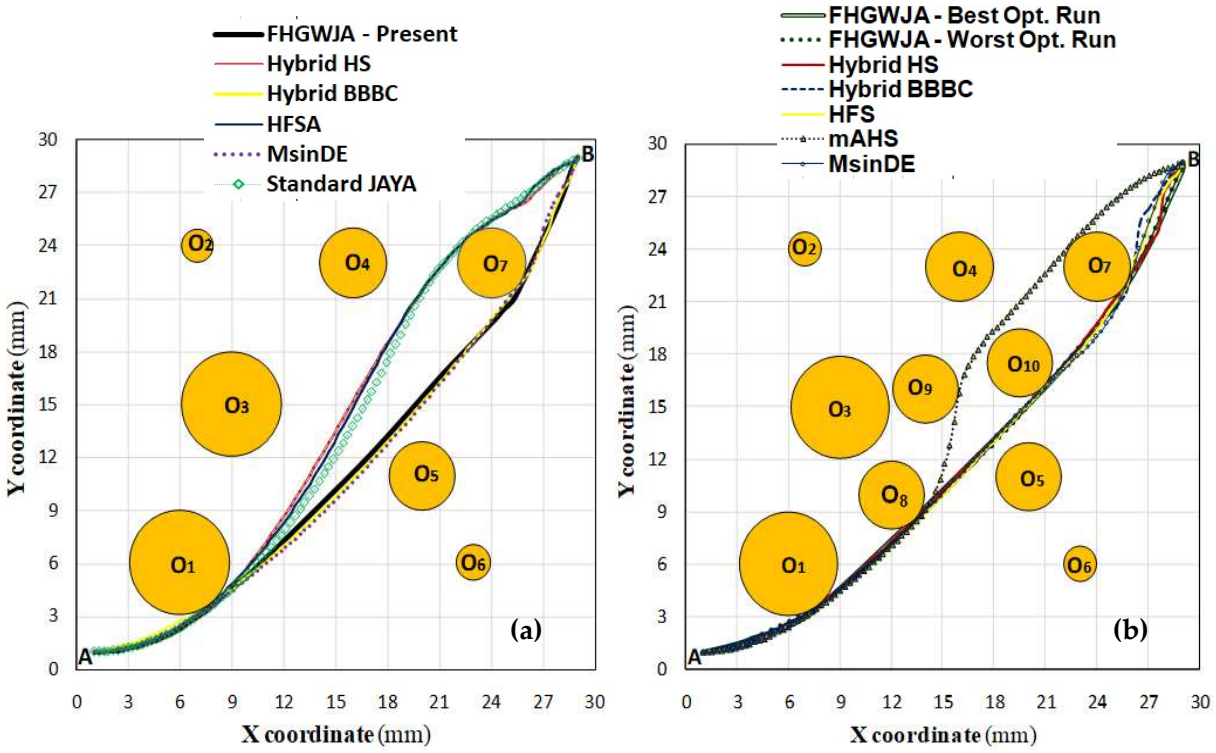

Figure 7. Comparison of optimized trajectories found with FHGWJA and its competitors in the 2D path planning problem: (**a**) 7 base points and 14 variables; (**b**) 10 base points and 20 variables.

The detailed analysis of the optimized trajectory revealed that, in order to minimize length, trajectory should be rectilinear between obstacles 1 and 7 as well as in the final branch from obstacle 7 to end point B. Interestingly, FHGWJA was the only optimizer to satisfy this requirement. In fact, the optimal path found with FHGWJA between obstacles 1 and 7 may be fitted in the XY plane via a linear regression with $R^2 = 0.999$ vs. only $R^2 = 0.992$ and $R^2 = 0.994$, respectively, for hybrid HS (HFSA's trajectory is very close to the one designed with hybrid HS) and hybrid BBBC (MsinDE's trajectory is very close to the one designed with hybrid BBBC). Furthermore, the optimal path of FHGWJA may be linearly fitted with $R^2 = 0.998$ vs. only $R^2 = 0.991$ and $R^2 = 0.994$ for hybrid HS and hybrid BBBC, respectively. Besides the higher curvature of the trajectories designed with the other algorithms between obstacles 1 and 7, the final branches of the optimal trajectories designed via hybrid HS, HFSA, hybrid BBBC, and MsinDE show changes in concavity that make trajectory deviate from a straight line.

Figure 7b compares the optimized trajectories obtained with FHGWJA and its competitors in the 2D path planning problem variant including 20 optimization variables. The figure shows the trajectories of the best and worst optimization runs of FHGWJA as well as the other five best solutions quoted in Table 2. The new obstacles 8 and 10 introduced in the problem became active for all algorithms while the optimized trajectory approached the new obstacle 9 only for mAHS. This explains why the optimized path lengths of hybrid HS, HFSA, MsinDE, and hybrid BBBC varied at most by 0.037 mm (i.e., between 40.997 mm

and 41.034 mm) while path length increased by 0.041 mm just passing from hybrid BBBC to mAHS (i.e., from 41.034 mm to 41.075 mm). Again, FHGWJA designed the straightest trajectory in the path branches between obstacles 1 and 7 and between obstacle 7 and end point B. Such a behavior occurred for both the best and worst FHGWJA optimization runs. Conversely, trajectories designed with hybrid HS/BBBC/SA and MsinDE algorithms exhibited significant changes in slope and curvature especially in the final branch of the path or between obstacles 10 and 7. For example, the final branch of the FHGWJA's optimal paths may be linearly fitted with $R^2 = 0.997$ vs. only $R^2 = 0.965$, $R^2 = 0.940$, and $R^2 = 0.978$ for hybrid HS, hybrid BBBC, and HFSA, respectively.

In summary, the FHGWJA algorithm developed in this study is an efficient and robust tool for solving highly nonlinear 2D path planning problems in robotics. The proposed algorithm could always design shorter paths requiring less analyses to complete optimization runs than its competitors.

3.2. Shape Optimization of a Concrete Gravity Dam Under Multiple Earthquake Loads

The test case (iii) solved in this study was the shape optimization of a concrete gravity dam under multiple earthquake loads. The real structure is located in the Shafaroud region of Northern Iran: the dam height (H_{dam}) is 150 m, water elevation in the normal state in the dam upstream (h_{water}) is 145 m, and the sediment height (h_{se}) is 7 m. The 3D schematic of the dam is shown in Figure 8a. Here, the concrete volume per unit width of the dam was minimized by optimizing the dam cross-section shown in Figure 8b with the nine shape variables (x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9) indicated in the figure.

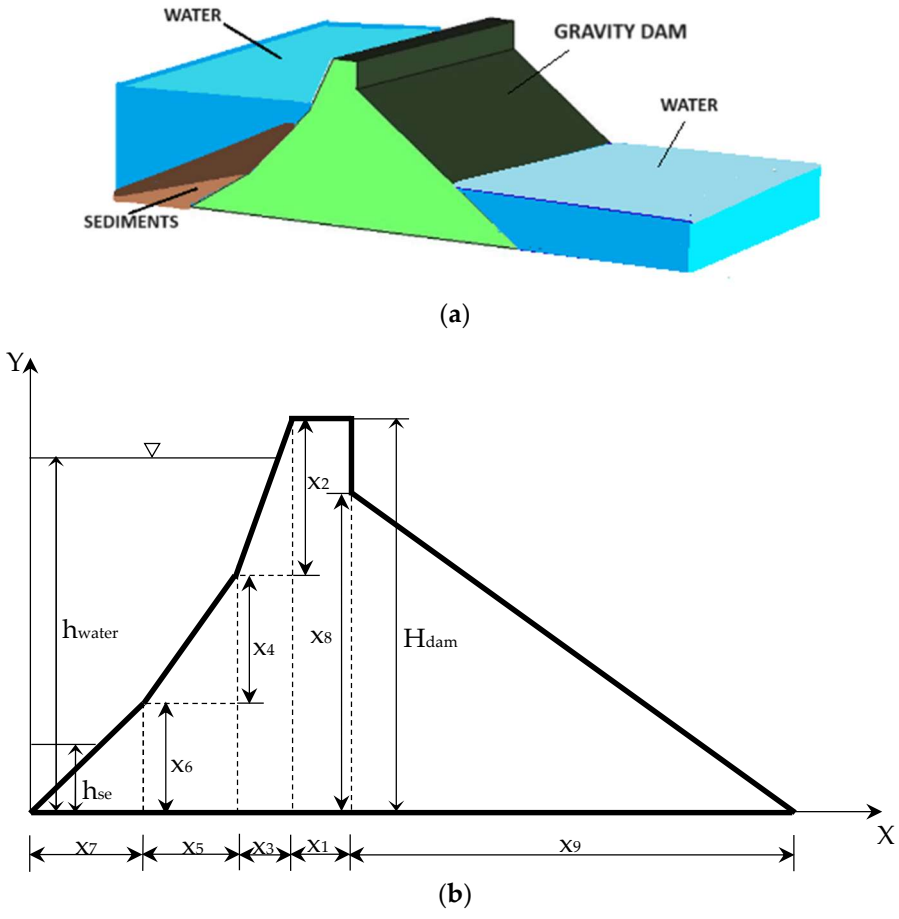

Figure 8. (**a**) 3D schematic of the gravity dam; (**b**) cross-section of the concrete gravity dam indicating shape variables and main geometric dimensions.

The optimization problem was stated as follows:

$$Minimize\ W(\vec{X}) = H_{dam}x_1 + \frac{1}{2}x_2x_3 + x_3(x_4 + x_6) + \frac{1}{2}x_4x_5 + x_5x_6 + \frac{1}{2}x_6x_7 + \frac{1}{2}x_8x_9 \quad (20)$$

$$Subject\ to \begin{cases} G_1(\vec{X}) = 4 - SFS \le 0 \\ G_2(\vec{X}) = 1.5 - SFO \le 0 \\ G_{3-4}(\vec{X}) = 0 \le \sigma_U \le \sigma_{MAX} \\ G_{5-6}(\vec{X}) = 0 \le \sigma_D \le \sigma_{MAX} \\ G_7(\vec{X}) = x_4 + x_6 - x_8 \le 0 \\ H_8(\vec{X}) = (x_2 + x_4 + x_6 - H_{dam})^2 = 0 \end{cases} \quad (21)$$

where SFS is the safety factor against dam sliding; SFO is the safety factor against dam overturning; σ_U, σ_D, and σ_{max} are the vertical fatigue specific loads in upstream and downstream and the corresponding fatigue limit, respectively. Side constraints of shape variables are as follows: $1 \le x_1, x_3 \le 20$ m; $1 \le x_2 \le 50$ m; $5 \le x_4, x_5, x_6, x_7 \le 100$ m; $50 \le x_8, x_9 \le 150$ m.

In Equation (21), the SFS factor (involved in the inequality constraint G_1) is defined as $(f \cdot \Sigma F_v + \sigma \cdot b)/\Sigma F_H$, where f = 0.7 is the static friction coefficient, σ = 3.5 MPa is the allowable shear tension in the cutting surface, and $b = (x_1 + x_3 + x_5 + x_7 + x_9)$ is the dam base length. The dam is subject to the vertical forces denoted as F_v: concrete weight W, vertical components of water pressure Fh_v, and sediment pressure P_{SV} directed downward; uplift force generated by the dam basement F_{uplift}, and earthquake force F_E directed upward. The dam is also subject to horizontal forces F_H in the positive X-direction: an additional earthquake force of magnitude $2F_E$, the horizontal components of water pressure Fh_h, and sediment pressure P_{SO}.

The SFO factor (involved in the inequality constraint G_2) is defined as $\Sigma M_R / \Sigma M_o$ where M_R and M_o, respectively, are the torque of resisting forces (W, Fh_v, P_{SV}) and the torque of driving forces (F_{uplift}, $2F_E$, Fh_h, P_{SO}) on the dam.

Vertical fatigue loads σ_U and σ_D (involved in inequality constraints G_5 and G_6) are defined as $(\Sigma F_v/b - 6\Sigma M_o/b^2)$ and $(\Sigma F_v/b + 6\Sigma M_o/b^2)$, respectively.

The last inequality constraint G_7 in Equation (21) is relative to geometric proportions in the vertical direction, while the equality constraint H_8 requires the dam height to be 150 m.

The concrete volume per unit width of 10,502.1 m^3 of the real dam was significantly reduced to 7448.774 m^3 via the multi-level cross entropy optimizer (MCEO) [97] and the flying squirrel optimizer (FSO) [98]. However, Refs. [97,98] neither considered the equality constraint H_8 nor the presence of the horizontal earthquake force $2F_E$. These issues were instead included in [81], which presented the results of hybrid HS, hybrid BBBC, hybrid fast simulated annealing (HFSA), adaptive harmony search (AHS) [94], big bang–big crunch with upper bound strategy (BBBC–UBS) [95], and sinusoidal differential evolution [96]. For this reason, the FHGWJA algorithm developed in this study was compared with hybrid HS/BBBC/SA as well as with modified adaptive harmony search (mAHS), modified big bang–big crunch with upper bound strategy (mBBBC–UBS), and modified sinusoidal differential evolution (MsinDE). Similar to the 2D path planning problem, mAHS, mBBBC–UBS, and MsinDE included the $W(\vec{X}_{i,tr}) \le 1.1W(\vec{X}_{best})$ elitist strategy implemented in FHGWJA to enhance the performance of the original algorithms.

Similar to the 2D path planning problem, population size was set equal to 20 in Ref. [81] for hybrid HS, hybrid BBBC, and improved JAYA, while mAHS, mBBBC–UBS, and MsinDE were run with N_{POP} = 10 in this study.

Table 3 compares the optimization results of FHGWJA and its competitors for the dam problem. The number of structural analyses corresponds to the number of evaluations of constraints $G_{1-7}(\vec{X})$ and $H_8(\vec{X})$. The average number of structural analyses and corre-

sponding standard deviation along with the number of structural analyses required by the fastest optimization run also are reported in the table when available.

Table 3. Optimization results of the concrete gravity dam design problem.

Algorithm	Best (m^3)	Average (m^3)	Worst (m^3)	ST Dev (m^3)	Number Structural Analyses	Constraint Violation (%)
FHGWJA (Present)	7079.558	7079.558	7079.558	0	3834 ± 1597 2207	1.441×10^{-3}
Standard GWO	7117.441	7119.288	7125.568	3.4995	50,000	2.423×10^{-3}
Standard JAYA	7121.134	7125.213	7129.822	5.3946	25,000	1.338×10^{-3}
Improved JAYA [81]	7117.437	7119.141	7120.890	1.7269	$22{,}537 \pm 1996$ 19,917	4.076×10^{-4}
Hybrid HS [81]	7109.608	7112.340	7113.474	2.5882	$18{,}487 \pm 3226$ 17,708	3.641×10^{-5}
Hybrid BBBC [81]	7108.772	7111.175	7112.275	1.8606	$18{,}074 \pm 2692$ 18,128	3.029×10^{-5}
HFSA [81]	7113.456	7114.225	7114.533	1.5385	$19{,}390 \pm 1324$ 18,232	1.343×10^{-3}
mAHS (derived from [94])	7109.419	7111.476	7113.292	2.0719	$24{,}687 \pm 2916$ 22,172	3.333×10^{-3}
mBBBC–UBS (derived from [95])	7109.122	7110.306	7111.004	0.7912	$18{,}376 \pm 1737$ 16,924	6.067×10^{-3}
MsinDE (dererived from [96])	7113.123	7114.350	7115.267	0.8983	$24{,}196 \pm 2048$ 22,809	4.145×10^{-3}
MCEO [97]	7448.74	N/A	N/A	3.274×10^{-2}	35,000	32
FSO [98]	7448.74	N/A	N/A	N/A	N/A	32

FHGWJA was the best optimizer overall also in this test problem. In fact, it converged to the lowest concrete volume for unit width of 7079.558 m^3, which is 0.411% smaller than the 7108.772 m^3 for the best design quoted in literature so far for hybrid BBBC [81]. All algorithms performed significantly better than MCEO [97] and FSO [98]: in fact, optimized values of unit volume ranged from 7059.558 m^3 to 7121.134 m^3 vs. 7448.774 m^3 of Refs. [97,98]. The solutions quoted in Table 3 fully satisfied design constraints $G_1(\vec{X})$ through $G_7(\vec{X})$ and practically the geometric equality constraint $H_8(\vec{X})$ on dam height (in fact, violation never exceeded 6.067×10^{-3}%). The algorithms ranked as follows in terms of optimized concrete volume values: FHGWJA, hybrid BBBC, mBBBC–UBS, mAHS, hybrid HS, MsinDE, HFSA, improved JAYA, standard GWO, and standard JAYA.

