

Article

A Novel Hybrid Metaheuristic Algorithm for Real-World Mechanical Engineering Optimization Problems

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.it² College of Engineering, Design and Physical Sciences, Mechanical and Aerospace Engineering, Brunel University London, Kingston Ln, Uxbridge UB83PH, UK

* Correspondence: c.furio1@phd.poliba.it (C.F.); catalin.pruncu@brunel.ac.uk (C.I.P.)

Abstract

Real-world constrained optimization problems often are highly nonlinear and present non-convex design spaces. Metaheuristic algorithms (MHOAs) are naturally suited to solving real-world optimization problems in view of their global optimization capability, but may require too many analyses to complete the optimization process. Hybrid methods enhance searching by combining two or more algorithms to better balance exploration and exploitation. Elitist strategies may be utilized to generate high-quality trial designs, yet with no guarantee that each new design always improves the current best record. In order to solve these issues and minimize the number of analyses, this study presents the novel HALSGWJA (Hybrid Approximate Line Search Grey Wolf JAYA) algorithm. HALSGWJA combined grey wolf optimizer (GWO) and JAYA (two powerful MHOAs still attracting optimization experts), enhanced by approximate line search. HALSGWJA utilized approximate gradient information to perform line searches, providing descent directions with respect to the current best record. This results in a complete renewal of the current population and a much higher probability of improving all individuals with respect to the previous iteration. The proposed HALSGWJA algorithm was successfully tested on 20 real-world mechanical engineering problems: (i) the CEC2020 test suite of 19 real-world mechanical engineering examples with up to 30 optimization variables and 86 nonlinear constraints and (ii) the optimal crashworthiness design of a vehicle subject to side impact with 11 optimization variables and 10 highly nonlinear constraints. Sizing and topology optimization problems, as well as problems with discrete variables, were considered. Remarkably, HALSGWJA outperformed 18 other state-of-the-art metaheuristic algorithms in the CEC2020 problems and 25 other algorithms in the crashworthiness design problem. HALSGWJA practically converged to target optima in all test cases (the largest penalty on target optimized cost was only 0.0263% in problem 13 of the CEC2020 library). Furthermore, it obtained in many cases 0 or nearly 0 standard deviation on optimized cost. Lastly, HALSGWJA always ranked first in terms of computational speed, requiring fewer analyses than its competitors and exhibiting, in most cases, a moderate dispersion on the number of analyses entailed by the optimization process.

Keywords: real-world optimization; hybrid metaheuristic algorithms; GWO; JAYA; approximate line search; mechanical engineering problems; crashworthiness optimization

Academic Editor: Lei Wang

Received: 27 September 2025

Revised: 14 November 2025

Accepted: 20 November 2025

Published: 27 November 2025

Citation: Furio, C.; Lamberti, L.; Pruncu, C.I. A Novel Hybrid Metaheuristic Algorithm for Real-World Mechanical Engineering Optimization Problems. *Appl. Sci.* **2025**, *15*, 12580. <https://doi.org/10.3390/app152312580>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Real-world optimization problems are formulated in terms of (i) practical design features representing the tasks to be accomplished and (ii) design variables that define the configuration of the particular scenario to be optimized. The goal is to minimize or maximize an NDV -variables cost function $W(\vec{X})$ subject to a set of N_{CON} inequality/equality constraint functions, $G(\vec{X})$ or $H(\vec{X})$. Real-world optimization problems are usually non-linear with respect to the cost function and constraints and present non-convex design spaces.

Gradient-based optimization algorithms (GBOAs) or metaheuristic optimization algorithms (MHOAs) may be used for solving real-world optimization problems. GBOAs formulate and solve a sequence of approximate sub-problems using gradient information and perform line searches to improve design in the current iteration. MHOAs replace local information provided by gradients at some specific point of design space with global information relative to a population of candidate designs distributed over the search space. MHOAs randomly perturb design mimicking natural phenomena, physics/chemistry/mathematics laws, evolution, animal behavior, human behavior and activities, social sciences, etc. The quality of candidate designs included in the population is progressively improved as the search process converges to the optimal solution.

Metaheuristic algorithms are better suited for real-world optimization problems than gradient-based algorithms in view of their ability to globally explore the search space, avoiding local optima towards which the search may be directed by some particular gradient configurations. Furthermore, metaheuristics can efficiently deal with discrete variables and non-convex search spaces that often characterize real-world design problems. MHOAs combine exploration and exploitation. The former governs early optimization stages; variables are largely perturbed to quickly approach the best regions of the search space. The latter governs the final stages of the optimization process, carrying out a local search in the neighborhood of the most promising solutions. Exploration and exploitation must be properly balanced to find the global optimum with low computational effort.

MHOAs broadly include four types of algorithms: (i) evolutionary, (ii) science-based, (iii) human-based, (iv) swarm intelligence-based. Evolutionary algorithms rely on evolution theory and mimic evolutionary mechanisms. Science-based MHOAs are inspired by physics, chemistry, astronomy and astrophysics, and mathematics laws. Human-based algorithms mimic human behaviors, including learning/education, social interactions, international relationships, and political activity. Swarm intelligence-based algorithms simulate the social/individual behavior of insects, terrestrial animals, birds and other flying animals, and fishes and other aquatic animals; their optimization search is inspired by mechanisms driving reproduction, food search, hunting, migration, etc. Hybrid/improved/enhanced methods attempt to minimize computation cost, improve robustness, and limit the number of internal parameters of metaheuristic formulations. Hybrid MHOAs may present parallel (component algorithms are independently run on parallel computers) or serial (algorithms are sequentially executed on the same machine) architectures. Comprehensive reviews of MHOAs have been presented in Refs. [1–7].

MHOAs are largely used in real-world problems, such as in the static/dynamic optimization of civil, aerospace, and mechanical systems and structures [8–34]; characterization of materials, and systems/structural identification [35–40]; and damage identification [41–46]. However, important issues still must be considered in metaheuristic optimization: (i) no algorithm is superior to its competitors in all problems; (ii) computational cost is, in most cases, significantly higher than for gradient-based optimizers; (iii) the setting of internal parameters and computational complexity of sophisticated optimizers are often difficult to handle.

While increasing computational power enabled the development of new MHOAs, it should be underlined that most of the new algorithms had a very limited impact in the optimization community. For this reason, optimization experts very often preferred to hybridize or modify the most efficient available MHOAs rather than developing new methods from scratch. In general, the development of hybrid formulations should rely on important principles such as (i) to complement the exploration and exploitation characteristics of each component algorithm, (ii) to simplify the algorithm, and (iii) to achieve good versatility. Following this rationale, Furio et al. [6,7] developed two efficient hybrid metaheuristic algorithms combining the grey wolf optimizer (GWO) and the JAYA method. GWO was developed by Mirjalili et al. [47] in 2014; GWO mimics the leadership hierarchy and hunting mechanism of grey wolves. GWO is the second most cited algorithm in the metaheuristic optimization literature after particle swarm optimization (PSO). Its simple formulation (much simpler than PSO and without internal parameters) and high computational efficiency explain the huge number of engineering applications documented in the literature for this method (e.g., [48–50]). JAYA, developed by Rao [51] in 2016, utilizes the most straightforward search scheme ever proposed in metaheuristic optimization; variables are perturbed using just one equation to approach the population's best solution and move away from the worst solution. JAYA's efficiency and versatility are confirmed by the many studies presented in the literature (e.g., [52,53]). The inherently hybrid nature of JAYA, combining the survival of the fittest individual and the leader's ability to guide the rest of the population to the optimal solution, makes JAYA an excellent component algorithm for the hybridization of metaheuristic methods.