FHGWJA was also the fastest and most robust optimizer. In fact, it converged to the global optimum of 7059.558 m^3 in all independent optimization runs with zero standard deviation. Conversely, none of its competitors could achieve zero standard deviation on optimized concrete volume values. The present algorithm was on average between 4.7- and 6.5-times faster than its competitors; the fastest optimization run converging to the global optimum was completed by FHGWJA within only 2207 structural analyses vs. 16,924 structural analyses required for mBBBC–UBS that, however, obtained a higher value for the dam's concrete volume.

The hybrid formulation implemented in FHGWJA was significantly more efficient than its component algorithms GWO and JAYA that converged in their standard formulation to the best solutions of 7117.441 m^3 and 7121.134 m^3, respectively. The improved JAYA algorithm of Ref. [81], implementing a simplified elitist strategy where each new trial solution $\vec{X}_{i,tr}$ is included in the new population only if it improves its counterpart individual $\vec{X}_i$ (i.e., if it holds $W(\vec{X}_{i,tr}) \leq W(\vec{X}_i)$), also could not reduce the unit volume of the dam below 7117.44 m^3. The classical GWO was on average about 13-times more computationally

expensive than the present algorithm, while classical JAYA and improved JAYA were on average about 6.5- and 5.9-times slower than FHGWJA.

The $W(\vec{X}_{i,tr}) \leq 1.1W(\vec{X}_{best})$ elitist strategy implemented in FHGWJA again improved significantly the performance of the mAHS, mBBBC–UBS, and MsinDE algorithms with respect to the original AHS, BBBC–UBS, and sinDE formulations documented in [94–96]: in particular, the best solutions of mAHS, mBBBC–UBS, and MsinDE became very close to those of hybrid HS/BBBC/SA while computational cost decreased on average by 16.5%, 6.1%, and 35.2%, respectively.

Figure 9 clearly shows the superior convergence behavior of FHGWJA with respect to its competitors. The figure compares the optimization histories of the best runs for the algorithms listed in Table 3; the average FHGWJA convergence curve also is shown in the figure. The plot is limited to the first 21,000 structural analyses of the optimization history and to the 7000–14,000 m^3 concrete volume interval for the sake of clarity. It can be seen from the figure that the best and average optimization runs' convergence curves of FHGWJA always lie below those of the other algorithms. FHGWJA started its best optimization run from the very large cost of 18,372.761 m^3 (i.e., about 2.6 times the globally optimum length of 7079.558 m^3 found with FHGWJA) while the initial cost for all other optimizers ranged between 13,265.001 and 14,209.219 m^3. However, the present algorithm immediately recovered the initial gap in cost function with respect to its competitors. The hybrid simulated annealing algorithm (HFSA) of Ref. [81] was the only optimizer to compete in convergence speed with FHGWJA for the first 100 structural analyses.

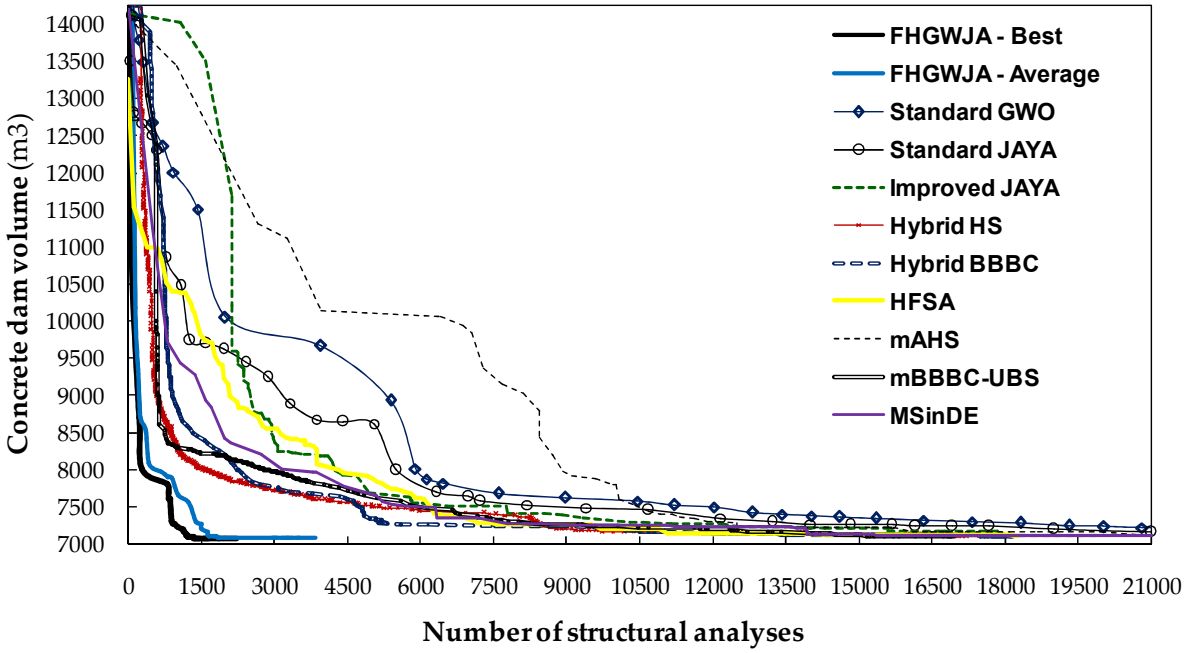

Figure 9. Comparison of convergence curves of FHGWJA and its competitors for the concrete dam optimization problem.

Table 4 lists the optimized designs obtained with FHGWJA and its main competitors; coordinate values are expressed in meters. The data relative to the standard GWO and JAYA and the improved JAYA are omitted for the sake of brevity as these algorithms converged to the worst solutions overall. The 7079.558 m^3 dam's concrete volume design obtained with FHGWJA is very similar to the optimal solutions obtained with the other algorithms: all design variables changed by at most 5.6% except X_3 that was fixed to its lower bound of 1 mm by FHGWJA and between 4.0001 m and 4.1358 m for the other algorithms.

Table 4. Comparison of optimized designs obtained with FHGWJA and its competitors in the concrete gravity dam problem.

Variables (m)	FHGWJA	Hybrid HS [81]	Hybrid BBBC [81]	HFSA [81]	mAHS (der. [94])	mBBC–UBS (der. [95])	MsinDE (der. [96])	MCEO/FSO [97,98]	Existing Dam
X_1	5.0000	5.0000	5.0006	5.0004	5.0394	5.0175	5.0256	5.1678	4
X_2	50.000	49.670	49.554	49.924	49.211	49.333	49.497	25.1978	42
X_3	1.0000	4.0132	4.0043	4.0001	4.1358	4.0846	4.0866	4.9556	4.2
X_4	69.784	70.084	70.309	68.244	70.006	70.053	69.483	55.7811	50
X_5	24.835	24.499	24.539	23.955	23.973	24.175	24.066	16.0874	12.5
X_6	30.000	30.239	30.131	31.673	30.696	30.924	31.090	21.0009	58
X_7	40.000	39.997	40.025	39.984	39.974	39.968	39.919	29.9036	23.2
X_8	99.770	100.39	100.49	99.241	100.51	100.75	100.26	120.169	140
X_9	80.000	80.261	80.164	81.874	80.566	81.348	80.947	85.382	105

Figure 10 compares the optimized shapes of the dam (coordinates are expressed in meters) obtained with FHGWJA and its competitors with the real structure and the optimum design of Refs. [97,98]. For the sake of clarity, the average of hybrid HS, hybrid BBBC, and HFSA configurations and the average of mAHS, mBBBC–UBS, and MsinDE configurations are shown in the figure. It can be seen that all designs resemble the aspect ratio of the real dam: in particular, the ratio between optimized base length and height is very close to 1. The optimized profiles are slightly shifted in the positive X-direction with respect to the real dam and the dam configurations optimized in [97,98] to counteract the effect of the additional horizontal earthquake force $2F_E$ included in the design problem. The smaller inclination of the bottom segment of the dam's upstream side increases the dam's storage capacity with respect to the real structure.

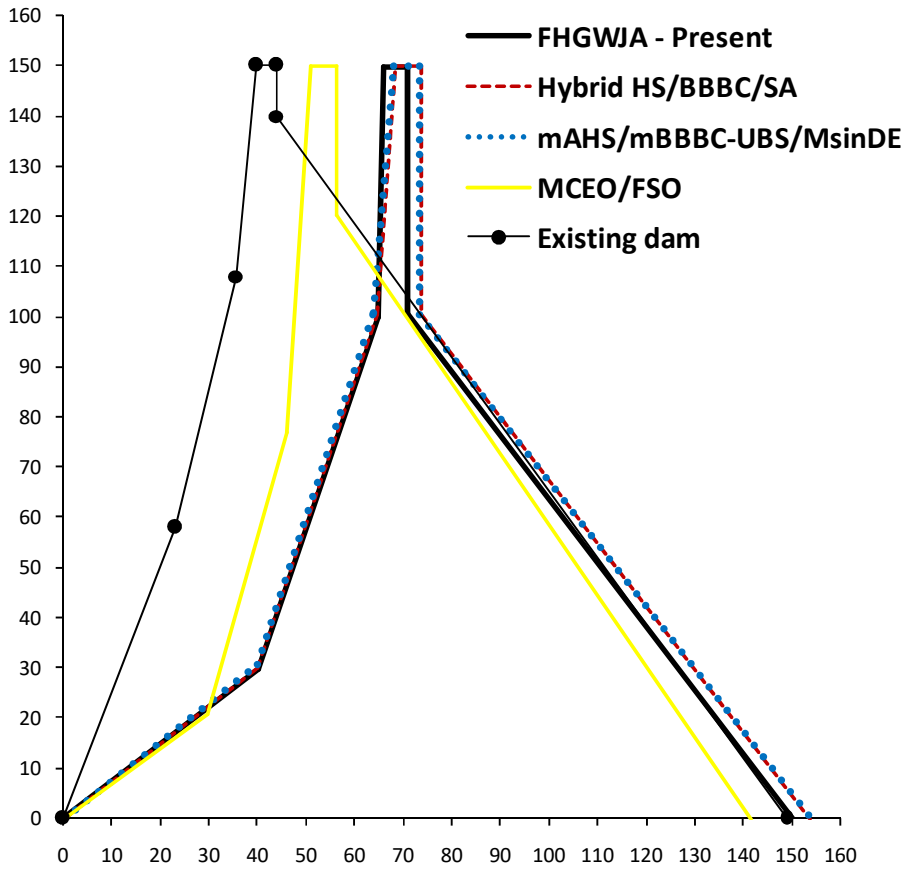

Figure 10. Comparison of concrete dam optimized shapes obtained with FHGWJA and its competitors.

The safety factors against sliding and overturning evaluated for the optimal solution of FHGWJA became 5.738 and 3.470, respectively, practically the same as those quoted in [81]. Hence, reducing the concrete volume did not affect at all the level of structural safety of the dam. This proves the suitability of the proposed FHGWJA algorithm for civil engineering design problems.

3.3. Calibration of Nonlinear Hydrologic Models

Test problems (iv–v) regarded the calibration of nonlinear hydrologic models including 3 or 25 unknown parameters. This is a typical inverse problem in the hydrology field [99–101]. The Muskingum model was selected in this study. The difference between observed river outflows and their counterpart computed with the Muskingum model over a period of 126 h, split in 21 intervals Δt = 6 h, was minimized. The inverse problem was formulated as an unconstrained optimization problem:

$$Minimize\ SSQ = \sum_{t=1}^{NCT} (O_{oro,t} - O_{cro,t})^2 \tag{22}$$

where $O_{oro,t}$ and $O_{cro,t}$, respectively, are the observed and computed river outflows at time t; NCT = 22 is the number of control points defined by the 21 time intervals. The computed river outflow $O_{cro,t}$ can be determined using the classical three-parameter nonlinear Muskingum model:

$$O_{cro,t} = \frac{1}{1-h}\left(\frac{S_t}{K}\right)^{\frac{1}{r}}(1-h)\ I_t \tag{23}$$

In Equation (23), K, h, and r are the parameters of Muskingum model to be calibrated, respectively, taken as optimization variables x_1, x_2, and x_3; S_t and I_t, respectively, are the channel storage and measured river inlet at time t. The channel storage S_t at time t is updated as follows:

$$S_{t+1} = S_t + \dot{S}_t \Delta t \tag{24}$$

The storage rate $\dot{S}_t$ at time t is as follows:

$$\dot{S}_t = \frac{1}{1-h} \cdot \left(\frac{S_t}{K}\right)^{\frac{1}{r}} + \frac{1}{1-h} \cdot I_t \tag{25}$$

Two variants of the hydrologic model calibration problem were solved in this study: (i) the classical problem including only 3 optimization variables, the unknown parameters K, h, and r [102–106]; (ii) the extended problem solved in [81] including also the 22 storage values S_t as additional design variables for a total of 25 optimization variables. The side constraints on Muskingum model parameters and storage values are as follows: $0.01 \le k \le 0.2$; $0.2 \le h \le 0.3$; $1.5 \le r \le 2.5$, and $1 \le S_t \le 1000\ m^3$.

The FHGWJA algorithm developed in this study was compared with the best solutions of the literature obtained with following: (i) metaheuristic methods such as the hybrid HS/BBBC/SA algorithms of Ref. [81], other HS variants [102], improved immune clonal selection (IICSA) [103], backtracking search with orthogonal initialization and chaotic map (COBSA) [104], genetic algorithms (GA) [104], differential evolution (DE) [104], and particle swarm optimization (PSO) [104]; (ii) modified adaptive harmony search (mAHS), modified big bang–big crunch with upper bound strategy (mBBBC–UBS), and modified sinusoidal differential evolution (MsinDE) including the FHGWJA's elitist strategy $W(\vec{X}_{i,tr}) \le 1.1W(\vec{X}_{best})$ to improve the original AHS/BBBC–UBS/sinDE algorithms of Refs. [94–96]; (iii) gradient-based methods such as the Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS) [105]; (iv) zero-order heuristic methods such as the Nelder-Mead Simplex algorithm (NMS) [106]. FHGWJA was also compared with the standard GWO as well as standard and improved JAYA variants [81].

Similar to the previous test cases, population size was set equal to 20 in Ref. [81] for hybrid HS, hybrid BBBC, and improved JAYA, while mAHS, mBBBC–UBS, and MsinDE were run with N_{POP} = 10 in the present study.

Table 5 presents the optimization results of FHGWJA and its competitors for the three-variables problem variant. The table reports the average number of evaluations of the SSQ cost function (*NFE*) and corresponding standard deviation, along with the number of function evaluations required by the fastest optimization run when available.