The term “hybrid MHOAs” also covers the integration of gradient information into the metaheuristic search. In this regard, Ficarella et al. [26] developed efficient hybrid formulations of three widely used metaheuristic optimization algorithms, such as simulated annealing (SA), harmony search (HS), and big bang-big crunch (BBBC). In [26], low-cost gradient evaluation and approximate line search were added to each metaheuristic search engine to generate very high-quality trial designs, always belonging to descent directions. This allowed the continuous improvement of the design by quickly approaching the global optimum. The population was dynamically renewed by simultaneously improving as many candidate designs as possible. The hybridization scheme of [26] may be applied, in principle, to any metaheuristic optimizers.

Based on the previous arguments, the novel HALSGWJA (Hybrid Approximate Line Search Grey Wolf JAYA) algorithm was developed in the present study. HALSGWJA combines the basic formulations of GWO and JAYA, but it utilizes approximate gradient information and line search to modify trial designs, always making them very likely to improve the current best record. Since this strategy is common to exploration and exploitation, these two stages are well balanced over the whole optimization process, thus maximizing search efficiency. HALSGWJA definitely improves the hybrid GWO-JAYA formulations previously developed in [6,7], which already included elitist and repair strategies to generate high-quality candidate solutions but did not directly use line search in the initial generation of trial solutions, as this is instead performed by the present algorithm. This allows HALSGWJA to avoid the heuristic criteria for the acceptance/rejection of trial designs that were necessary in [6,7], thus achieving faster and more robust convergence behavior.

The HALSGWJA algorithm was tested on 20 real-world mechanical engineering optimization problems including (i) the 19 examples of the CEC2020 library on real-world mechanical engineering and (ii) a highly nonlinear optimal crashworthiness design problem. These 20 test problems are very significant as they include up to 30 optimization variables and 86 nonlinear constraints, as well as sizing/topology, static/dynamic optimization problems, and discrete variable problems. HALSGWJA was extensively compared

with more than 40 other metaheuristics. Remarkably, HALSGWJA always competed well with CEC2020's best performing algorithms and other advanced state-of-the-art metaheuristic optimizers, requiring fewer analyses to complete the optimization process.

The remainder of this article is organized as follows. Section 2 describes the HALSGWJA algorithm. Optimization results are discussed in Section 3. Section 4 summarizes this study and outlines future investigations.

2. The HALSGWJA Algorithm

The HALSGWJA algorithm developed in this study has six steps: (i) initialization; (ii) GWO phase, including approximate gradient information; (iii) JAYA phase, including evaluation and improvement of trial designs with elitist strategies; (iv) population resorting, definition of the new best/worst designs, and updating of α - β - δ wolves; (v) convergence check; (vi) end of search process and output of optimum design.

Step 1. Initialization

HALSGWJA randomly generates the population of N_{POP} candidate solutions (i.e., wolves) as

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

where NDV is the number of optimization variables, x_j^L and x_j^U are the lower and upper bounds of the j th variable, respectively, and ρ_j^i is a random number in the interval $(0,1)$.

The cost function $W(\vec{X})$ (depending on the NDV variables stored in the solution vector $\vec{X}$) is collapsed with the N_{CON} inequality constraint functions $G_k(\vec{X}) \leq 0$ into the penalized cost function $W_p(\vec{X})$:

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

where p is the penalty coefficient that may be constant or adaptively change in the optimization process. The penalty function ψ is

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

The penalized cost $W_p(\vec{X})$ and the cost function $W(\vec{X})$ coincide if the trial solution $\vec{X}$ satisfies optimization constraints. Equality constraints $H(\vec{X}) = 0$ are transformed into inequality constraints of the $G(\vec{X}) \leq 0$ and $-G_k(\vec{X}) \leq 0$ types. Candidate solutions are sorted with respect to penalized cost function values; the current best solution $\vec{X}_{best}$ and the worst solution $\vec{X}_{worst}$, respectively, correspond to the lowest (highest) and highest (lowest) values of ψ for minimization (maximization) problems.

Step 2. GWO phase: Generation of new trial designs with approximate gradient information

GWO simulates the leadership hierarchy and hunting mechanism of grey wolves [47]. GWO defines four hierarchical types of grey wolves: α , β , δ , and ω . The hunting phase is simulated by mathematical operators depicting searching, encircling, and attacking prey. The group is led by the α wolf. The β wolf helps the α wolf in decision-making or other pack activities. The δ wolves rank below the α and β wolves but above the lowest-ranked ω wolves. While α , β , and δ wolves estimate the prey's position, other wolves randomly update their positions around the prey.

The proposed HALSGWJA algorithm defines new trial designs by enhancing classical GWO with gradient information. The α , β , and δ wolves are the three best individuals

ranked with respect to penalized cost function. All other designs are classified as wolves ω . Wolves encircle the prey during the hunt. This behavior can be modeled as

$$\vec{D} = |\vec{C} \cdot \vec{X}_p| - \vec{X}_{it} \quad (4)$$

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

where it is the current iteration, $\vec{A}$ and $\vec{C}$ are coefficient vectors, $\vec{X}_p$ is the prey's position vector, and $\vec{X}_{it}$ is the generic search agent (i.e., grey wolf) of population and its updated position $\vec{X}_{it+1}$. The “ $\cdot$ ” is the term-by-term vector multiplication defining another vector.

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

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \quad (6)$$

$$\vec{C} = 2\vec{r}_2 \quad (7)$$

where the components of vector $\vec{a}$ decrease linearly from 2 to 0 in the optimization process; $\vec{r}_1$ and $\vec{r}_2$ are random vectors extracted from [0,1]. In the exploration phase, search agents tend to search for another prey (i.e., another optimum solution) for $\|\vec{A}\| > 1$. Conversely, search agents converge towards prey (i.e., optimum solution) in the exploration phase for $\|\vec{A}\| < 1$.

To simulate hunting, GWO assumes that the three best individuals (i.e., wolves α , β , and δ) can estimate the prey's location (i.e., optimal solution) better than the rest of the population. All other designs are hence updated using the design vectors $\vec{X}_\alpha$, $\vec{X}_\beta$, and $\vec{X}_\delta$ of the best individuals. The generic design $\vec{X}_i$ of population is updated to $\vec{X}_{i,tr}$ as follows:

$$\begin{cases} \vec{D}_\alpha = |\vec{C}_1 \cdot \vec{X}_\alpha| - \vec{X}_i \\ \vec{D}_\beta = |\vec{C}_2 \cdot \vec{X}_\beta| - \vec{X}_i \\ \vec{D}_\delta = |\vec{C}_3 \cdot \vec{X}_\delta| - \vec{X}_i \end{cases} \quad (8)$$

$$\begin{cases} \vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot \vec{D}_\alpha \\ \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot \vec{D}_\beta \\ \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot \vec{D}_\delta \end{cases} \quad (9)$$

$$(\vec{X}_{i,tr})^{prel} = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \quad (10)$$

Vectors $\vec{X}_1, \vec{X}_2$, and $\vec{X}_3$ vary for each candidate design $\vec{X}_i$. Equations (8)–(10) update the positions of all individuals.

Classical GWO sorts the new population of trial solutions $(\vec{X}_{i,tr})^{prel}$ to update positions $\vec{X}_\alpha$, $\vec{X}_\beta$, and $\vec{X}_\delta$ in each iteration. This process ends when the optimizer completes the limit number of iterations or function evaluations specified by the user. However, classical GWO has two inherent limitations; (i) the search is biased toward the α wolf, with the risk of stagnating search in the neighborhood of current best record $\vec{X}_{best}$, and (ii) each new trial solution $(\vec{X}_{i,tr})^{prel}$ does not necessarily improve $\vec{X}_{best}$ or at least the corresponding search agent $\vec{X}_i$ of the previous population. The present authors attempted to overcome these problems by introducing elitist strategies that refine the $(\vec{X}_{i,tr})^{prel}$ trial designs [6,7]. In particular, only the trial solutions that are very likely to improve design are retained in the updated population and are then further refined based on their rank

with respect to $\overrightarrow{X_{best}}$. However, the perturbation step given to $(\overrightarrow{X_{l, tr}})^{prel}$ could be too small to effectively improve $\overrightarrow{X_{best}}$ or at least $\overrightarrow{X_l}$.

In order to solve these issues, HALSGWJA utilizes gradient information to refine trial designs $(\overrightarrow{X_{l, tr}})^{prel}$. Ideally, gradients of cost function and constraint functions should be computed with respect to each trial design $\overrightarrow{X_l}$ currently perturbed. However, this is not affordable in terms of computational cost. Hence, HALSGWJA evaluates the rate of variation in penalized cost function with respect to $\overrightarrow{X_l}$. The largest improvement would obviously be obtained by perturbing the design from $\overrightarrow{X_l}$ to $\overrightarrow{X_{best}}$. However, this may not be the fastest improvement if one considers the $\Delta W_p(i,k)/\Delta S(i,k)$ ratio between the $\Delta W_p(i,k) = \|W_p(\overrightarrow{X_l}) - W_p(\overrightarrow{X_k})\|$ variation in penalized cost function and the corresponding $\Delta S(i,k) = \|\overrightarrow{X_l} - \overrightarrow{X_k}\|$ perturbation size from $\overrightarrow{X_l}$ to another search agent $\overrightarrow{X_k}$ better ranked in the population than $\overrightarrow{X_l}$. If $\overrightarrow{X_k}$ ranks better than $\overrightarrow{X_l}$, the corresponding direction $\overrightarrow{S_{l,k}} = (\overrightarrow{X_k} - \overrightarrow{X_l})$ is a descent direction with respect to $\overrightarrow{X_l}$. Conversely, if $\overrightarrow{X_k}$ ranks below $\overrightarrow{X_l}$, perturbing the design along the $(\overrightarrow{X_k} - \overrightarrow{X_l})$ direction would not improve the design with respect to $\overrightarrow{X_l}$: hence, it is better to perturb the design in the opposite direction $\overrightarrow{S_{l,k}} = -(\overrightarrow{X_k} - \overrightarrow{X_l})$ or, equivalently, along the direction $\overrightarrow{S_{l,k}} = (\overrightarrow{X_l} - \overrightarrow{X_k})$.

In view of this, HALSGWJA computes the average gradient of the penalized cost function with respect to $\overrightarrow{X_l}$ as $\gamma^{(i,k)} = \Delta W_p(i,k)/\Delta S(i,k)$ for each design $\overrightarrow{X_k}$ stored in the population. The preliminary trial design $(\overrightarrow{X_{l, tr}})^{prel}$, generated by the GWO scheme, is hence perturbed as follows:

$$\begin{aligned} \overrightarrow{X_{l, tr}} = & (\overrightarrow{X_{l, tr}})^{prel} + \tau_{best}(\overrightarrow{X_{best}} - \overrightarrow{X_l}) + \tau_{FST, better}(\overrightarrow{X_{FST, better}} - \overrightarrow{X_l}) - \\ & - \tau_{FST, worse}(\overrightarrow{X_{FST, worse}} - \overrightarrow{X_l}) \end{aligned} \quad (11)$$

In the above equation, $\overrightarrow{X_{FST, better}}$ is the candidate design of population ranking above $\overrightarrow{X_l}$, yet different from $\overrightarrow{X_{best}}$, that achieves the largest value of average gradient $\gamma^{(i,k)}$; $\overrightarrow{X_{FST, worse}}$ is the candidate design of population ranking below $\overrightarrow{X_l}$, that achieves the largest value of average gradient $\gamma^{(i,k)}$; and τ_{best} , $\tau_{FST, better}$, and $\tau_{FST, worse}$ are three random numbers in the interval [0,1]. It can be seen that Equation (11) refines the position of $(\overrightarrow{X_{l, tr}})^{prel}$ by combining three descent directions that may potentially improve the design with respect to $\overrightarrow{X_l}$. The $(\overrightarrow{X_{best}} - \overrightarrow{X_l})$ direction would lead to achieving the largest improvement for $\overrightarrow{X_l}$ as the search is directed towards the current best record stored in the population. The $(\overrightarrow{X_{FST, better}} - \overrightarrow{X_l})$ direction is the one that would lead to an improvement in $\overrightarrow{X_l}$ as quickly as possible, moving towards designs that rank above $\overrightarrow{X_l}$. The $-(\overrightarrow{X_{FST, worse}} - \overrightarrow{X_l})$ direction is the one that would lead to escape the most quickly from any design of the population ranking below $\overrightarrow{X_l}$. The perturbation steps taken along these three directions are weighted by the random numbers τ_{best} , $\tau_{FST, better}$, and $\tau_{FST, worse}$, respectively.

Step 3. Evaluation and correction of new trial designs: JAYA phase and elitist strategies

The trial designs $\overrightarrow{X_{l, tr}}$ must be evaluated to check whether they should be included in the new population. Classical GWO replaces the old design $\overrightarrow{X_l}$ with the new design $\overrightarrow{X_{l, tr}}$ if $W_p(\overrightarrow{X_{l, tr}}) < W_p(\overrightarrow{X_l})$ but this task entails a new constraint evaluation. The present authors introduced in [6,7] an elitist strategy where the cost function $W(\overrightarrow{X_{l, tr}})$ of the new trial design $\overrightarrow{X_{l, tr}}$ was compared with the penalized cost function $W_p(\overrightarrow{X_{best}})$ of the current best record $\overrightarrow{X_{best}}$. If $W(\overrightarrow{X_{l, tr}}) > W_p(\overrightarrow{X_{best}})$, the new trial design $\overrightarrow{X_{l, tr}}$ was directly rejected. Conversely, if $W(\overrightarrow{X_{l, tr}}) < W_p(\overrightarrow{X_{best}})$ and $W(\overrightarrow{X_{l, tr}}) \leq 1.1 W(\overrightarrow{X_{best}})$, the new trial solution $\overrightarrow{X_{l, tr}}$ was provisionally included in the new population. This step is no longer necessary in HALSGWJA because Equation (11) refines trial designs $\overrightarrow{X_{l, tr}}$, combining three significant directions that lead to a decrease in the cost function value at least with respect to the old

design $\overrightarrow{X}_i$. Hence, HALSGWJA includes in the new population Π_{tr} all $\overrightarrow{X}_{i,tr}$ trial designs defined with Equation (11). Unlike classical GWO and the improved GWO formulations developed in [6,7], HALSGWJA now defines the new trial solutions $\overrightarrow{X}_{i,tr}$ of the new population not only considering the positions of wolves α , β , and δ but also the effect of the rest of the population. This strategy reduces, by a great deal, the risk of stagnation for the new trial solutions.