Table 5. Optimization results of the hydrologic model calibration problem solved with 3 unknown parameters.

Algorithm	*K*	*h*	*m*	Best	Average	Worst	ST Dev	NFE
FHGWJA (Present)	0.08620	0.2869	1.8681	36.760	36.760	36.760	0	2005 ± 386 1637
Standard GWO	0.08621	0.2869	1.8681	36.761	36.762	36.764	1.258×10^{-3}	50,000
Standard JAYA	0.08620	0.2869	1.8681	36.760	36.761	36.763	1.500×10^{-3}	5438 ± 756 4331
Improved JAYA [81]	0.08620	0.2869	1.8681	36.760	36.761	36.762	1.442×10^{-4}	2412 ± 684 1766
Hybrid HS [81]	0.08633	0.2869	1.8677	36.761	36.762	36.762	4.892×10^{-4}	2547 ± 1035 1331
Hybrid BBBC [81]	0.08618	0.2869	1.8682	36.761	36.761	36.762	1.769×10^{-5}	2468 ± 1102 1281
HFSA [81]	0.08632	0.2869	1.8678	36.761	36.762	36.762	3.891×10^{-4}	2755 ± 1344 1326
mAHS (der. [94])	0.08620	0.2869	1.8681	36.760	36.761	36.762	1.543×10^{-4}	2368 ± 967 1670
mBBBC–UBS (der. [95])	0.08620	0.2869	1.8681	36.760	36.761	36.762	2.044×10^{-4}	2344 ± 835 1711
MsinDE (der. [96])	0.08620	0.2869	1.8681	36.760	36.762	36.763	2.119×10^{-3}	2966 ± 1293 2213
HS [102]	0.0870	0.2870	1.8661	36.77	N/A	N/A	N/A	20,000
IICSA [103]	0.0865	0.2870	1.8675	36.77	N/A	N/A	N/A	N/A
COBSA [104]	0.0864	0.2869	1.8678	36.76	36.77	36.77	0.01	100,000
GA [104]	0.0865	0.2869	1.8674	36.77	36.85	37.24	0.13	N/A
DE [104]	0.0863	0.2869	1.8680	36.77	37.09	38.04	0.36	N/A
PSO [104]	0.0867	0.2867	1.8668	36.77	36.84	37.25	0.14	N/A
BFGS [105]	0.0863	0.2869	1.8679	36.768	N/A	N/A	N/A	N/A
NMS [106]	0.0862	0.2869	1.8681	36.76	N/A	N/A	N/A	221 *

* Number of optimization iterations.

It can be seen from Table 5 that the present algorithm always converged to the lowest value of SSQ = 36.760 in all independent optimization runs thus reaching 0 standard deviation on optimized cost. Hence, FHGWJA was the most robust optimizer overall. Standard and improved JAYA, modified AHS/BBBC–UBS/sinDE, COBSA, and NMS also found the globally optimum solution. However, none of these algorithms converged to the same design in all optimization runs: standard deviations on optimized SSQ values ranged between 1.442×10^{-4} (improved JAYA) and 0.01 (COBSA). All other methods found optimized solutions ranging between SQQ = 36.761 (standard GWO and hybrid HS/BBBC/SA) and 36.77 (HS, IICSA, GA, DE, and PSO).

The hybrid FHGWJA algorithm was significantly more efficient than its components, the GWO and JAYA. In particular, the standard GWO converged to SSQ = 36.761 after 50,000 function evaluations (i.e., 25-times slower than FHGWJA) while standard JAYA converged to SSQ = 36.760 but it was on average 2.7-times slower than the present algorithm. Improved JAYA was on average 20.3% slower than FHGWJA while their fastest optimization runs practically required the same number of function evaluations (i.e., 1637 for FHGWJA vs. 1766 for improved JAYA). However, the fastest optimization run of im-

proved JAYA converged to SSQ = 36.762 vs. the global optimum of 36.760 reached with the present algorithm in all runs.

The computational efficiency of FHGWJA operators generating new trial solutions in the search process is confirmed by the fact that the $W(\vec{X}_{i,tr}) \leq 1.1W(\vec{X}_{best})$ elitist strategy allowed mAHS, mBBBC–UBS, and MsinDE algorithms to converge to the global optimum value SSQ = 36.760 and reduce computational cost by 9.1%, 11%, and 35.6% with respect to the original AHS, BBBC–UBS, and sinDE algorithms of Refs. [94–96]. Interestingly, mAHS and mBBBC–UBS became very competitive with FHGWJA in terms of average computational cost (respectively, 2368 and 2344 function evaluations vs. only 2005 of FHGWJA) and peak computational speed of the fastest optimization run (respectively, 1670 and 1711 vs. only 1637 of FHGWJA).

The hybrid HS/BBBC/SA algorithms of Ref. [81] that utilized gradient information and approximate line searches could even complete their fastest optimization runs within less function evaluations than FHGWJA (i.e., between 1281 and 1331 vs. 1637 of FHGWJA) but converged to higher values of SSQ than the global optimum 37.760 found with FHGWJA. Furthermore, hybrid HS/BBBC/SA required on average more function evaluations than FHGWJA (i.e., between 2468 and 2755 vs. only 2005 of FHGWJA). Again, in a highly nonlinear problem like the hydrologic model calibration, the elitist strategies implemented in FHGWJA were extremely efficient in generating high-quality trial solutions over the whole search process. This happened because generation of trial solutions in FHGWJA is not biased to any extent by the quality of gradient information that may instead drive hybrid HS/BBBC/SA towards poor trial solutions should the cost function gradient not be well approximated. However, looking at Table 5, it appears that computational cost was not an issue for this test problem: HS [102] and COBSA [104], respectively, required 20,000 and 100,000 function evaluations vs. only 2000 of FHGWJA while no information on the effective number of function evaluations were given in Refs. [103–106] for metaheuristic (i.e., IICSA, GA, DE, and PSO) as well as gradient-based algorithms (BFGS) and zero-order heuristic methods (NMS).

Figure 11 confirms the superiority of FHGWJA over its competitors in terms of convergence behavior also in this test problem. The figure compares the optimization histories of the best runs for the algorithms listed in Table 5; the average FHGWJA convergence curve is also shown in the figure. Since FHGWJA always converged to the global optimum of 36.760 in all optimization runs, its best run corresponds to the fastest one. The plot is limited to the first 4500 function evaluations of optimization history and to the 35–70 cost function interval for the sake of clarity. It can be seen from the figure that the best and average optimization runs' convergence curves for FHGWJA lie below those of the other algorithms practically over the whole search history for this test problem. FHGWJA started its best optimization run from the very large cost of 1520.145 (i.e., about 41.3 times the target optimum of 36.760 quoted in Table 5) while the initial cost for all other optimizers ranged between 181.683 and 1122.263. However, the present algorithm immediately recovered the initial gap in cost function with respect to its competitors reducing cost to 37.5 (just 2% more than the target optimum) within only 280 function evaluations while all other algorithms required at least 520 function evaluations to reach the same intermediate cost function value. The hybrid harmony search algorithm of Ref. [81] was competitive with FHGWJA only for the first 150 function evaluations, while the hybrid simulated annealing algorithm (HFSA) of Ref. [81] went close to the average convergence curve of FHGWJA at about 325 function evaluations. The $W(\vec{X}_{i,tr}) \leq 1.1W(\vec{X}_{opt})$ elitist strategy of FHGWJA allowed mAHS, mBBBC–UBS, and MsinDE to approach the cost function reduction rate of FHGWJA from 400 to 700 function evaluations, while the hybrid big bang–big crunch algorithm of Ref. [81] showed a similar convergence speed to FHGWJA after about 750 function evaluations.

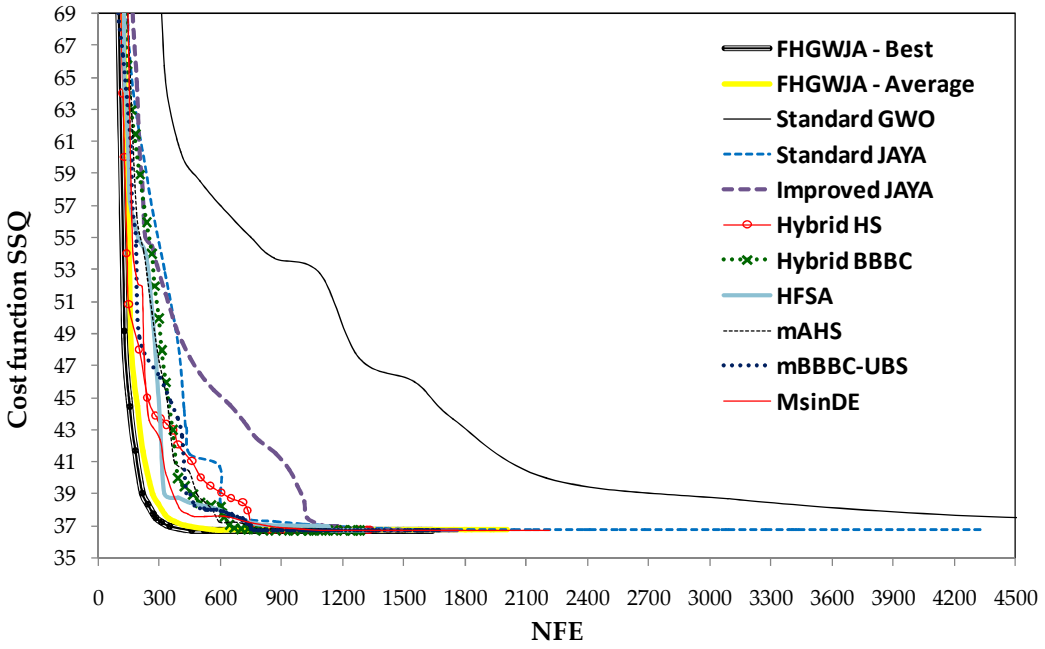

Figure 11. Comparison of convergence curves of FHGWJA and its competitors for the 3-variable hydrologic model calibration problem.

Table 6 presents the results obtained with FHGWJA and its competitors in the problem variant with 25 variables; the same nomenclature of Table 5 is utilized. For the sake of clarity, the table reports only the optimal values of channel storage S_t (included as optimization variables 4 through 25) found with FHGWJA. The present algorithm was the best optimizer also in this test problem variant. In fact, it converged to the lowest value overall of SSQ = 36.761, followed by the HFSA and hybrid HS/BBBC algorithms of Ref. [81] that obtained SSQ = 36.762 and 36.763, respectively.

Table 6. Optimization results of the hydrologic model calibration problem solved with 25 variables.

Algorithm	*K*	*h*	*M*	Best	Average	Worst	ST Dev	NFE
FHGWJA (Present) *	0.08611	0.2869	1.8681	36.761	36.761	36.761	2.919×10^{-12}	4438 ± 544 4122
Standard GWO	0.08576	0.2870	1.8694	36.818	36.870	36.934	1.395×10^{-2}	50,000
Standard JAYA	0.08620	0.2868	1.8677	36.765	36.766	36.769	1.435×10^{-3}	22,663 ± 8359 13,820
Improved JAYA [81]	0.08591	0.2866	1.8689	36.806	36.806	36.807	1.078×10^{-7}	7276 ± 2367 5476
Hybrid HS [81]	0.08614	0.2869	1.8705	36.763	36.763	36.763	3.333×10^{-11}	4744 ± 1954 3277
Hybrid BBBC [81]	0.08567	0.2868	1.8718	36.763	36.763	36.763	8.416×10^{-11}	4863 ± 1607 3656
HFSA [81]	0.08599	0.2867	1.8670	36.762	36.762	36.762	1.032×10^{-10}	5144 ± 1733 3887
mAHS (der. [94])	0.08624	0.2862	1.8678	36.774	36.774	36.775	1.263×10^{-8}	5385 ± 760 4610
mBBBC–UBS (der. [95])	0.08621	0.2872	1.8693	36.786	36.786	36.787	1.149×10^{-8}	5146 ± 797 4488
MsinDE (der. [96])	0.08604	0.2864	1.8676	36.786	36.786	36.787	1.456×10^{-8}	5848 ± 1016 5261

* The optimal channel storage values S_t found with FHGWJA are {27.723;26.729;35.245;80.441;147.329; 199.195;244.112;279.252;296.016;300.904;291.677;263.154;229.724;187.670;146.693;107.274;75.785;54.842;41.014;30.810; 25.732;20.491} m^3.

Standard JAYA ranked 5th overall in terms of SSQ but required on average 5.1-times more function evaluations than FHGWJA to complete optimization runs. Improved JAYA and standard GWO found the worst solutions amongst all optimizers (respectively, SSQ = 36.806 and 36.818), requiring on average, respectively, about 1.64- and 11.3-times more function evaluations than the present algorithm. These results confirm the validity of the hybridization scheme implemented in FHGWJA.

The $W(\vec{X}_{i,tr}) \leq 1.1W(\vec{X}_{best})$ elitist strategy of FHGWJA significantly improved the performance of the AHS/BBBC–UBS/sinDE algorithms of Refs. [94–96] also in the 25-variables problem variant. In particular, in their best optimization runs, mAHS reduced the optimized cost from 37.078 of AHS [94] to 36.774 (i.e., the 6th best result amongst those quoted in Table 6), mBBBC–UBS from 36.826 of BBBC–UBS [95] to 36.786, and MsinDE from 37.101 of sinDE [96] to 36.786. Furthermore, mAHS, mBBBC–UBS, and MsinDE reduced average computational cost by 32.7%, 34.4%, and 28.4% with respect to the original algorithms of Refs. [94–96], thus becoming competitive enough with FHGWJA in terms of average computational speed.

The superiority of FHGWJA over its competitors is confirmed by the fact that the present algorithm achieved the lowest standard deviation on optimized cost (only 2.919×10^{-12} vs. 3.333×10^{-11} to 1.395×10^{-2} of the other algorithms). Remarkably, the worst solution of FHGWJA practically coincided with its best solution, and it was always better than the best solutions found with its competitors. Furthermore, FHGWJA was faster than the other optimizers completing on average the 20 independent optimization runs within only 4438 function evaluations vs. 4744 to 5848 evaluations of hybrid HS/BBBC/SA and modified AHS/BBBC–UBS/sinDE. Interestingly, the hybrid HS/BBBC/SA algorithms of Ref. [81] completed their fastest optimization runs within less function evaluations than FHGWJA (respectively, only 3277, 3656, and 3887 vs. 4122). However, FHGWJA found intermediate solutions corresponding to SSQ = 36.763 (i.e., the best solutions of hybrid HS and hybrid BBBC) always within only 3200 function evaluations, and SSQ = 36.762 (i.e., the best solution of HFSA) always within only 3700 function evaluations.

Figure 12 compares the optimization histories of the best runs for the algorithms listed in Table 6; the average FHGWJA convergence curve also is shown in the figure. The plot is limited to the first 4500 function evaluations of optimization history and to the 35–80 cost function interval for the sake of clarity. The convergence behavior depicted in the figure resembles that observed for the three-variable problem variant. The best and average optimization runs' convergence curves for FHGWJA again lie below those of the other algorithms practically over the whole search history. FHGWJA started its best optimization run from the very large cost of 3547.602 (i.e., about 96.5 times the target optimum of 36.761 quoted in Table 6) while the initial cost for all other optimizers ranged between 461.502 and 2063.123, except the improved JAYA variant of Ref. [81] that started from 75.965. However, the present algorithm immediately recovered the initial gap in cost function with respect to its competitors reducing cost to 37.2 (just 1.2% more than the target optimum) within only 320 function evaluations while all other algorithms required at least 995 function evaluations to reach the same intermediate cost function value. The hybrid big bang–big crunch algorithm of Ref. [81] was competitive with FHGWJA only for the first 190 function evaluations. The $W(\vec{X}_{i,tr}) \leq 1.1W(\vec{X}_{opt})$ elitist strategy of FHGWJA allowed mAHS and MsinDE to approach the cost function reduction rate of FHGWJA from 220 to 310 function evaluations, while mBBBC–UBS generated the closest intermediate designs to those of FHGWJA between 650 and 1000 function evaluations.