HALSGWJA adopts a JAYA-based scheme to further improve trial designs $\overrightarrow{X}_{i,tr}$. The hybridization of GWO and JAYA was considered also in [6,7], using JAYA operators to exploit the trial designs $\overrightarrow{X}_{i,tr}$ or to globally explore the design space returning to perturb the old design $\overrightarrow{X}_i$ originally included in the population. The former was performed if $W(\overrightarrow{X}_{i,tr}) \leq 1.1 W(\overrightarrow{X}_{best})$, while the latter was performed if $W(\overrightarrow{X}_{i,tr}) > 1.1 W(\overrightarrow{X}_{best})$. The selection of the 1.1 threshold made in [6,7] resembles the probabilistic acceptance/rejection criterion of simulated annealing that allows local minima to be bypassed and becomes very useful in the exploitation phase. In the present study, HALSGWJA modified the above-mentioned criterion to be $W(\overrightarrow{X}_{i,tr}) \leq \rho_w W(\overrightarrow{X}_{best})$ or $W(\overrightarrow{X}_{i,tr}) > \rho_w W(\overrightarrow{X}_{best})$. The threshold value ρ_w , which replaces the 1.1 constant threshold value, is adaptively changed by HALSGWJA in each iteration based on the characteristics of the original population Π_{iter} of candidate designs $\overrightarrow{X}_i$ at the beginning of current iteration, the population of trial designs $\overrightarrow{X}_{i,tr}$ defined in the GWO phase with Equations (4)–(11), and the approximate gradients evaluated at the best designs of these two populations. This allows search agents to be adaptively perturbed in a significantly more dynamic way than in [6,7]. The new threshold value ρ_w defined by HALSGWJA is

$$\rho_w = \text{Min} \left\{ \frac{W_p(\overrightarrow{X}_\delta)}{W_p(\overrightarrow{X}_{best})}, \frac{W_{\delta,tr}}{W_{best,tr}}, \frac{\|\overline{\nabla} W_p(\overrightarrow{X}_{best})\|}{\|\overline{\nabla} W(\overrightarrow{X}_{best,tr})\|}, \frac{\|\overline{\nabla} W(\overrightarrow{X}_{best,tr})\|}{\|\overline{\nabla} W_p(\overrightarrow{X}_{best})\|} \right\} \quad (12)$$

In order to use Equation (12), HALSGWJA sorts the trial designs $\overrightarrow{X}_{i,tr}$ in terms of cost function. Let $W_{best,tr}$ and $W_{\delta,tr}$, respectively, be the best value and the third best value of cost function evaluated for the Π_{tr} population of trial designs $\overrightarrow{X}_{i,tr}$; the corresponding trial designs $\overrightarrow{X}_{best,tr}$ and $\overrightarrow{X}_{\delta,tr}$ practically represent the new wolves α_{tr} and δ_{tr} of Π_{tr} estimated on the basis of cost function values without evaluating constraints.

The approximate gradient of penalized cost function $\|\overline{\nabla} W_p(\overrightarrow{X}_{best})\|$ is evaluated at $\overrightarrow{X}_{best}$ for the original population $\Pi_{iter-initial}$ of candidate designs $\overrightarrow{X}_i$ at the beginning of the current iteration, considering wolves α (that coincides with $\overrightarrow{X}_{best}$), β , and δ , and the $\overrightarrow{X}_{FAST}$ design yielding the fastest variation in penalized cost function toward the current best record. For the Π_{tr} population of trial designs, the average gradient of cost function $\|\overline{\nabla} W(\overrightarrow{X}_{best,tr})\|$ is evaluated at $\overrightarrow{X}_{best,tr}$, considering wolves α_{tr} (that coincides with $\overrightarrow{X}_{best,tr}$), β_{tr} , and δ_{tr} and the $\overrightarrow{X}_{FAST,tr}$ trial design for which cost function decreases most quickly, moving toward $\overrightarrow{X}_{best,tr}$.

$$\begin{cases} \|\overline{\nabla} W_p(\overrightarrow{X}_{best})\| = \text{Mean} \left\{ \frac{W_p(\overrightarrow{X}_{best}) - W_p(\overrightarrow{X}_\beta)}{\|\overrightarrow{X}_{best} - \overrightarrow{X}_\beta\|}, \frac{W_p(\overrightarrow{X}_{best}) - W_p(\overrightarrow{X}_\delta)}{\|\overrightarrow{X}_{best} - \overrightarrow{X}_\delta\|}, \frac{W_p(\overrightarrow{X}_{best}) - W_p(\overrightarrow{X}_{FAST})}{\|\overrightarrow{X}_{best} - \overrightarrow{X}_{FAST}\|} \right\} \\ \|\overline{\nabla} W(\overrightarrow{X}_{best,tr})\| = \text{Mean} \left\{ \frac{W(\overrightarrow{X}_{best,tr}) - W(\overrightarrow{X}_{\beta,tr})}{\|\overrightarrow{X}_{best,tr} - \overrightarrow{X}_{\beta,tr}\|}, \frac{W(\overrightarrow{X}_{best,tr}) - W(\overrightarrow{X}_{\delta,tr})}{\|\overrightarrow{X}_{best,tr} - \overrightarrow{X}_{\delta,tr}\|}, \frac{W(\overrightarrow{X}_{best,tr}) - W(\overrightarrow{X}_{FAST,tr})}{\|\overrightarrow{X}_{best,tr} - \overrightarrow{X}_{FAST,tr}\|} \right\} \end{cases} \quad (13)$$

It can be seen from Equation (13) that HALSGWJA computes approximate gradients by averaging the relative variations in the penalized cost function W_p and the cost function W with respect to design perturbations taken along three representative descent directions. The computational cost of these operations is very low because penalty function values are available for all search agents at the beginning of each iteration while the eval-

uation of cost function for trial designs is usually much cheaper than evaluating constraints, especially for the CEC 2020 mechanical engineering test suite solved in this study, which includes explicitly stated test problems.