The excellent performance of FHGWJA in the two variants of the hydrologic model calibration problems confirms the suitability of the proposed algorithm for highly nonlinear optimization problems. Interestingly, the higher level of design freedom introduced in the optimization search by the additional 22 variables did not affect at all the robustness of FHGWJA that was able to converge practically to the same values (with at most 0.105% difference) of Muskingum model parameters (i.e., K, h, and m) and channel storage values S_t found in the three-variables problem variant.

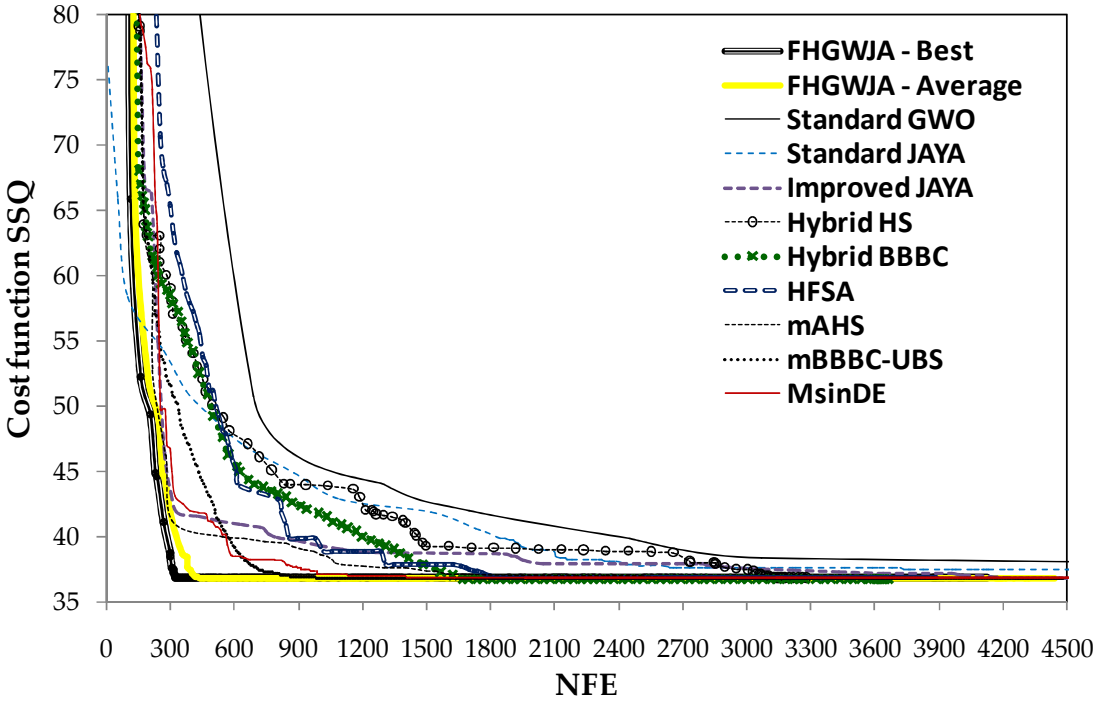

Figure 12. Comparison of convergence curves of FHGWJA and its competitors for the 25-variable hydrologic model calibration problem.

3.4. Optimal Crashworthiness Design of a Vehicle Subject to Side Impact

The test case (vi) solved in this study regarded the optimal crashworthiness design of a vehicle. The goal was to minimize the weight of the vehicle subject to side impact. Crashworthiness optimization must ensure that the vehicle effectively absorbs and dissipates impact energy, minimizing forces transferred to occupants. The vehicle structure must be optimized to provide maximum protection without significantly increasing weight or compromising other performance aspects [107–109]. Figure 13a shows the typical finite element schematization of a vehicle subject to side impact by the barrier [108], while Figure 13b illustrates the vehicle frame parts to be optimized. Since in side impact tests the B-pillar zone often is most affected, design variables are correlated with this zone.

The optimal crashworthiness design problem solved in this study included 11 optimization variables of which 7 related with the vehicle's structural geometry: B-pillar inner thickness (x_1), B-pillar reinforcement thickness (x_2), floor side inner thickness (x_3), cross members thickness (x_4), door beam thickness (x_5), door beltline reinforcement (x_6), and roof rail thickness (x_7). Two discrete variables refer to material selection and quantify the efficiency of the structure in absorbing energy during impact: shape factor for the B-pillar inner (x_8) and shape factor for the floor side inner (x_9). The last two variables are related with the presence of the obstacle: the barrier height (x_{10}), and the hitting position of the barrier with respect to the center of mass of the vehicle (x_{11}). In summary, the optimization problem includes nine continuous and two discrete variables.

Here, the optimal crashworthiness design problem was stated as follows:

$$Minimize\ W(\vec{X}) = 1.98 + 4.90x_1 + 6.67x_2 + 6.98x_3 + 4.01x_4 + 1.78x_5 + 2.73x_7 \quad (26)$$

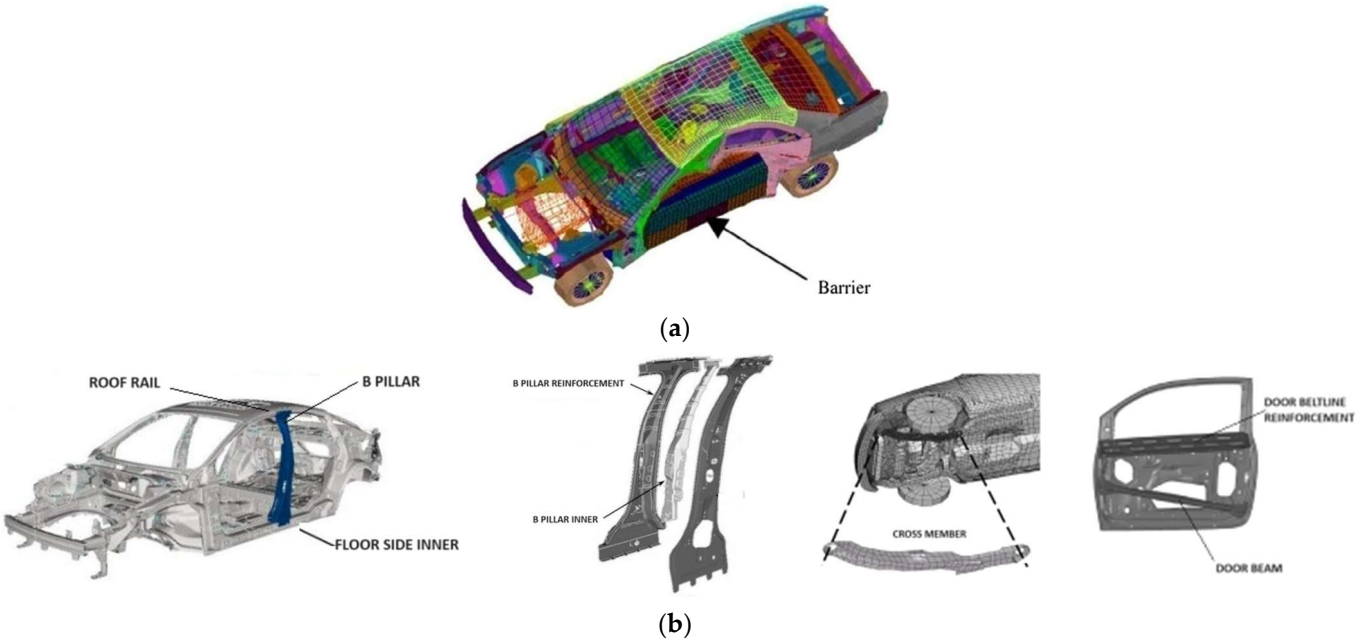

Figure 13. (**a**) Finite element model of vehicle subject to side impact (taken from [108]); (**b**) schematic of vehicle parts to be optimized.

Subject to:

$$\begin{cases} G_1(\vec{X}) = & 1.16 - 0.3717x_2x_4 - 0.00931x_2x_{10} - 0.484x_3x_9 + 0.01343x_6x_{10} \leq 1 \\ G_2(\vec{X}) = & 0.261 - 0.0159x_1x_2 - 0.188x_1x_8 - 0.019x_2x_7 + 0.0144x_3x_5 + 0.0008757x_5x_{10} + \\ & 0.08045x_6x_9 + 0.00139x_8x_{11} + 0.00001575x_{10}x_{11} \leq 0.32 \\ G_3(\vec{X}) = & 0.214 + 0.00817x_5 - 0.131x_1x_8 - 0.0704x_1x_9 + 0.03099x_2x_6 - 0.018x_2x_7 + \\ & 0.0208x_3x_8 + 0.121x_3x_9 - 0.00364x_5x_6 + 0.0007715x_5x_{10} - 0.0005354x_6x_{10} + \\ & 0.00121x_8x_{11} + 0.00184x_9x_{10} - 0.02{x_2}^2 \leq 0.32 \\ G_4(\vec{X}) = & 0.074 - 0.61x_2 - 0.163x_3x_8 + 0.001232x_3x_{10} - 0.166x_7x_9 + 0.227{x_2}^2 \leq 0.32 \\ G_5(\vec{X}) = & 28.98 + 3.818x_3 - 4.2x_1x_2 + 0.0207x_5x_{10} + 6.63x_6x_9 - 7.7x_7x_8 + 0.32x_9x_{10} \leq 32 \\ G_6(\vec{X}) = & 33.86 + 2.95x_3 + 0.1792x_{10} - 5.057x_1x_2 - 11.0x_2x_8 - 0.0215x_5x_{10} - 9.98x_7x_8 + \\ & 22.0x_8x_9 \leq 32 \\ G_7(\vec{X}) = & 46.36 - 9.9x_2 - 12.9x_1x_8 + 0.1107x_3x_{10} \leq 32 \\ G_8(\vec{X}) = & 4.72 - 0.5x_4 - 0.19x_2x_3 - 0.0122x_4x_{10} + 0.009325x_6x_{10} + 0.000191{x_{11}}^2 \leq 4 \\ G_9(\vec{X}) = & 10.58 - 0.674x_1x_2 - 1.95x_2x_8 + 0.02054x_3x_{10} - 0.0198x_4x_{10} + 0.028x_6x_{10} \leq 9.9 \\ G_{10}(\vec{X}) = & 16.45 - 0.489x_3x_7 - 0.843x_5x_6 + 0.0432x_9x_{10} - 0.0556x_9x_{11} - 0.000786{x_{11}}^2 \leq 15.7 \end{cases} \quad (27)$$

The expressions written above for the objective function $W(\vec{X})$ (i.e., the structural weight of vehicle parts that must be optimized) and the constraints $G(\vec{X})$ were fitted in [108] using response surface models. The 10 inequality constraint functions stated by Equation (27) define limitations on the force transferred to the dummy's abdomen (G_1), velocities of the dummy's upper/middle/lower chest (respectively, G_2, G_3, and G_4), deflections of the dummy's upper/middle/lower ribs (respectively, G_5, G_6, and G_7), force transferred to the dummy's pubic symphysis (G_8), velocity at B-pillar middle-point (G_9), and velocity at B-pillar front door (G_{10}). Forces are expressed in kN, velocities in mm/s, and deflections in mm.

The side constraints on optimization variables are as follows: $0.5 \leq x_1,x_2,x_3,x_4,x_5,x_6,x_7 \leq 1.5$ mm (continuous variables); $x_8,x_9 \in \{0.192;0.345\}$ (discrete variables); $-30 \leq x_{10},x_{11} \leq 30$ mm (continuous variables).

The best solution available in the literature for the optimal crashworthiness problem is that obtained with the multi-strategy fusion improved gray wolf optimization (IGWO) algorithm with a structural weight of 21.39473 kg [69]. In particular, IGWO outperformed the chaotic GWO, CSO, and tunicate swarm algorithm (TSA) that converged to optimized structural weights between 21.46164 and 22.70296 kg. The adaptive dynamic self-learning grey wolf optimization algorithm (ASGWO) [70] obtained a slightly higher optimized weight than TSA, 22.87188 kg. The starling murmuration optimizer (SMO) of Ref. [47] converged to the optimized weight of 22.84298 kg, practically the same as the improved continuous ant colony optimization algorithm ($LIACO_R$) that obtained 22.84299 kg. The other metaheuristic algorithms tested in [47] (i.e., the krill herd algorithm (KH), the best variant of artificial bee colony (ABC), the Harris hawks optimizer (HHO), the comprehensive learning particle swarm optimizer (CLPSO), and the whale optimization algorithm (WOA)) found optimized weights ranging between 22.88596 and 23.12717 kg.

Table 7 compares the optimal solution obtained with the FHGWJA algorithm developed in this study with those of the aforementioned optimizers. The results are grouped as follows: (i) FHGWJA and its component algorithms, the standard GWO and JAYA; (ii) IGWO and its three best competitors compared in [69]; (iii) ASGWO [70]; (iv) SMO and its six best competitors compared in [47]. All designs reported in the table are practically feasible as the maximum constraint violation was 0.001%. Details on computational cost and statistical performance for independent optimization runs are reported below the table when available.

Table 7. Comparison of optimized designs obtained with FHGWJA and its competitors in the optimal crashworthiness problem.

Algorithms	x_1 (mm)	x_2 (mm)	x_3 (mm)	x_4 (mm)	x_5 (mm)	x_6 (mm)	x_7 (mm)	x_8	x_9	x_{10} (mm)	x_{11} (mm)	Structural Weight (kg)
FHGWJA * Present	0.50000	1.21204	0.50000	0.77908	0.50000	1.49004	0.50000	0.345	0.345	−28.9781	0.00010	21.38340
Standard GWO *	0.50002	1.11343	0.50000	1.30794	0.50041	1.5	0.50011	0.345	0.192	−20.0960	−0.86260	22.84755
Standard JAYA *	0.50000	1.22573	0.50000	1.20711	0.50000	1.49194	0.50000	0.345	0.345	0.00000	0.00000	23.19115
IGWO [69] ♣	0.50000	0.88641	0.50000	1.25781	0.64809	0.91372	0.50000	1.00000	0.52483	1.94790	15.3719	21.39473
Chaotic GWO ♣	0.84252	0.50000	0.50000	1.36186	0.83852	0.86445	0.57679	0.94066	0.24622	4.55062	12.3084	21.46164
CSO ♣	0.78544	0.56882	0.50000	1.34514	0.82107	0.86975	0.74586	0.89302	0.39281	0.89510	1.17997	22.00444
TSA ♣	0.50315	0.89471	0.50000	1.40533	0.86396	0.84245	0.59580	0.86533	0.06579	0.12364	2.06338	22.70296
ASGWO [70] ♦	0.50004	1.13454	0.50009	1.27905	0.50020	1.49996	0.50005	0.34496	0.33248	−16.3332	−2.14912	22.87188
SMO [47] **	0.5000	1.11634	0.5000	1.30224	0.5000	1.50000	0.5000	0.345	0.345	−19.566	0.000001	22.84298
$LIACO_R$ **	0.5000	1.11593	0.5000	1.30293	0.5000	1.50000	0.5000	0.192	0.345	−19.640	−0.000003	22.84299
KH **	0.5000	1.14747	0.5000	1.26118	0.5000	1.5000	0.5000	0.345	0.345	−13.998	−0.8984	22.88596
Best ABC **	0.5000	1.30539	0.5000	1.10312	0.5000	0.50000	0.5000	0.345	0.345	14.213	20.3306	22.88605
HHO **	0.5000	1.15627	0.5000	1.27133	0.5000	1.47770	0.5000	0.345	0.192	−14.592	−2.4898	22.98537
CLPSO **	0.5061	1.17379	0.5013	1.24706	0.5037	1.49560	0.5000	0.345	0.345	−9.5985	3.3627	23.06244
WOA **	0.5000	1.09276	0.5000	1.41233	0.5000	1.45497	0.5000	0.345	0.192	−24.038	−3.1789	23.12717

* Optimized weight: 21.3905 ± 0.007125 kg for FHGWJA; 22.9318 ± 0.1423 kg for the standard GWO; 23.2067 ± 0.02431 kg for standard JAYA; computational cost: 1309 ± 343 structural analyses for FHGWJA; 50,000 for standard GWO; 7560 ± 998 for standard JAYA; the best optimization run of FHGWJA was completed within 1367 structural analyses. ♣ Computational cost: 30,000 structural analyses, determined as the product between the population size N_{POP} = 30 and the limit number of iterations $N_{itermax}$ = 1000. No statistical information on optimized weight and computational cost were given in [69]. ♦ computational cost: 15,000 structural analyses, determined as the product between the population size N_{POP} = 30 and the limit number of iterations $N_{itermax}$ = 500. No statistical information on optimized weight and computational cost were given in [70]. ** computational cost: 30,000 structural analyses, determined as the product between the population size N_{POP} = 20 and the limit number of iterations $N_{itermax}$ = 1500. No statistical information on optimized weight and computational cost were given in [47].

It can be seen that FHGWJA again was the best optimizer also for this test case. In fact, the present algorithm converged to the very low structural weight of 21.38340 kg while the optimized weights obtained with the other algorithms ranged between 21.39473 kg

(IGWO used in [69]) and 23.12717 kg (WOA used in [70]). The chaotic GWO (used in [69]) ranked 3rd overall, finding a very close structural weight to those of FHGWJA and IGWO: 21.46164 kg vs. 21.38340 and 21.39473 kg, respectively. However, the optimized values of variables x_8 and x_9 listed in Table 7 for the IGWO and chaotic GWO do not coincide with the available discrete values {0.192;0.345} originally set for this test problem: the optimized designs of the IGWO and GWO would become infeasible as soon as x_8 is set equal to 0.192 or 0.345, thus violating constraint functions G_6, G_7, G_8, and G_9 by up to 13.9 and 23.7%, respectively.

Interestingly, the average optimized weight achieved with FHGWJA over the 20 independent runs was lower than the best solutions of IGWO and chaotic GWO: only 21.3905 kg vs. 21.39473 and 21.46164 kg, respectively. Furthermore, the present algorithm required on average only 1309 structural analyses vs. the 30,000 analyses indicated in [69] for the IGWO and chaotic GWO. The hybrid optimizer proposed here clearly outperformed the standard GWO and standard JAYA that obtained 22.84755 and 23.19155 kg, respectively. Furthermore, FHGWJA required on average only 1309 structural analyses vs. 50,000 of standard GWO and 7560 of standard JAYA for completing the search process. The adaptive dynamic self-learning grey wolf optimization algorithm (ASGWO) [70] ranked below the standard GWO in terms of optimized weight (22.87188 kg vs. 22.84755 kg) but it required only 15,000 structural analyses vs. 50,000 analyses of the standard GWO. In summary, using rather simple elitist strategies and properly refining/repairing trial solutions (so that they always lie on descent directions) allowed FHGWJA to outperform complex GWO schemes combining special perturbations of design variables or several strategies taken from various metaheuristic algorithms. This happened because only the present algorithm can always generate high-quality trial solutions throughout the optimization process.

The starling murmuration optimizer (SMO) and the other algorithms compared in Ref. [70] also were less efficient than FHGWJA. In fact, they required 30,000 structural analyses vs. only 1309 of the present algorithm to find heavier optimized weights than FHGWJA (i.e., from 22.84298 to 23.12717 kg vs. only 21.38340 of FHGWJA).

The present algorithm was very robust: the standard deviation on optimized weight was only 0.0333% of the average optimized weight while the ratio between the standard deviation on number of structural analyses and the average number of structural analyses was 26.2%. Statistical data relative to the other algorithms compared with FHGWJA usually were not available in the literature. This can be explained in view of the formulation of the optimal crashworthiness design problem. The cost function $W(\vec{X})$ stated by Equation (26) depends only on variables x_1, x_2, x_3, x_4, x_5, and x_7, while x_{10} is the only variable entering in the expressions of all constraint functions (see Equation (27)). Hence, the search space of this problem hosts many competitive solutions corresponding to significantly different optimized values of design variables with obvious implications in terms of statistical dispersion of solutions. For example, the optimized values of the first seven variables for the solutions listed in Table 7 varied by at most 27.8% while optimized values of variables x_8 through x_{11} varied from 36% to 259.5% where the largest variation occurred for variable x_{10} that must be adjusted to satisfy all constraints. Remarkably, FHGWJA was always able to precisely approach and efficiently explore and exploit the best region of search space in all optimization runs in view of its inherent ability to generate high-quality trial solutions over the whole search process. This reduced by a great deal the standard deviation of the optimized solution.

The excellent convergence behavior and robustness of FHGWJA is confirmed in Figure 14, which compares the optimization history of the proposed algorithm and its component algorithms GWO and JAYA. The convergence curves relative to the best and average optimization runs of FHGWJA are shown in the figure while those of the other algorithms listed in Table 7 were not reported in [47,69,70]. For the sake of clarity, the plot is limited to the first 1800 structural analyses. FHGWJA started its best optimization run from the structural weight of 40.032 kg, which is about 87.2% larger than the best optimized weight of 21.383 kg found by the present algorithm. In spite of this, in its best run, FHGWJA

generated a feasible intermediate design just 1.48% worse than the global optimum after only 350 structural analyses, that is at approximately 26% of the optimization process. Furthermore, the best and average optimization runs' convergence curves for FHGWJA practically coincided after only 830 structural analyses, that is at just 60.7% of the best run's optimization history.

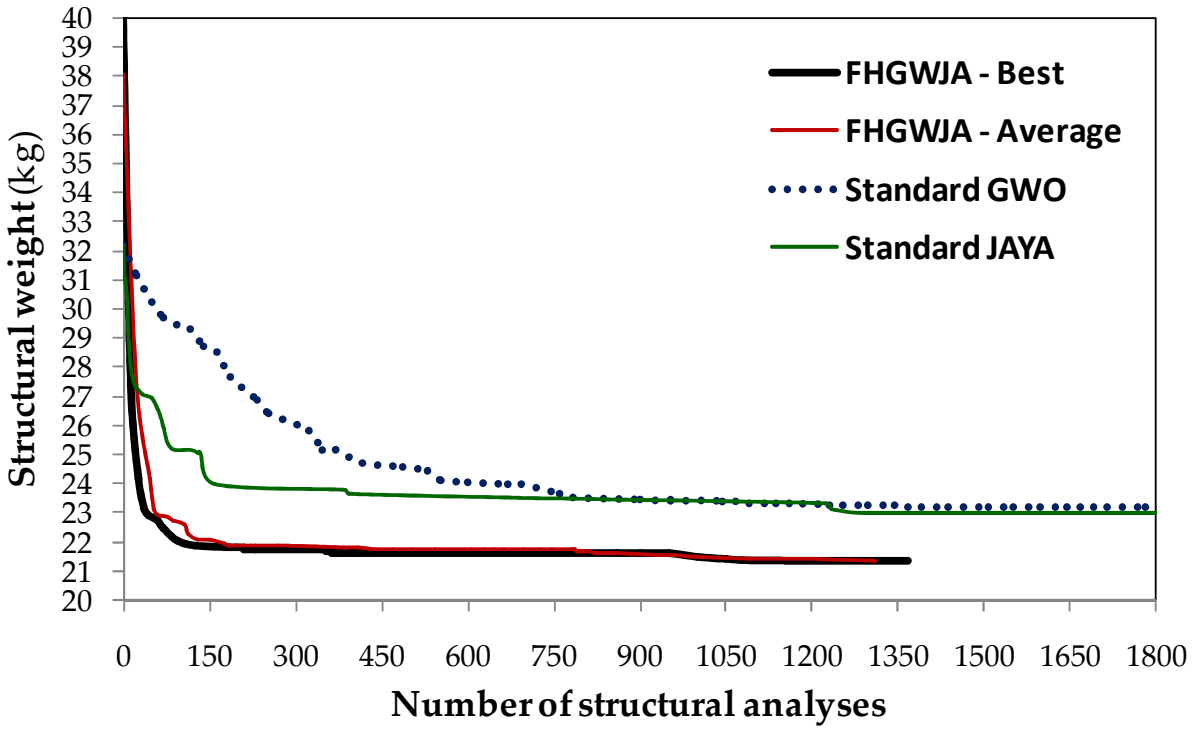

Figure 14. Comparison of convergence curves of FHGWJA and its competitors for the optimal crashworthiness problem.

The results presented in this section confirm once again the suitability of FHGWJA for solving complex engineering optimization problems. This conclusion is supported by the fact that the present algorithm was compared with another 14 (actually 28, considering all methods evaluated in Refs. [47,69,70]) state-of-the-art metaheuristic algorithms also including three advanced GWO variants developed just a few months before the present study.

3.5. Weight Minimization of a Planar 200-Bar Truss Structure

The last test case solved in this study regarded the weight minimization of the planar 200-bar truss structure shown in Figure 15. The structure, made of steel (Young's modulus of 206.91 GPa; mass density of 7833.413 kg/m^3), is composed of 200 elements connected by 77 nodes. Because of structural symmetry, the 200 elements are categorized in 29 groups as indicated in Table 8: the elements of each group have the same cross-sectional area, which is taken as a sizing optimization variable. Hence, this test problem has 29 design variables.

The structure is subjected to three independent loading conditions:

(a) Concentrated forces of 4.45 kN (i.e., 1000 lbf) acting in the positive X-direction at nodes 1, 6, 15, 20, 29, 34, 43, 48, 57, 62, and 71 (denoted by the green horizontal arrows in Figure 15);

(b) Concentrated forces of 44.497 kN (i.e., 10,000 lbf) acting in the negative Y-direction at nodes 1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 15, 16, 17, 18, 19, 20, 22, 24, 26, 28, 29, 30, 31, 32, 33, 34, 36, 38, 40, 42, 43, 44, 45, 46, 47, 48, 50, 52, 54, 56, 57, 58, 59, 60,61,62,64, 66, 68, 70, 71, 72, 73, 74, and 75 (denoted by the vertical horizontal arrows in Figure 15);

(c) Loading conditions (a) and (b) acting together.

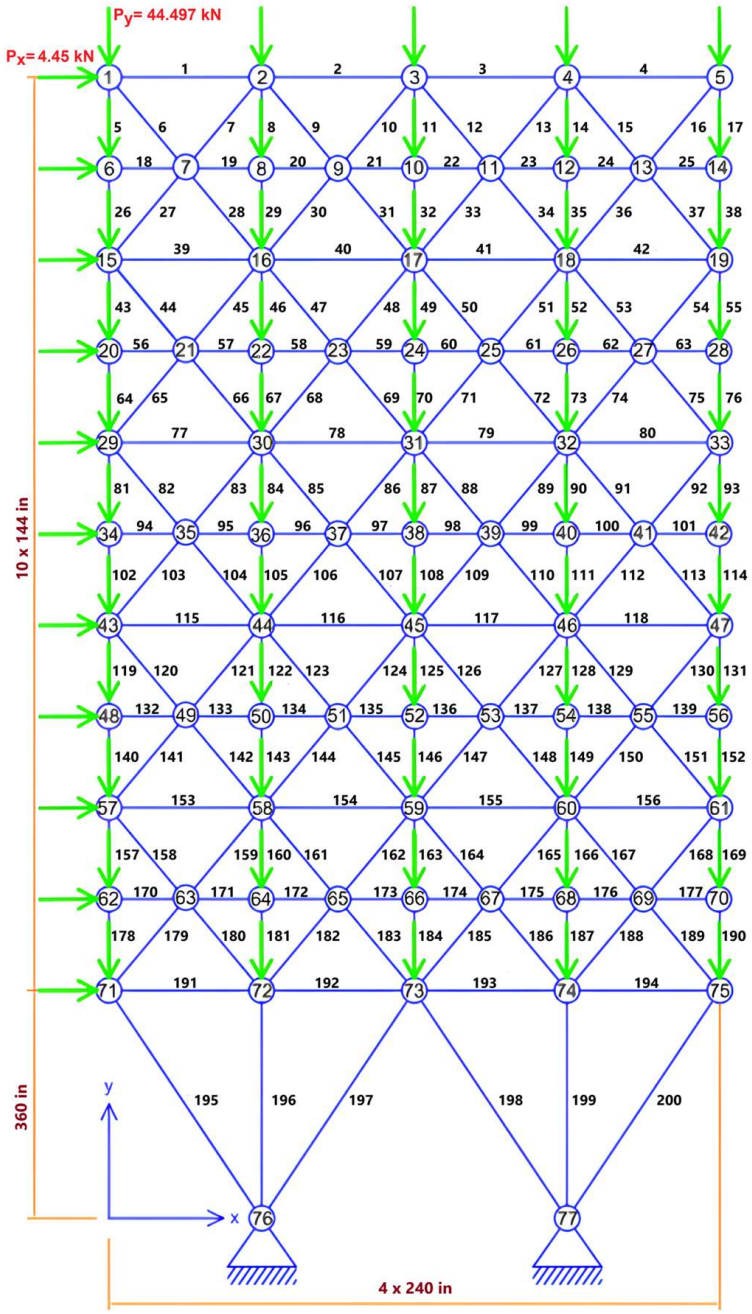

Figure 15. Schematic of the planar 200-bar truss including applied loads and kinematic constraints.

The truss weight must be minimized under 1200 nonlinear constraints on element stresses that should not exceed ±68.97 MPa (i.e., ±10,000 psi, the same limit stress under tension and compression). The cross-sectional areas taken as sizing variables can vary between 0.64516 cm^2 (i.e., 0.1 in^2) and 645.16 cm^2 (i.e., 100 in^2).

Table 8. Groups of elements defined in the 200-bar truss design problem.