The rationale behind Equation (12) as follows. Basically, HALSGWJA checks if the Π_{tr} population of trial designs has a higher quality than the original population $\Pi_{iter-initial}$ that should be updated in the current iteration. Such a scenario is likely to occur considering that trial designs were defined in the GWO phase as trying to perturb optimization variables along descent directions. However, HALSGWJA adopts a conservative approach in order to minimize the probability of exploiting, in the JAYA phase, some design that is significantly worse than the three best search agents of Π_{tr} and $\Pi_{iter-initial}$ and that hence cannot effectively contribute to searching the optimum. For that purpose, the ρ_w threshold takes the minimum ratio between the responses of the δ and α wolves (i.e., the best records of the two populations) included in the two populations and further penalizes this ratio by considering the minimum ratio between approximate gradients. If the Π_{tr} population of trial designs tends to the target optimum, the gradient of cost function should decrease and finally approach zero, passing from $\Pi_{iter-initial}$ to Π_{tr} .

In the JAYA phase, HALSGWJA exploits the trial design $\overrightarrow{X_{l,tr}}$ if it occurs that $W(\overrightarrow{X_{l,tr}}) \leq \rho_w W(\overrightarrow{X_{best}})$. Since the JAYA search scheme consists of approaching the best design and escaping from the worst design of the population, it is necessary to define these two individuals. For that purpose, HALSGWJA evaluates the value of the penalized cost function $W_p(\overrightarrow{X_{best,tr}})$ at $\overrightarrow{X_{best,tr}}$ and compares it with its counterpart $W_p(\overrightarrow{X_{best}})$, evaluated at the best record $\overrightarrow{X_{best}}$ obtained so far. The $\overrightarrow{X_{best,JAYA}}$ design is hence defined as follows:

$$\begin{cases} W_p(\overrightarrow{X_{best,tr}}) \leq W_p(\overrightarrow{X_{best}}) \Rightarrow \overrightarrow{X_{best,JAYA}} = \overrightarrow{X_{best,tr}} \\ W_p(\overrightarrow{X_{best,tr}}) > W_p(\overrightarrow{X_{best}}) \Rightarrow \overrightarrow{X_{best,JAYA}} = \overrightarrow{X_{best}} \end{cases} \quad (14)$$

The $\overrightarrow{X_{worst,JAYA}}$ design is instead defined as follows:

$$\begin{cases} W(\overrightarrow{X_{\delta,tr}}) \leq W(\overrightarrow{X_{\delta}}) \Rightarrow \overrightarrow{X_{worst,JAYA}} = \overrightarrow{X_{\delta,tr}} \\ W(\overrightarrow{X_{\delta,tr}}) > W(\overrightarrow{X_{\delta}}) \Rightarrow \overrightarrow{X_{worst,JAYA}} = \overrightarrow{X_{\delta}} \end{cases} \quad (15)$$

The rationale of Equations (14) and (15) is to combine the best characteristics of the original population $\Pi_{iter-initial}$ and the trial population Π_{tr} that should replace it. This allows the efficiency of the exploitation phase to be maximized because HALSGWJA is forced to locally search in a region of design space containing only high-quality solutions such as the best α and δ wolves of the two populations. Remarkably, HALSGWJA may either select both $\overrightarrow{X_{best,JAYA}}$ and $\overrightarrow{X_{worst,JAYA}}$ from Π_{tr} or from Π_{iter} , as well as one from one population and the other from another population. The new trial solution $(\overrightarrow{X_{l,tr}})'$ is hence defined as:

$$(\overrightarrow{X_{l,tr}})' = \overrightarrow{X_{l,tr}} + \overrightarrow{\lambda_1} \cdot (\overrightarrow{X_{best,JAYA}} - \overrightarrow{X_{l,tr}}) - \overrightarrow{\lambda_2} \cdot (\overrightarrow{X_{worst,JAYA}} - \overrightarrow{X_{l,tr}}) \quad (16)$$

where $\overrightarrow{\lambda_1}$ and $\overrightarrow{\lambda_2}$ are two vectors of NDV random numbers in the [0,1] interval.

The cost function $W((\overrightarrow{X_{l,tr}})')$ is evaluated for the new trial solution $(\overrightarrow{X_{l,tr}})'$. Then HALSGWJA includes the $(\overrightarrow{X_{l,tr}})'$ design in the final population $\Pi_{iter-final}$, defined at the end of the current iteration if it occurs that $W((\overrightarrow{X_{l,tr}})') < W(\overrightarrow{X_{l,tr}})$; hence, the exploiting strategy set by HALSGWJA for $\overrightarrow{X_{l,tr}}$ is rated successful. Conversely, if $W((\overrightarrow{X_{l,tr}})') > W(\overrightarrow{X_{l,tr}})$, $(\overrightarrow{X_{l,tr}})'$ is rejected and $\overrightarrow{X_{l,tr}}$ is finally retained in the updated population $\Pi_{iter-final}$, HALSGWJA rates only the improvement in design achieved by $\overrightarrow{X_{l,tr}}$ as being satisfactory. HALSGWJA performs an exploration in the JAYA phase if the trial design $\overrightarrow{X_{l,tr}}$ yields $W(\overrightarrow{X_{l,tr}}) > \rho_w W(\overrightarrow{X_{best}})$. While $\overrightarrow{X_{l,tr}}$ is not good for exploitation, it may still be better than most of the designs included in the original population $\Pi_{iter-initial}$ as well as being highly ranked in the trial population Π_{tr} . Hence, it is necessary to explore over all of the $2 \times N_{POP}$

search agents included in $\Pi_{\text{iter-initial}}$ and Π_{tr} populations before deciding on whether to dismiss the trial design $\overrightarrow{X_{l, \text{tr}}}$. Again, HALSGWJA defines the $\overrightarrow{X_{\text{best}, \text{JAYA}}}$ and $\overrightarrow{X_{\text{worst}, \text{JAYA}}}$ designs required in the JAYA phase by combining the properties of $\Pi_{\text{iter-initial}}$ and Π_{tr} . Similarly to the exploitation phase, HALSGWJA compares the value of penalized cost function $W_p(\overrightarrow{X_{\text{best}, \text{tr}}})$ with its counterpart $W_p(\overrightarrow{X_{\text{best}}})$, and the $\overrightarrow{X_{\text{best}, \text{JAYA}}}$ design is defined by Equation (14). The $\overrightarrow{X_{\text{worst}, \text{JAYA}}}$ design is instead defined by comparing the worst designs of the two populations:

$$\begin{cases} W(\overrightarrow{X_{\text{worst}, \text{tr}}}) \leq W(\overrightarrow{X_{\text{worst}}}) \Rightarrow \overrightarrow{X_{\text{worst}, \text{JAYA}}} = \overrightarrow{X_{\text{worst}, \text{tr}}} \\ W(\overrightarrow{X_{\text{worst}, \text{tr}}}) > W(\overrightarrow{X_{\text{worst}}}) \Rightarrow \overrightarrow{X_{\text{worst}, \text{JAYA}}} = \overrightarrow{X_{\text{worst}}} \end{cases} \quad (17)$$