Group	Elements	Group	Elements
1	1, 2, 3, 4	16	82, 83, 85, 86, 88, 89, 91, 92, 103, 104, 106, 107, 109, 110, 112, 113
2	5, 8, 11, 14, 17	17	115,116,117,118
3	19, 20, 21, 22, 23, 24	16	119, 122, 125, 128, 131
4	18, 25, 56, 63, 94, 101, 132, 139, 170, 177	19	133, 134, 135, 136, 137, 138
5	26, 29, 32, 35, 38	20	140, 143, 146, 149, 152
6	6, 7, 9, 10, 12, 13, 15, 16, 27, 28, 30, 31, 33, 34, 36, 37	21	120, 121, 123, 124, 126, 127, 129, 130, 141, 142, 144, 145, 147, 148, 150, 151
7	39, 40, 41, 42	22	153, 154,155, 156
8	43,46,49,52,55	23	157, 160, 163, 166, 169
9	57,58,59,60,61,62	24	171, 172, 173, 174, 175, 176
10	64, 67, 70, 73, 76	25	178, 181, 184, 187, 190
11	44, 45, 47, 48, 50, 51, 53, 54, 65, 66, 68, 69, 71, 72, 74, 75	26	158, 159, 161, 162, 164, 165, 167, 168, 179, 180, 182, 183, 185, 186, 188, 189
12	77, 78, 79, 80	27	191, 192, 193, 194
13	81, 84, 87, 90, 93	28	195, 197, 198, 200
14	95, 96, 97, 98, 99, 100	29	196,199
15	102, 105, 108, 111, 114		

The optimization problem was stated as follows:

$$Minimize\ W\left(\vec{X}\right) = \rho\gamma \sum_{j=1}^{NEL} A_j L_j \tag{28}$$

$$Subject\ to \begin{cases} G\left(\vec{X}\right) = \frac{\sigma_{j,ilc}}{\sigma_{t,lim}} - 1 \le 0 \\ G\left(\vec{X}\right) = \frac{\sigma_{j,ilc}}{\sigma_{c,lim}} - 1 \le 0 \end{cases} \quad \begin{cases} j = 1, \ldots, NEL \\ ilc = 1, 2, 3 \end{cases} \tag{29}$$

In Equation (28), ρ and γ, respectively, are the mass density and the specific weight of the material; NEL is the number of elements of the truss; A_j and L_j, respectively, denote the cross-sectional area and the length of the j-th element of the structure. The design vector $\vec{X}$ includes the values of the 29 sizing variables corresponding to the cross-sectional areas of the 200 truss elements.

In Equation (29), $\sigma_{j,ilc}$ denotes the stress developed in the j-th element of the structure under the ilc-th loading condition, while $\sigma_{t,lim}$ = 10,000 psi and $\sigma_{c,lim}$ = −10,000 psi, respectively, denote the limit stresses in tension and compression. A total of 1200 nonlinear constraints on element stresses are included in the optimization problem.

This design example is an average-scale structural optimization problem and has been extensively investigated in the literature. The target optimum weight for this problem is 11,542.4 kg, but there is also a local optimum corresponding to a structural weight of 11,544 kg. The non convexity of the search space makes this test problem hard to be solved with metaheuristic optimizers. The hybrid harmony search, big bang–big crunch, and simulated annealing algorithms developed in [81] converged to a structural weight of 11,542.409 kg, practically the same as the target optimum. Other very competitive solutions were obtained with the following: (i) the eigenvectors of covariance matrix (ECM) algorithm derived from the covariance matrix adaptation evolution strategy high-performance optimizer (i.e., 11,543.521 kg, [110]); (ii) the variant of success history-based adaptive differential evolution with linear population size reduction algorithm using an ensemble sinusoidal approach to automatically adapt the DE scaling factor (LSHADE-Epsin) (i.e., 11,543.980 kg, [111]); (iii) the modified coyote optimization algorithm

(MCOA) that uses chaotic sequences instead of random sequences for perturbing solutions (i.e., 11,544.007 kg, [112]); (iv) the cyclic neighborhood network topology particle swarm optimizer (CNNT-PSO) that handles population diversity based on the interactions between each particle and its neighboring individuals (i.e., 11,545.330 kg, [113]); (v) the hybrid geometric mean optimizer (hGMO) that integrates GMO with variable neighborhood to enhance exploitation (i.e., 11,545.568, [114]).

Other advanced MHAs converged to lighter designs than the 11,542.4 kg target weight, but their optimized solutions were infeasible. For example, the corrected multi-level and multi-point simulated annealing (CMLPSA) algorithm of Ref. [90], that generates the new trial design using a population of candidate designs lying on descent directions rather than developing a single point as is conducted in classical SA, found an optimized weight of 11,542.137 kg (practically the same value of target optimum weight) violating stress constraints by only 0.0709%. The hybrid HPSACO algorithm [115], combining harmony search, particle swarm, and ant colony optimization, found an optimum weight of 11,410.737 kg but stress constraints were violated by 9.97%. An improved Harris hawks optimization algorithm was used in [116]: the optimized design weighted 11,533.983 kg (i.e., 0.073% less than target optimum weight) with 9.55% violation on stress constraints. PSOHHO [117], combining the particle swarm optimization and Harris hawks optimization methods, obtained 11,374.477 kg structural weight (i.e., 1.46% reduction) but the corresponding solution violated stress constraints by about 27.6%. Finally, the biogeography-based optimization [118] reduced structural weight to 11,320.758 kg (i.e., 1.92% reduction) but the corresponding design violated stress constraints by 11.33%.

In view of the above arguments, the optimization results obtained with the FHGWJA algorithm developed in this study were compared with those of the hybrid HS/BBBC/SA [81], ECM [110], LSHADE-Epsin [111], MCOA [112], CNNT-PSO [113], and hGMO [114] algorithms providing the best feasible solutions available in the literature. Furthermore, FHGWJA was compared with the classical GWO and improved GWO [119] as well as with classical and improved JAYA [81]. The political optimizer (PO) algorithm implemented in [120] also was included in the comparison as its best optimized weight was close to those of the classical GWO and standard and improved JAYA. The comparisons presented in this section should be considered highly significant because each competitor selected from Refs. [81,110–114,119,120] was reported to outperform from 5 to 76 other state-of-the-art metaheuristic algorithms.

According to the literature, population size values for this test problem were as follows: 50 for hybrid HS [81], MCOA [112], and hGMO [114]; 100 for hybrid BBBC [81]; 20 for improved JAYA [81]; 14 for ECM [110]; 1000 for CNNT-PSO [113]; 121 for PO [120]. Such a large variation in the number of search agents confirms the difficulties encountered with metaheuristic optimizers in solving the 200-bar truss problem. Interestingly, FHGWJA could use the smallest population size overall. This was the logical consequence of the inherent ability of FHGWJA to select high-quality trial solutions throughout the search process.

Table 9 presents the optimization results obtained with FHGWJA and its competitors in the 200-bar truss problem. The table reports the optimized values of cross-sections (expressed in in^2) and the structural weight for the best run along with the number of structural analyses; the best, average, and worst values of optimized weight along with the corresponding standard deviation (*ST Dev*) over the independent runs; and the number of structural analyses (*NSA*) required in the search process. Statistical data on computational cost (i.e., average number of structural analyses and corresponding standard deviation) are reported when available. The detailed data reported for this test problem in Refs. [115–118] that present solutions violating stress limits were not replicated in Table 9 for the sake of brevity.

Table 9. Comparison of optimized designs obtained with FHGWJA and its competitors in the 200-bar truss problem.

Design Variable	FHGWJA (Present)	Standard GWO	Standard JAYA	Improved JAYA [81]	Hybrid HS/BBBC/SA [81]	ECM [110]	LSHADE-Epsin [111]	MCOA [112]	CNNT-PSO [113]	hGMO [114]	PO [120]
1	0.1484	0.1467	0.1473	0.1473	0.1484	0.1471	0.14838	0.1390	0.1482	0.1484	0.13911
2	0.9447	0.9415	0.9417	0.9404	0.9445	0.9399	0.94448	0.9355	0.9405	0.9408	0.96277
3	0.1000	0.1001	0.1002	0.1001	0.1000	0.1000	0.10000	0.1000	0.1000	0.1000	0.10996
4	0.1000	0.1006	0.1001	0.1001	0.1000	0.1000	0.10000	0.1000	0.1000	0.1000	0.10000
5	1.9434	1.9419	1.9423	1.9412	1.9445	1.9399	1.94448	1.9355	1.9408	1.9409	1.94300
6	0.2976	0.2965	0.2970	0.2968	0.2980	0.2965	0.29796	0.2909	0.2975	0.2975	0.29526
7	0.1000	0.1001	0.1001	0.1001	0.1000	0.1000	0.10000	0.1000	0.1000	0.1000	0.10006
8	3.1177	3.1147	3.1076	3.1067	3.1227	3.1049	3.12260	3.0816	3.1067	3.1097	3.09870
9	0.1000	0.1001	0.1001	0.1001	0.1000	0.1000	0.10000	0.1000	0.1000	0.1000	0.13761
10	4.1149	4.1416	4.1093	4.1081	4.1227	4.1049	4.12260	4.0816	4.1067	4.1097	4.09920
11	0.3989	0.4022	0.4034	0.4040	0.3990	0.4037	0.39897	0.3967	0.4057	0.4047	0.41926
12	0.1000	0.1871	0.1931	0.1931	0.1000	0.1906	0.10044	0.2959	0.1897	0.1722	0.15592
13	5.3783	5.4324	5.4406	5.4342	5.3934	5.4298	5.39336	5.3854	5.4343	5.4303	5.43440
14	0.1000	0.1001	0.1001	0.1001	0.1000	0.1006	0.10000	0.1000	0.1000	0.1179	0.10437
15	6.3731	6.4298	6.4360	6.4342	6.3934	6.4298	6.39336	6.3853	6.4340	6.4303	6.43550
16	0.5262	0.5728	0.5745	0.5753	0.5264	0.5739	0.52663	0.6332	0.5745	0.5745	0.56508
17	0.4521	0.1456	0.1352	0.1355	0.4353	0.1332	0.43485	0.1842	0.1366	0.1332	0.15753
18	7.9242	7.9712	7.9825	7.9802	7.5904	7.9744	7.95063	8.0396	7.9803	7.9762	7.96720
19	0.1000	0.1001	0.1002	0.1002	0.1000	0.1000	0.10000	0.1000	0.1000	0.1001	0.10012
20	8.9151	8.9715	8.9829	8.9804	8.5904	8.9744	8.95063	9.0395	8.9802	8.9762	8.96750
21	0.8691	0.7147	0.7092	0.7090	0.8592	0.7064	0.85901	0.7460	0.7109	0.7096	0.72223
22	0.1555	0.4572	0.4363	0.4372	0.1500	0.4339	0.14995	0.1306	0.4659	0.4441	0.48433
23	10.9621	10.8968	10.8940	10.8912	10.9977	10.8790	10.9977	10.9114	10.9110	10.8931	10.9130
24	0.1219	0.1001	0.1001	0.1002	0.1000	0.1000	0.100074	0.1000	0.1000	0.1000	0.10472
25	11.9512	11.8973	11.8948	11.8914	11.9977	11.8790	11.9977	11.9114	11.9112	11.8931	11.9140
26	0.9359	1.0598	1.0394	1.0491	0.9125	1.0453	0.91252	0.8627	1.0712	1.0548	1.08420
27	6.5048	6.5394	6.6153	6.6106	6.6620	6.6300	6.66179	6.9169	6.5030	6.5761	6.50930
28	10.8708	10.8368	10.8721	10.7791	10.8061	10.7827	10.8061	10.9674	10.7210	10.7574	10.7000
29	13.8713	13.9062	13.8872	13.8783	13.8236	13.8691	13.8237	13.6742	13.9310	13.8942	13.9520
Best (kg)	11,541.380	11,558.650	11,553.355	11,550.054	11,542.409 11,542.409 11,542.409	11,543.521	11,543.980	11,544.007	11,545.330	11,545.568	11,557.384
Average (kg)	11,541.716	11,572.101	11,564.050	11,556.377	11,542.410 11,542.410 11,542.411	11,580.980	11,545.681	11,576.616	11,548.048	11,609.697	11,618.419
Worst (kg)	11,542.390	11,590.931	11,579.700	11,569.059	11,542.411 11,542.410 11,542.415	N/A	11,548.448	11,592.701	11,551.227	N/A	N/A
ST DEV (kg)	0.4157	37.109	25.403	10.941	0.00141 0.00010 0.00622	19.125	0.9888	21.600	1.431	39.694	170.601
NSA	3356 ± 431 3464	50,000	40,941	31,580	2736 ± 283 * 1676 ± 137 * 5806 ± 304 *	96,600	290,000	27,720	150,000	20,000	27,984

* Best optimization run completed with Hybrid HS in 2912 structural analyses; with Hybrid BBBC in 1669 analyses; with HFSA in 5600 analyses.

It can be seen from Table 9 that the FHGWJA was the best optimizer also for this test case. In fact, it converged to the overall lowest structural weight of 11,541.380 kg. Furthermore, all optimization runs of FHGWJA found practically feasible designs (in fact, the maximum element stress evaluated for the optimized solutions of FHGWJA was only 10,000.412 psi vs. the 10,000 psi stress limit set for this problem) that were lighter than the target optimum weight of 11,542.4 kg. The hybrid HS/BBBC/SA algorithms of Ref. [81] were very competitive with the present algorithm because they always converged to the target optimum with at most 0.00622 kg standard deviation vs. 0.4157 kg of FHGWJA. The average computational cost of FHGWJA (i.e., 3356 structural analyses) practically coincides with the mean of average computational costs recorded for hybrid HS (i.e., 2736 analyses), hybrid BBBC (i.e., 1676 analyses), and hybrid fast SA (i.e., 5806 analyses).

The other best competitors of FHGWJA, namely ECM [110], LSHADE-Epsin [111], MCOA [112], CNNT-PSO [113], and hGMO [114], obtained optimized weights ranging between 11,543.521 kg (ECM) and 11,545.568 kg (hGMO) and required between 20,000 (hGMO) and 290,000 (LSHADE-Epsin) structural analyses vs. only 3356 structural analyses required on average by the present algorithm. FHGWJA clearly outperformed its component algorithms: the best solutions of improved JAYA [81], standard JAYA, and the standard GWO, respectively, achieved optimized weights of 11,550.054 kg, 11,553.355 kg and 11,558.650 kg vs. only 11,541.380 kg of FHGWJA; furthermore, they, respectively, required 31,580, 40,941, and 50,000 structural analyses vs. only 3356 analyses of FHGWJA. The improved GWO variant (IGWO) of Ref. [119], using an exponential variation scheme for the vectors $\vec{A}_1$, $\vec{A}_2$, and $\vec{A}_3$ of Equations (6) and (9), prematurely converged to the optimized weight of 11,689.883 kg within 23,760 structural analyses. The political optimizer (PO) of Ref. [120] obtained a slightly better solution than the standard GWO (the 2nd worst solution overall) weighing 11,557.384 kg vs. 11,558.650 kg within about 28,000 structural analyses vs. 50,000 analyses required for the standard GWO.

The present algorithm was very robust. In fact, it ranked 4th overall in terms of standard deviation on optimized weight but it achieved a rate of success of 100% in finding slightly lighter designs than the target global optimum of 11,542.4 kg, and it never got trapped in the 11,544 kg local minimum of structural weight. Furthermore, the ratio between standard deviation of computational cost and average computational cost achieved with FHGWJA was only 12.8%.

The $W(\vec{X}_{i,tr}) \leq 1.1W(\vec{X}_{best})$ elitist strategy of FHGWJA was again effective in improving the performance of the AHS/BBBC–UBS/sinDE algorithms of Refs. [94–96], which were reported in [81] to obtain optimized structural weights of 11,542.410 kg (yet with 0.9998% violation on stress constraints), 11,542.410 kg (yet with 1.076% constraint violation), and 11,555.006 kg (yet with 0.806% constraint violation), respectively. The modified mAHS, mBBBC–UBS, and MsinDE variants now converged to the fully feasible design listed in Table 9 for hybrid HS/BBBC/SA weighing 11,542.409 kg. The computational cost of mAHS, mBBBC–UBS, and MsinDE decreased on average by about 30% with respect to the original algorithms, yet remaining about 3.5- to 8.5-times higher than its counterpart for FHGWJA. These results are not reported in Table 9 for the sake of brevity.

Figure 16 compares the optimization histories of the best runs for the algorithms listed in Table 9; the average FHGWJA convergence curve also is shown in the figure. The plot is limited to the first 8000 structural analyses of the optimization history and to the 11,000–31,000 kg structural weight interval for the sake of clarity. FHGWJA started its best optimization run from the very large cost of 208,700.225 kg (i.e., about 18.1 times the target optimum of 11,541.380 kg quoted in Table 9) while the initial cost for all other optimizers ranged between 157,234.459 and 185,304.986 kg, except for the ECM (eigenvectors of covariance matrix) [110] and CNNT-PSO (cyclic neighborhood network topology particle swarm optimizer) [113] algorithms that started from 20,437.901 and 11,754.998 kg, respectively. However, the present algorithm immediately recovered the initial gap in cost function with respect to its competitors: for example, within only 210 and 1330 structural analyses for ECM and CNNT-PSO, respectively. The best and average optimization runs' convergence

curves for FHGWJA again lie below those of all other algorithms practically over the whole search history. The hybrid harmony search and big bang–big crunch algorithms of Ref. [81] were competitive enough with FHGWJA approaching the best optimization run of the present algorithm after about 2000 and 1400 structural analyses, respectively, that is towards the end of their optimization histories. However, while hybrid HS/BBBC stopped to improve design after reaching their best structural weight of 11,542.409 kg, the present algorithm continued to reduce structural weight to 11,541.380 kg. The best and average optimization runs' convergence curves of FHGWJA practically coincided after about 1750 structural analyses, that is at only 50.5% of the best run's optimization history.

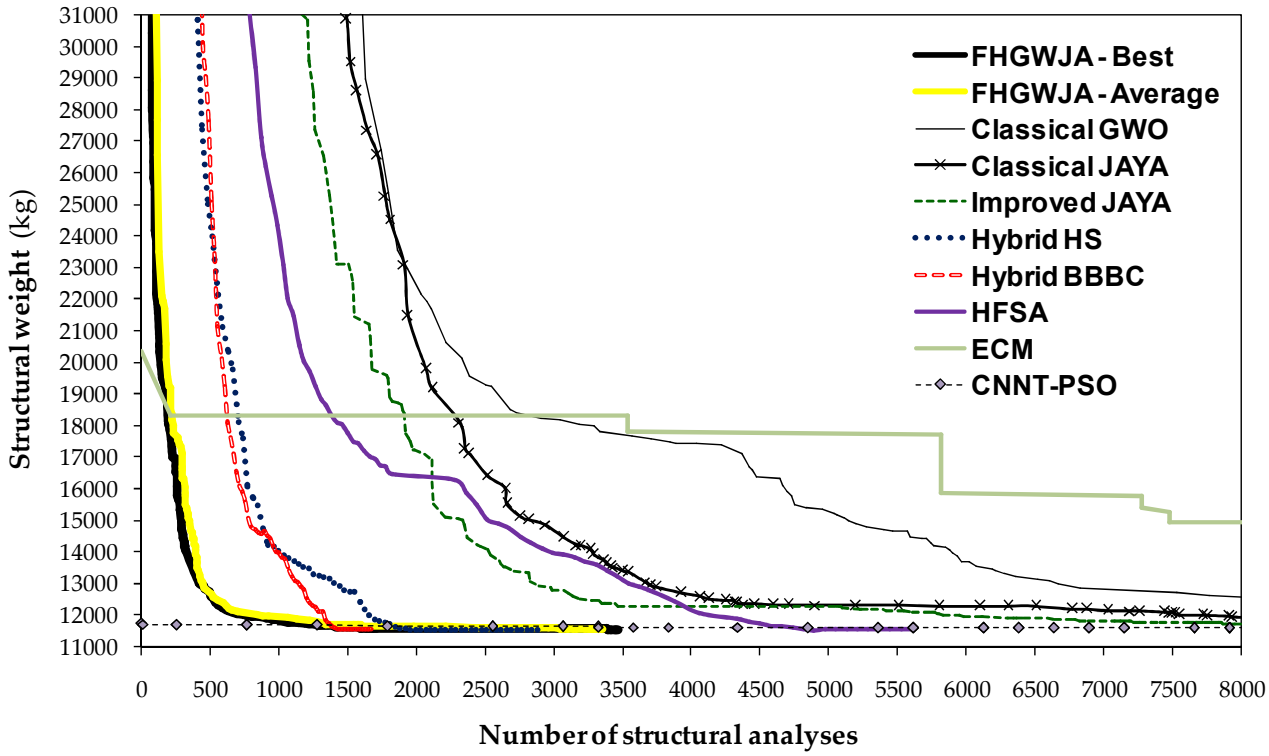

Figure 16. Comparison of convergence curves of FHGWJA and its competitors for the 200-bar optimization problem.

The data presented in this section confirmed the tendency of metaheuristic optimizers to prematurely converge to sub-optimal designs in the 200-bar truss weight minimization problem. However, FHGWJA emerged once again as the best algorithm in view of its inherent ability to generate high-quality trial solutions throughout the optimization process. Remarkably, the present algorithm was superior to high-performance optimizers such as ECM and LSHADE-Epsin that are competitive even with CEC (IEEE Congress on Evolutionary Computing) competition winners.

4. Conclusions

The novel hybrid metaheuristic algorithm FHGWJA combining the grey wolf optimizer (GWO) and JAYA was presented in this study. FHGWJA is an advanced grey wolf optimizer using elitist strategies and JAYA-based schemes to improve the current best record in each iteration. Exploration and exploitation are enhanced by forcing FHGWJA to increase diversity and select high-quality trial designs lying on descent directions. All operations carried out in FHGWJA to generate new trial solutions are computationally cheap because decision making mostly relies on evaluation of cost function rather than optimization constraints.

The present algorithm was successfully tested in seven engineering problems including up to 29 optimization variables and 1200 nonlinear constraints. Test cases covered

various fields in robotics (i.e., 2D path planning), hydrology (i.e., calibration of a nonlinear hydrologic model), and hydraulic, mechanical, and civil engineering (i.e., shape optimization of a concrete gravity dam, optimal crashworthiness design, and weight minimization of a planar 200-bar truss structure).

The extensive comparison with the optimization literature carried out in this study proved the superiority of FHGWJA over a huge number of competitors that were reported to have found the best available solutions for each test problem. In fact, the present algorithm always converged to or improved the target global optimum designs indicated in the literature. Remarkably, FHWGJA was the 1st or 2nd fastest algorithm in all test problems. The present algorithm was considerably faster than the standard GWO and JAYA formulations, and advanced variants of its component algorithms, thus confirming the efficiency of the proposed hybridization scheme. Finally, FHGWJA was very robust, achieving a high rate of success in all test problems with low standard deviations on optimized cost and number of analyses required in the search process.

In summary, FHGWJA is a very efficient tool for engineering optimization. Further research will investigate the suitability of FHGWJA for large scale optimization problems with thousands of constraints. Optimization with discrete variables also will be analyzed. In this regard, the optimal solution found with FHGWJA for the 200-bar truss problem was simply matched with a discrete set of available cross-sections obtaining competitive results with the best solution available in the literature.

Author Contributions: Conceptualization, methodology, validation, C.F. and L.L.; formal analysis, C.F. and C.I.P.; software, investigation, data curation, C.F.; writing—original draft preparation, C.F.; writing—review and editing, L.L and C.I.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data relative to this study are available upon reasonable request.

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

References

1. Goldberg, D.E. *Genetic Algorithms in Search, Optimization, and Machine Learning*; Addison-Wesley: Reading, MA, USA, 1989.
2. Holland, J.H. Genetic algorithms. *Sci. Am.* **1992**, *267*, 66–72. [CrossRef]
3. Storn, R.M.; Price, K.V. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
4. Price, K.V.; Storn, R.M.; Lampinen, J.A. *Differential Evolution a Practical Approach to Global Optimization*; Springer: Berlin/Heidelberg, Germany, 2005.
5. Yao, X.; Liu, Y.; Lin, G. Evolutionary programming made faster. *IEEE Trans. Evol. Comput.* **1999**, *3*, 82–102.
6. Beyer, H.G.; Schwefel, H.P. Evolution strategies—A comprehensive introduction. *Nat. Comput.* **2002**, *1*, 3–52. [CrossRef]
7. Simon, D. Biogeography-based optimization. *IEEE Trans. Evolution. Comp.* **2008**, *12*, 702–713. [CrossRef]
8. Kirkpatrick, S.; Gelatt, C.D., Jr.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [CrossRef] [PubMed]
9. Van Laarhoven, P.J.M.; Aarts, E.H.L. *Simulated Annealing: Theory and Applications*; Kluwer Academic Publishers: Dordrecht, The Netherlands, 1987.
10. Kaveh, A.; Talatahari, S. A novel heuristic optimization method: Charged system search. *Acta Mech.* **2010**, *213*, 267–289. [CrossRef]
11. Kaveh, A.; Motie Share, M.A.; Moslehi, M. Magnetic charged system search: A new meta-heuristic algorithm for optimization. *Acta Mech.* **2013**, *224*, 85–107. [CrossRef]
12. Kaveh, A.; Mahdavi, V.R. Colliding bodies optimization: A novel meta-heuristic method. *Comput. Struct.* **2014**, *139*, 18–27. [CrossRef]
13. Kaveh, A.; Bakhshpoori, T. A new metaheuristic for continuous structural optimization: Water evaporation optimization. *Struct. Multidiscip. Optim.* **2016**, *54*, 23–43. [CrossRef]
14. Kaveh, A.; Dadras, A. A novel meta-heuristic optimization algorithm: Thermal exchange optimization. *Adv. Eng. Softw.* **2017**, *110*, 69–84. [CrossRef]
15. Faramarzi, A.; Heidarinejad, M.; Stephens, B.; Mirjalili, S. Equilibrium optimizer: A novel optimization algorithm. *Knowl. Based Syst.* **2020**, *191*, 105190. [CrossRef]
16. Abdechiri, M.; Meybodi, M.R.; Bahrami, H. Gases Brownian motion optimization: An algorithm for optimization (GBMO). *Appl. Soft Comput.* **2013**, *13*, 2932–2946. [CrossRef]

17. Hashim, F.A.; Houssein, E.H.; Mabrouk, M.S.; Al-Atabany, W.; Mirjalili, S. Henry gas solubility optimization: A novel physics based algorithm. *Fut. Gen. Comput. Syst.* **2019**, *101*, 646–667. [CrossRef]
18. Kaveh, A.; Khayat Azad, M. A new meta-heuristic method: Ray optimization. *Comput. Struct.* **2012**, *112–113*, 283–294. [CrossRef]
19. Abdel-Basset, M.; Mohamed, R.; Sallam, K.M.; Chakrabortty, R.K. Light spectrum optimizer: A novel physics-inspired metaheuristic optimization algorithm. *Mathematics* **2022**, *10*, 3466. [CrossRef]
20. Erol, O.K.; Eksin, I. A new optimization method: Big Bang-Big Crunch. *Adv. Eng. Softw.* **2006**, *37*, 106–111. [CrossRef]
21. Rashedi, E.; Nezamabadipour, H.; Saryazdi, S. GSA: A gravitational search algorithm. *Inform. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
22. Mirjalili, S. SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowl. Based Syst.* **2016**, *96*, 120–133. [CrossRef]
23. Ahmadianfar, I.; Heidari, A.A.; Gandomi, A.H.; Chu, X.; Chen, H. RUN beyond the metaphor: An efficient optimization algorithm based on Runge Kutta method. *Expert Syst. Appl.* **2021**, *181*, 115079. [CrossRef]
24. Abualigah, L.; Diabat, A.; Mirjalili, S.; Elaziz, M.A.; Gandomi, A.H. The arithmetic optimization algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609. [CrossRef]
25. Glover, F.; Laguna, M. *Tabu Search*; Kluwer Academic Publishers: Boston, MA, USA, 1997.
26. Geem, Z.W.; Kim, J.H.; Loganathan, G. A new heuristic optimization algorithm: Harmony search. *Simulation* **2001**, *76*, 60–68. [CrossRef]
27. Rao, R.V.; Savsani, V.J.; Vakharia, D.P. Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems. *CAD Comput. Aided Des.* **2011**, *43*, 303–315. [CrossRef]
28. Rao, R.V. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *Int. J. Ind. Eng. Comp.* **2016**, *7*, 19–34.
29. Zhang, Y.; Jin, Z. Group teaching optimization algorithm: A novel metaheuristic method for solving global optimization problems. *Expert Syst. Appl.* **2020**, *148*, 113246. [CrossRef]
30. Matoušová, I.; Trojovský, P.; Dehghani, M.; Trojovská, E.; Kostra, J. Mother optimization algorithm: A new human-based metaheuristic approach for solving engineering optimization. *Sci. Rep.* **2023**, *13*, 10312. [CrossRef] [PubMed]
31. Trojovský, P. A new human-based metaheuristic algorithm for solving optimization problems based on preschool education. *Sci. Rep.* **2023**, *13*, 21472. [CrossRef] [PubMed]
32. Gopi, S.; Mohapatra, P. Learning cooking algorithm for solving global optimization problems. *Sci. Rep.* **2024**, *14*, 13359. [CrossRef]
33. Kaveh, A.; Talatahari, S. Optimum design of skeletal structures using imperialist competitive algorithm. *Comput. Struct.* **2010**, *88*, 1220–1229. [CrossRef]
34. Askari, Q.; Younas, I.; Saeed, M. Political Optimizer: A novel socio-inspired meta-heuristic for global optimization. *Knowl. Based Syst.* **2020**, *195*, 105709. [CrossRef]
35. Clerc, M. *Particle Swarm Optimization*; ISTE Publishing Company: London, UK, 2006.
36. Dorigo, M.; Stutzle, T. *Ant Colony Optimization*; MIT Press: Cambridge, MA, USA, 2004.
37. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [CrossRef]
38. Yang, X.-S. Firefly algorithm, stochastic test functions and design optimization. *Int. J. Bio-Inspir. Com.* **2010**, *2*, 78–84. [CrossRef]
39. Mirjalili, S. The ant lion optimizer. *Adv. Eng. Softw.* **2015**, *83*, 80–98. [CrossRef]
40. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
41. Pierezan, J.; Dos Santos Coelho, L. Coyote Optimization Algorithm: A new metaheuristic for global optimization problems. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation, Rio de Janeiro, Brazil, 8–13 July 2018; p. 8477769.
42. Hashim, F.A.; Hussien, A.G. Snake Optimizer: A novel meta-heuristic optimization algorithm. *Knowl. Based Syst.* **2022**, *242*, 108320. [CrossRef]
43. Coufal, P.; Hubálovský, S.; Hubálovská, M.; Balogh, Z. Snow Leopard Optimization algorithm: A new nature-based optimization algorithm for solving optimization problems. *Mathematics* **2021**, *9*, 2832. [CrossRef]
44. Yang, X.S.; Deb, S. Engineering optimisation by cuckoo search. *Int. J. Math. Model.* **2010**, *1*, 330–343. [CrossRef]
45. Askarzadeh, A. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. *Comput. Struct.* **2016**, *169*, 1–12. [CrossRef]
46. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. [CrossRef]
47. Zamani, H.; Nadimi-Shahraki, M.H.; Gandomi, A.H. Starling murmuration optimizer: A novel bio-inspired algorithm for global and engineering optimization. *Comput. Methods Appl. Mech. Eng.* **2022**, *392*, 114616. [CrossRef]
48. Yang, X.-S.; Gandomi, A.H. Bat algorithm: A novel approach for global engineering optimization. *Eng. Comput.* **2012**, *29*, 464–483. [CrossRef]
49. Kaveh, A.; Farhoudi, N. A new optimization method: Dolphin echolocation. *Adv. Eng. Softw.* **2013**, *59*, 53–70. [CrossRef]
50. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
51. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [CrossRef]
52. Faramarzi, A.; Heidarinejad, M.; Mirjalili, S.; Gandomi, A.H. Marine Predators Algorithm: A nature-inspired metaheuristic. *Expert Syst. Appl.* **2020**, *152*, 113377. [CrossRef]