The new trial solution $(\overrightarrow{X_{l, \text{tr}}})'$ defined by HALSGWJA in the JAYA-based exploration phase is

$$(\overrightarrow{X_{l, \text{tr}}})' = \overrightarrow{X_{l, \text{JAYA}}} + \vec{\lambda}_1 \cdot (\overrightarrow{X_{\text{best}, \text{JAYA}}} - \overrightarrow{X_{l, \text{JAYA}}}) - \vec{\lambda}_2 \cdot (\overrightarrow{X_{\text{worst}, \text{JAYA}}} - \overrightarrow{X_{l, \text{JAYA}}}) \quad (18)$$

where random vectors $\vec{\lambda}_1$ and $\vec{\lambda}_2$ are similar to those used in Equation (16). The absolute values of the $\overrightarrow{X_l}$ or $\overrightarrow{X_{l, \text{tr}}}$ vectors' components included in Equations (16) and (17) are considered. The $\overrightarrow{X_{l, \text{JAYA}}}$ design included in Equation (18) is defined as follows:

$$\begin{cases} W(\overrightarrow{X_{l, \text{tr}}}) \leq W(\overrightarrow{X_l}) \Rightarrow \overrightarrow{X_{l, \text{JAYA}}} = \overrightarrow{X_{l, \text{tr}}} \\ W(\overrightarrow{X_{l, \text{tr}}}) > W(\overrightarrow{X_l}) \Rightarrow \overrightarrow{X_{l, \text{JAYA}}} = \overrightarrow{X_l} \end{cases} \quad (19)$$

Similarly to $\overrightarrow{X_{\text{best}, \text{JAYA}}}$ and $\overrightarrow{X_{\text{worst}, \text{JAYA}}}$, the $\overrightarrow{X_{l, \text{JAYA}}}$ design can either belong to $\Pi_{\text{iter-initial}}$ or Π_{tr} . The rationale of Equations (18) and (19) is as follows. Ideally, each individual included in the new population $\Pi_{\text{iter-final}}$ that has been updated in the current iteration should be better than its counterpart included in the original population $\Pi_{\text{iter-initial}}$. While the trial design $\overrightarrow{X_{l, \text{tr}}}$ may not be good for exploitation because it does not have enough quality for improving the current best record, HALSGWJA attempts to explore the search space to improve at least the i -th individual of the population. For this reason, HALSGWJA perturbs the best design $\overrightarrow{X_{l, \text{JAYA}}}$ amongst $\overrightarrow{X_l}$ and $\overrightarrow{X_{l, \text{tr}}}$ with the JAYA-based scheme of Equation (18); the algorithm searches on the $(\overrightarrow{X_{\text{best}, \text{JAYA}}} - \overrightarrow{X_{l, \text{JAYA}}})$ descent direction that leads to a better design than $\overrightarrow{X_{l, \text{JAYA}}}$ and escape from the worst design $\overrightarrow{X_{\text{worst}, \text{JAYA}}}$, which certainly cannot improve (or it is very unlikely to improve) $\overrightarrow{X_{l, \text{JAYA}}}$.

Again, the new trial solution $(\overrightarrow{X_{l, \text{tr}}})'$ is included in the final population $\Pi_{\text{iter-final}}$ when $W((\overrightarrow{X_{l, \text{tr}}})') < \rho_w W(\overrightarrow{X_{\text{best}}})$, which obviously yields $W((\overrightarrow{X_{l, \text{tr}}})') < \rho_w W(\overrightarrow{X_{\text{best}}}) < W(\overrightarrow{X_{l, \text{tr}}})$ as the last inequality leads to the exploration phase being carried out.

When the JAYA-based exploration phase fails, still yielding $W((\overrightarrow{X_{l, \text{tr}}})') > \rho_w W(\overrightarrow{X_{\text{best}}})$ besides $W(\overrightarrow{X_{l, \text{tr}}}) > \rho_w W(\overrightarrow{X_{\text{best}}})$, HALSGWJA utilizes a mirroring strategy for $\overrightarrow{X_{l, \text{tr}}}$ and $(\overrightarrow{X_{l, \text{tr}}})'$. The new trial solution $(\overrightarrow{X_{l, \text{tr}}})''^{(\text{mirr})}$ is defined as follows:

$$\begin{cases} W(\overrightarrow{X_{l, \text{tr}}})' \leq W(\overrightarrow{X_{l, \text{tr}}}) \Rightarrow (\overrightarrow{X_{l, \text{tr}}})''^{(\text{mirr})} = \overrightarrow{X_{\text{best}}} + \eta_{\text{mirr}}(\overrightarrow{X_{\text{best}}} - (\overrightarrow{X_{l, \text{tr}}})') \\ W(\overrightarrow{X_{l, \text{tr}}})' > W(\overrightarrow{X_{l, \text{tr}}}) \Rightarrow (\overrightarrow{X_{l, \text{tr}}})''^{(\text{mirr})} = \overrightarrow{X_{\text{best}}} + \eta_{\text{mirr}}(\overrightarrow{X_{\text{best}}} - \overrightarrow{X_{l, \text{tr}}}) \end{cases} \quad (20)$$

where η_{mirr} is a random number in the [0,1] interval. Basically, HALSGWJA perturbs the design by moving from the current best record along a descent direction with respect to the best design amongst $\overrightarrow{X_{l, \text{tr}}}$ and $(\overrightarrow{X_{l, \text{tr}}})'$. Hence, the search direction is mirrored with respect to the non-descent directions $(\overrightarrow{X_{l, \text{tr}}})' - \overrightarrow{X_{\text{best}}}$ and $\overrightarrow{X_{l, \text{tr}}} - \overrightarrow{X_{\text{best}}}$ that did not improve design. In this way, the trial solution $(\overrightarrow{X_{l, \text{tr}}})''^{(\text{mirr})}$ is very likely to improve design. The random step size set by η_{mirr} serves to limit the risk of generating infeasible designs. The new solution $(\overrightarrow{X_{l, \text{tr}}})''^{(\text{mirr})}$ is evaluated following the same steps described above for all other trial solutions generated by HALSGWJA.

Step 4. Resort population, update α , β , and δ wolves, $\overrightarrow{X_{best}}$, and $\overrightarrow{X_{worst}}$

The N_{POP} updated individuals are sorted in terms of the penalty function. The best three individuals of the new population are taken as α , β , and δ wolves, with their corresponding vectors of optimization variables $\overrightarrow{X_\alpha}$, $\overrightarrow{X_\beta}$, and $\overrightarrow{X_\delta}$. The best and worst search agents are taken as $\overrightarrow{X_{best}}$ and $\overrightarrow{X_{worst}}$, respectively.

To avoid search stagnation, HALSGWJA checks if wolves α , β , and δ were updated in the current iteration. This task is accomplished by an elitist criterion based on the concept of descent direction. The $\overrightarrow{S_\beta} = (\overrightarrow{X_{best}} - \overrightarrow{X_\beta})$ and $\overrightarrow{S_\delta} = (\overrightarrow{X_{best}} - \overrightarrow{X_\delta})$ directions are descent with respect to positions $\overrightarrow{X_\beta}$ and $\overrightarrow{X_\delta}$ of β and δ wolves as they lead to the improvement of the design, moving towards the current best record, represented by the α wolf. Similarly to [7], HALSGWJA perturbs β and δ wolves by moving along $\overrightarrow{S_\beta}$ and $\overrightarrow{S_\delta}$. The positions of wolves β and δ are displaced from the center of mass of the three best individuals of the population and then “mirrored” with respect to $\overrightarrow{X_{best}}$ as follows:

$$\begin{cases} (\overrightarrow{X_\beta})^{mirr} = \left(\frac{\overrightarrow{X_{best}} + \overrightarrow{X_\beta} + \overrightarrow{X_\delta}}{3} - \overrightarrow{X_\delta} \right) + (1 + \eta_{mirr,\beta})\overrightarrow{X_{best}} - \eta_{mirr,\beta}\overrightarrow{X_\beta} \\ (\overrightarrow{X_\delta})^{mirr} = \left(\frac{\overrightarrow{X_{best}} + \overrightarrow{X_\beta} + \overrightarrow{X_\delta}}{3} - \overrightarrow{X_\beta} \right) + (1 + \eta_{mirr,\delta})\overrightarrow{X_{best}} - \eta_{mirr,\delta}\overrightarrow{X_\delta} \end{cases} \quad (21)$$

where the random numbers $\eta_{mirr,\beta}$ and $\eta_{mirr,\delta}$ are extracted from the interval (0,1); they limit step sizes on $\overrightarrow{S_\beta}$ and $\overrightarrow{S_\delta}$ so as to minimize the risk of generating infeasible solutions. Interestingly, since the original perturbation scheme of Ref. [7] did not account for the relative position of the center of mass of the three best individuals and the δ wolf, the following “limit” cases could occur in [7]: (i) for $\eta_{mirr,\beta} = \eta_{mirr,\delta} = 0$, $(\overrightarrow{X_\beta})^{mirr}$ and $(\overrightarrow{X_\delta})^{mirr}$ coincided with $\overrightarrow{X_{best}}$; (ii) for $\eta_{mirr,\beta} = \eta_{mirr,\delta} = 1$, $(\overrightarrow{X_\beta})^{mirr}$ and $(\overrightarrow{X_\delta})^{mirr}$, respectively, were the symmetric of $\overrightarrow{X_\beta}$ and $\overrightarrow{X_\delta}$ about $\overrightarrow{X_{best}}$. Conversely, HALSGWJA includes in Equation (21) the $\left(\frac{\overrightarrow{X_{best}} + \overrightarrow{X_\beta} + \overrightarrow{X_\delta}}{3} - \overrightarrow{X_\delta} \right)$ term that forces the optimizer to define wolves that are always at least better than the δ wolf.

The best three search agents amongst $\overrightarrow{X_\alpha}$, $\overrightarrow{X_\beta}$, $\overrightarrow{X_\delta}$, $(\overrightarrow{X_\beta})^{mirr}$ and $(\overrightarrow{X_\delta})^{mirr}$ are stored by HALSGWJA as the new α , β , and δ -wolves for the next iteration. The two worst search agents are compared by HALSGWJA with the rest of the population and may replace $\overrightarrow{X_{worst}}$ and the second-worst design of the population. This strategy also accounts for $(\overrightarrow{X_\beta})^{mirr}$ and $(\overrightarrow{X_\delta})^{mirr}$ not improving any of the α , β , and δ wolves. By replacing the two worst designs of population, it is possible to improve the average quality of search agents and finally achieve a higher probability of generating high-quality trial designs in the subsequent iteration.

Step 5. Convergence check

Standard deviations on optimization variables and penalty function values of candidate designs have to decrease as the search process progresses towards the global optimum. For this reason, HALSGWJA normalizes the standard deviation of the design vectors with respect to the average design vector $\overrightarrow{X_{aver}} = (\sum_{i=1}^{N_{POP}} \overrightarrow{X_i}) / N_{POP}$ of the population and the standard deviation of the penalty function values with respect to the average penalty function value $W_{p,aver} = (\sum_{i=1}^{N_{POP}} W_p(\overrightarrow{X_i})) / N_{POP}$. The termination criterion used by HALSGWJA is

$$\text{Max} \left\{ \frac{\text{STD}\{\|\vec{X}_1 - \vec{X}_{aver}\|, \|\vec{X}_2 - \vec{X}_{aver}\|, \dots, \|\vec{X}_{NPOP} - \vec{X}_{aver}\|\}}{\|\vec{X}_{aver}\|}, \frac{\text{STD}\{|W_{p,1} - W_{p,aver}|, |W_{p,2} - W_{p,aver}|, \dots, |W_{p,NPOP} - W_{p,aver}|\}}{W_{p,aver}} \right\} \leq \varepsilon_{conv} \quad (22)$$

The convergence limit ε_{conv} is 10^{-7} . Penalty function and cost function values coincide in feasible solutions or in the case of unconstrained optimization problems.

Steps 2 through 5 are repeated until HALSGWJA converges to the global optimum.

It should be noted that the convergence criterion of Equation (22) utilized by HALSGWJA is different from the typical stopping condition usually adopted by metaheuristic optimizers that terminate search when the limit number of iterations specified by the user is completed. HALSGWJA directly drives all candidate designs towards the optimum solution by performing multiple line searches (inherently based on elitist strategies) that generate high-quality designs very likely to improve the current best record or at least a very large fraction of the current population. This makes it no longer necessary to perform many optimization iterations, and the computational cost of optimization process is significantly reduced.

Step 6. *Terminate optimization process*

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

Algorithm 1 presents the pseudo-code of the HALSGWJA algorithm developed in this study, while Figure 1 shows the algorithm flow chart.