53. Sadee, H.T.; Abdulazeez, A.M. Giant trevally optimizer (GTO): A novel metaheuristic algorithm for global optimization and challenging engineering problems. *IEEE Access* **2022**, *10*, 121615–121640. [CrossRef]
54. Knypinski, L.; Devarapalli, R.; Gillon, F. The hybrid algorithms in constrained optimization of the permanent magnet motors. *IET Sci. Meas. Technol.* **2024**, 1–7. [CrossRef]
55. Ficarella, E.; Lamberti, L.; Degertekin, S.O. Mechanical identification of materials and structures with optical methods and metaheuristic optimization. *Materials* **2019**, *12*, 2133. [CrossRef]
56. Degertekin, S.O.; Minooei, S.M.; Santoro, L.; Trentadue, B.; Lamberti, L. Large-scale truss-sizing optimization with enhanced hybrid HS algorithm. *Appl. Sci.* **2021**, *11*, 3270. [CrossRef]
57. Degertekin, S.O.; Yalcin Bayar, G.; Lamberti, L. Parameter free Jaya algorithm for truss sizing-layout optimization under natural frequency constraints. *Comput. Struct.* **2021**, *245*, 106461. [CrossRef]
58. Guo, J.; Liu, C.; Cao, J.; Jiang, D. Damage identification of wind turbine blades with deep convolutional neural networks. *Renew. Energy* **2021**, *174*, 122–133. [CrossRef]
59. Silvestrin, P.V.; Ritt, M. An iterated tabu search for the multi-compartment vehicle routing problem. *Comput. Oper. Res.* **2017**, *81*, 191–202. [CrossRef]
60. Brion, D.A.J.; Pattinson, S.W. Generalisable 3D printing error detection and correction via multi-head neural networks. *Nat. Commun.* **2022**, *13*, 4654. [CrossRef]
61. Meenachi, L.; Ramakrishnan, S. Metaheuristic search based feature selection methods for classification of cancer. *Pattern Recognit.* **2021**, *119*, 108079. [CrossRef]
62. Oliva, D.; Ortega-Sanchez, N.; Hinojosa, S.; Perez-Cisneros, M. *Modern Metaheuristics in Image Processing*; CRC Press: Boca Raton, FL, USA, 2023.
63. Sharma, I.; Kumar, V.; Sharma, S. A comprehensive survey on grey wolf optimization. *Recent Adv. Comput. Sci. Commun.* **2022**, *15*, 323–333.
64. Nadimi-Shahraki, M.H.; Taghian, S.; Mirjalili, S.; Zamani, H.; Bahreininejad, A. GGWO: Gaze cues learning-based grey wolf optimizer and its applications for solving engineering problems. *J. Comput. Sci.* **2022**, *61*, 101636. [CrossRef]
65. Tsai, H.-C.; Shi, J.-Y. Potential corrections to grey wolf optimizer. *Appl. Soft. Comp.* **2024**, *161*, 111776. [CrossRef]
66. Nadimi-Shahraki, M.H.; Taghian, S.; Mirijalili, S. An improved grey wolf optimizer for solving engineering problems. *Expert Syst. Appl.* **2021**, *166*, 113917. [CrossRef]
67. Li, K.; Li, S.; Huang, Z.; Zhang, M.; Xu, Z. Grey Wolf Optimization algorithm based on Cauchy-Gaussian mutation and improved search strategy. *Sci. Rep.* **2022**, *12*, 18961. [CrossRef]
68. Ma, S.; Fanga, Y.; Zhao, X.; Liu, Z. Multi-swarm improved Grey Wolf Optimizer with double adaptive weights and dimension learning for global optimization problems. *Math. Comput. Simul.* **2023**, *205*, 619–641. [CrossRef]
69. Qiu, Y.; Yang, X.; Chen, S. An improved gray wolf optimization algorithm solving to functional optimization and engineering design problems. *Sci. Rep.* **2024**, *14*, 14190. [CrossRef] [PubMed]
70. Zhang, Y.; Cai, Y. Adaptive dynamic self-learning grey wolf optimization algorithm for solving global optimization problems and engineering problems. *Math. Biosci. Eng.* **2024**, *21*, 3910–3943. [CrossRef]
71. Singh, N.; Singh, S.B. Hybrid algorithm of particle swarm optimization and grey wolf optimizer for improving convergence performance. *J. Appl. Math.* **2017**, *2017*, 2030489. [CrossRef]
72. Zhang, X.; Kang, Q.; Cheng, J.; Wang, X. A novel hybrid algorithm based on Biogeography-Based Optimization and Grey Wolf Optimizer. *Appl. Soft Comp.* **2018**, *67*, 197–214. [CrossRef]
73. Wang, J.-S.; Li, S.-X. An improved grey wolf optimizer based on differential evolution and elimination mechanism. *Sci. Rep.* **2019**, *9*, 7181. [CrossRef]
74. Yang, Q.; Liu, J.; Wu, Z.; He, S. A fusion algorithm based on whale and grey wolf optimization algorithm for solving real-world optimization problems. *Appl. Soft Comp.* **2023**, *146*, 110701. [CrossRef]
75. Tu, B.; Wang, F.; Huo, Y.; Wang, X. A hybrid algorithm of grey wolf optimizer and Harris hawks optimization for solving global optimization problems with improved convergence performance. *Sci. Rep.* **2023**, *13*, 22909. [CrossRef]
76. Zitar, R.A.; Al-Betar, M.A.; Awadallah, M.A.; Doush, I.A.; Assaleh, K. An intensive and comprehensive overview of JAYA algorithm, its versions and applications. *Arch. Comput. Methods Eng.* **2022**, *29*, 763–792. [CrossRef]
77. Da Silva, L.S.A.; Lucio, Y.L.S.; Coelho, L.D.; Mariani, V.C.; Rao, R.V. A comprehensive review on Jaya optimization algorithm. *Artif. Intell. Rev.* **2023**, *56*, 4329–4361. [CrossRef]
78. Rao, R.V.; Saroj, A. A self-adaptive multi-population based Jaya algorithm for engineering optimization. *Swarm Evol. Comput.* **2017**, *37*, 1–26.
79. Zhang, Y.; Chi, A.; Mirjalili, S. Enhanced Jaya algorithm: A simple but efficient optimization method for constrained engineering design problems. *Knowl.-Based Syst.* **2021**, *233*, 107555. [CrossRef]
80. Zhang, G.; Wan, C.; Xue, S.; Xie, L. A global-local hybrid strategy with adaptive space reduction search method for structural health monitoring. *Appl. Math. Model.* **2023**, *121*, 231–251. [CrossRef]
81. Ficarella, E.; Lamberti, L.; Degertekin, S.O. Comparison of three novel hybrid metaheuristic algorithms for structural optimization problems. *Comput. Struct.* **2021**, *244*, 106395. [CrossRef]
82. Degertekin, S.O.; Lamberti, L.; Ugur, I.B. Discrete sizing/layout/topology optimization of truss structures with an advanced Jaya algorithm. *Appl. Soft Comp.* **2019**, *79*, 363–390. [CrossRef]

83. Chakraborty, U.K. Semi-steady-state Jaya algorithm for optimization. *Appl. Sci.* **2020**, *10*, 5388. [CrossRef]
84. Cheng, Y.; Lyu, X.; Mao, S. Optimization design of brushless DC motor based on improved JAYA algorithm. *Sci. Rep.* **2024**, *14*, 5427. [CrossRef]
85. Zhang, H.L.; Liu, Q.; Zhang, C. Hybrid Jaya algorithm with genetic algorithm for solving non-linear optimization problems. *Math. Probl. Eng.* **2019**, *2019*, 6397802.
86. Gholami, K.; Olfat, H.; Gholami, J. An intelligent hybrid JAYA and crow search algorithms for optimizing constrained and unconstrained problems. *Soft Comput.* **2021**, *25*, 14393–14411. [CrossRef]
87. Zhang, Y.J.; Wang, Y.F.; Tao, L.W.; Yan, Y.X.; Zhao, J.; Gao, Z.M. Self-adaptive classification learning hybrid JAYA and Rao-1 algorithm for large-scale numerical and engineering problems. *Eng. Appl. Artif. Intell.* **2022**, *114*, 105069. [CrossRef]
88. Gholami, J.; Nia, F.A.; Sanatifar, M.; Zawbaa, H.M. Effective hybridization of JAYA and teaching–learning-based optimization algorithms for numerical function optimization. *Soft Comput.* **2023**, *27*, 9673–9691. [CrossRef]
89. Mlaouhi, I.; Ben Guedria, N.; Bouraoui, C. An efficient hybrid differential evolution–Jaya algorithm for enhancing vibration behaviour in automotive turbocharger systems. *Eng. Opt.* **2023**, *56*, 1493–1515. [CrossRef]
90. Lamberti, L. An efficient simulated annealing algorithm for design optimization of truss structures. *Comput. Struct.* **2008**, *86*, 1936–1953. [CrossRef]
91. Ghafil, H.N.; Jármai, K. Dynamic differential annealed optimization: New metaheuristic optimization algorithm for engineering applications. *Appl. Soft Comp.* **2020**, *93*, 106392. [CrossRef]
92. Gasparetto, A.; Zanotto, V. Optimal trajectory planning for industrial robots. *Adv. Eng. Softw.* **2010**, *41*, 548–556. [CrossRef]
93. Li, B.; Shao, Z. Simultaneous dynamic optimization: A trajectory planning method for nonholonomic car-like robots. *Adv. Eng. Softw.* **2020**, *87*, 30–42. [CrossRef]
94. Hasancebi, O.; Erdal, F.; Saka, M.P. Adaptive harmony search method for structural optimization. *ASCE J. Struct. Eng.* **2010**, *136*, 419–431. [CrossRef]
95. Kazemzadeh Azad, S.; Hasancebi, O.; Kazemzadeh Azad, S.; Erol, O.K. Upper bound strategy in optimum design of truss structures: A big bang-big crunch algorithm based application. *Adv. Struct. Eng.* **2013**, *16*, 1035–1046. [CrossRef]
96. Draa, A.; Bouzoubia, S.; Boukhalfa, I. A sinusoidal differential evolution algorithm for numerical optimisation. *Appl. Soft Comp.* **2015**, *27*, 99–126. [CrossRef]
97. MiarNaeimi, F.; Azizyan, G.; Rashki, M. Multi-level cross entropy optimizer (MCEO): An evolutionary optimization algorithm for engineering problems. *Eng. Comput.* **2018**, *34*, 719–739. [CrossRef]
98. Azizyan, G.; MiarNaeimi, F.; Rashki, M.; Shabakhty, S. Flying Squirrel Optimizer (FSO): A novel SI-based optimization algorithm for engineering problems. *Iran. J. Optim.* **2019**, *11*, 177–205.
99. Wilson, E.M. *Engineering Hydrology*; MacMillan: London, UK, 1974.
100. Tung, Y.K. River flood routing by nonlinear Muskingum method. *ASCE J. Hydraul. Eng.* **1985**, *111*, 1447–1460. [CrossRef]
101. Wang, W.-C.; Tian, W.-C.; Xu, D.-M.; Chau, K.-W.; Qiang Ma, Q.; Liu, C.-J. Muskingum models' development and their parameter estimation: A state-of-the-art review. *Water. Resour. Manag.* **2023**, *37*, 3129–3150. [CrossRef]
102. Lee, K.S.; Geem, Z.W. A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice. *Comput. Methods Appl. Mech. Eng.* **2005**, *194*, 3902–3933. [CrossRef]
103. Luo, J.; Yang, X.; Xie, J. Evaluation and improvement of routing procedure for nonlinear Muskingum models. *Int. J. Civ. Eng.* **2016**, *14*, 47–59. [CrossRef]
104. Yua, X.; Wu, X.; Tian, H.; Yuan, Y.; Adnan, R.M. Parameter identification of nonlinear Muskingum model with backtracking search algorithm. *Water. Resour. Manag.* **2016**, *30*, 2767–2783.
105. Geem, Z.W. Issues in optimal parameter estimation for the nonlinear Muskingum flood routing model. *Eng. Optimiz.* **2014**, *46*, 328–339. [CrossRef]
106. Barati, R. Parameter estimation of nonlinear Muskingum models using Nelder-Mead Simplex algorithm. *ASCE J. Hydrol. Eng.* **2014**, *16*, 946–954. [CrossRef]
107. Gu, L.; Yang, R.; Tho, C.-H.; Makowskit, M.; Faruquet, O.; Li, Y.L.Y. Optimisation and robustness for crashworthiness of side impact. *Int. J. Veh. Des.* **2001**, *26*, 348–360. [CrossRef]
108. Youn, B.D.; Choi, K.K. A new response surface methodology for reliability-based design optimization. *Comput. Struct.* **2004**, *82*, 241–256. [CrossRef]
109. Youn, B.D.; Choi, K.; Yang, R.-J.; Gu, L. Reliability-based design optimization for crashworthiness of vehicle side impact. *Struct. Multidiscip. Optim.* **2004**, *26*, 272–283. [CrossRef]
110. Pouriyanezhad, E.; Rahami, H.; Mirhosseini, S.M. Truss optimization using eigenvectors of the covariance matrix. *Eng. Comput.* **2021**, *37*, 2207–2224. [CrossRef]
111. Ozturk, H.T.; Kahraman, H.T. Meta-heuristic search algorithms in truss optimization: Research on stability and complexity analyses. *Appl. Soft Comp.* **2023**, *145*, 110573. [CrossRef]
112. Pierezan, J.; Dos Santos Coelho, L.; Cocco Mariani, V.; Vasconcelos Segundo, E.H.; Prayogo, D. Chaotic coyote algorithm applied to truss optimization problems. *Comput. Struct.* **2021**, *242*, 106353. [CrossRef]
113. Kim, T.-H.; Byun, J.-I. Truss sizing optimization with a diversity-enhanced cyclic neighborhood network topology particle swarm optimizer. *Mathematics* **2020**, *8*, 1087. [CrossRef]

114. Pham, V.H.S.; Nguyen Dang, N.T.; Nguyen, V.N. Efficient truss design: A hybrid geometric mean optimizer for better performance. *Appl. Comput. Intell. Soft Comput.* **2024**, *2024*, 4216718. [CrossRef]
115. Kaveh, A.; Talatahari, S. Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. *Comput. Struct.* **2009**, *87*, 267–283. [CrossRef]
116. Yassami, M.; Ashtari, P. PSOHHO hybrid optimization algorithm for truss optimization. *AUT J. Civil Eng.* **2022**, *6*, 295–318.
117. Khajeh, A.; Kiani, A.; Seraji, M.; Dashti, H. Weight minimization of truss structures using an improved Harris hawks optimization algorithm. *Innov. Infrastruct. Solut.* **2023**, *8*, 112. [CrossRef]
118. Massah, S.R.; Ahmadi, H. Weight optimization of truss structures by the biogeography-based optimization algorithm. *Civil Eng. Infrastruct. J.* **2021**, *54*, 129–144.
119. Kaveh, A.; Zakian, P. Improved GWO algorithm for optimal design of truss structures. *Eng. Comput.* **2018**, *34*, 685–707. [CrossRef]
120. Awad, R. Sizing optimization of truss structures using the political optimizer (PO) algorithm. *Structures* **2021**, *33*, 4871–4894. [CrossRef]