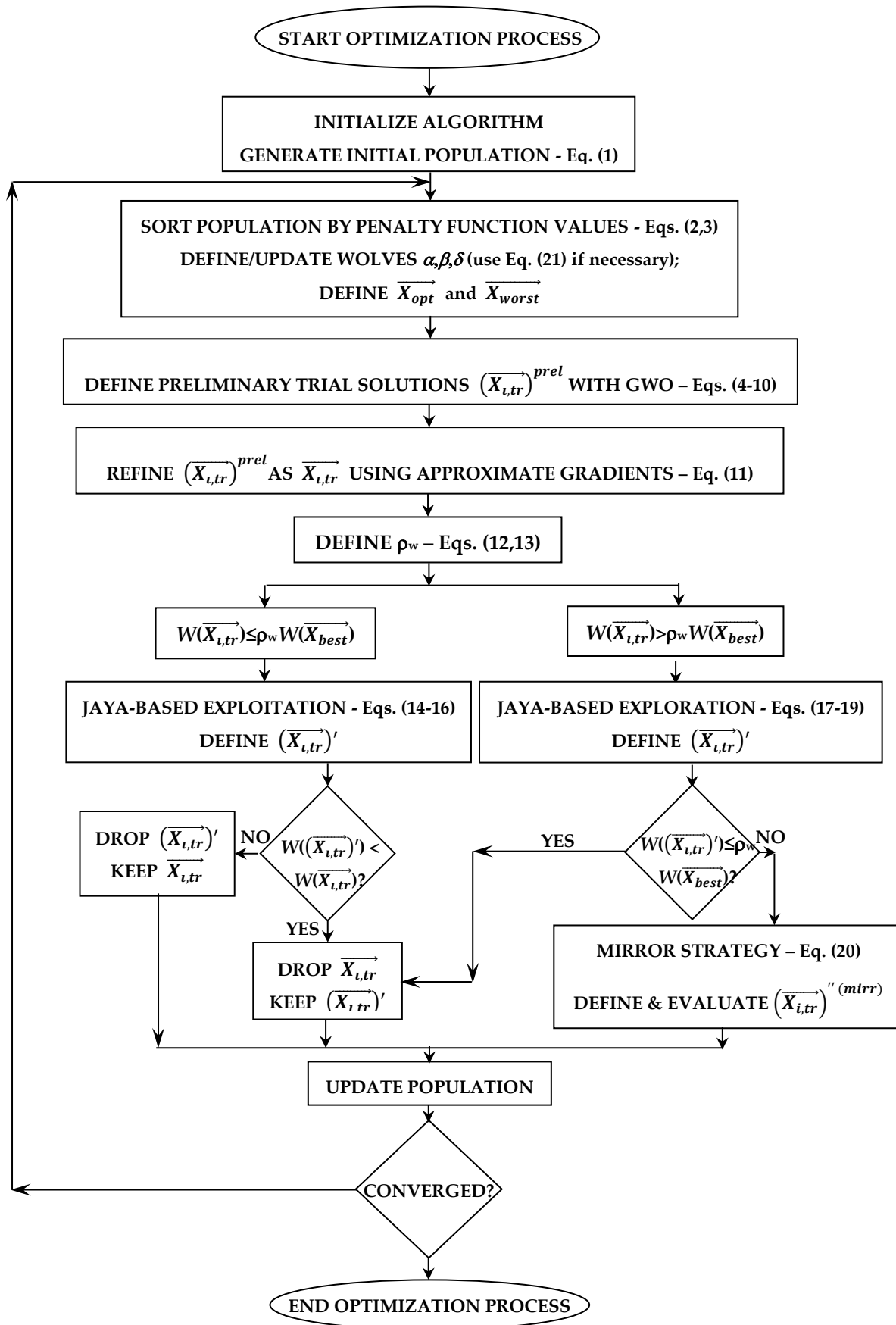


Figure 1. Flowchart of the HALSGWJA algorithm developed in this research.

Algorithm 1 Pseudo-code of the HALSGWJA algorithm**START HALSGWJA**

1. Step 1. Initialization
2. Set population size N_{POP} and limit number of iterations $N_{itermax}$.
3. Set iteration number as $ITER = 1$.
4. Randomly generate the population of N_{POP} candidate designs $\vec{X}_i$ ($i = 1, \dots, N_{POP}$) with Equation (1).
5. Denominate the population as $\Pi_{iter-initial}$ that should be updated in the current iteration.
6. **For** $i = 1, \dots, N_{POP}$
7. Evaluate cost function $W(\vec{X}_i)$ and constraint functions $G(\vec{X}_i), H(\vec{X}_i)$ for each $\vec{X}_i$.
8. Evaluate penalized cost $W_p(\vec{X}_i)$ for each candidate design $\vec{X}_i$ with Equations (2) and (3).
9. **end for**
10. Rank candidate designs $\vec{X}_i$ included in the population by sorting $W_p(\vec{X}_i)$ values from lowest to highest. Define $\vec{X}_{opt}$ and $\vec{X}_{worst}$, respectively, as the best and worst candidate designs of population yielding the minimum and maximum penalized cost values.
11. Set the best three individuals with the lowest values of W_p as α, β and δ wolves: let $\vec{X}_\alpha, \vec{X}_\beta$ and $\vec{X}_\delta$ be the corresponding design vectors. The α wolf is the current best record $\vec{X}_{opt} \equiv \vec{X}_\alpha$ with the corresponding penalized cost $W_{p,opt} = W_p(\vec{X}_\alpha)$.
12. **For** $i = 1, \dots, N_{POP}$
13. Step 2-A. Update individuals $\vec{X}_i$ to preliminary trial designs $(\vec{X}_{i,tr})^{prel}$ with classical GWO equations (4–10)
14. Step 2-B. Refine preliminary trial designs $(\vec{X}_{i,tr})^{prel}$ as $\vec{X}_{i,tr}$ using approximate gradients
15. **end for**
16. Store trial designs $\vec{X}_{i,tr}$ in the Π_{tr} population that should include better designs than the original population.
17. **For** $i = 1, \dots, N_{POP}$
18. Compute the variation of penalized cost function $\Delta W_p^{(i,k)} = \|W_p(\vec{X}_i) - W_p(\vec{X}_k)\|$ and the corresponding step size $\Delta S^{(i,k)} = \|\vec{X}_i - \vec{X}_k\|$ if design was perturbed from $\vec{X}_i$ to $\vec{X}_k$.
19. Determine average gradient of penalized cost function $\gamma^{(i,k)} = \Delta W_p^{(i,k)} / \Delta S^{(i,k)}$ for each $\vec{X}_i$.
20. Find candidate designs $\vec{X}_{FST,better}$ and $\vec{X}_{FST,worse}$, respectively, ranking above and below $\vec{X}_i$, and yielding the largest values of $\gamma^{(i,k)}$.
21. Define trial designs $\vec{X}_{i,tr}$ with Equation (11).
22. **end for**
23. Define threshold value Q_w with Equations (12) and (13).
24. Step 3. Evaluate trial designs $\vec{X}_{i,tr}$ and exploit them or re-explore search space with JAYA schemes
25. **For** $i = 1, \dots, N_{POP}$
26. **If** $W(\vec{X}_{i,tr}) \leq Q_w W(\vec{X}_{opt})$
27. Define $\vec{X}_{best,JAYA}$ and $\vec{X}_{worst,JAYA}$ for $\vec{X}_{i,tr}$ with Equation (14) and Equation (15), respectively.
28. Exploit trial design $\vec{X}_{i,tr}$ and update it into $(\vec{X}_{i,tr})'$ with the JAYA scheme of Equation (16).
29. **If** $W((\vec{X}_{i,tr})') < W(\vec{X}_{i,tr})$
30. Drop $\vec{X}_{i,tr}$ and include $(\vec{X}_{i,tr})'$ in the updated population $\Pi_{iter-final}$ for current iteration.
31. **else**
32. Drop $(\vec{X}_{i,tr})'$ and include $\vec{X}_{i,tr}$ in the updated population $\Pi_{iter-final}$ for current iteration.
33. **end if**
34. **end if**
35. **If** $W(\vec{X}_{i,tr}) > Q_w W(\vec{X}_{opt})$
36. Define $\vec{X}_{best,JAYA}$ and $\vec{X}_{worst,JAYA}$ for $\vec{X}_{i,tr}$ with Equation (14) and Equation (17), respectively.

-
37. Explore search space combining populations $\Pi_{\text{iter-initial}}$ and Π_{iter} . Define $\overrightarrow{X_{i,JAYA}}$ with Equation (19). Update $\overrightarrow{X_{i,tr}}$ into $(\overrightarrow{X_{i,tr}})'$ with the JAYA scheme of Equation (18).
 38. **If** $W((\overrightarrow{X_{i,tr}})') \leq \rho_w W(\overrightarrow{X_{opt}})$
 39. Drop $\overrightarrow{X_{i,tr}}$ and include $(\overrightarrow{X_{i,tr}})'$ in the updated population $\Pi_{\text{iter-final}}$ for current iteration.
 40. **end if**
 41. **If** $W((\overrightarrow{X_{i,tr}})') > \rho_w W(\overrightarrow{X_{opt}})$
 42. Define the new trial design $(\overrightarrow{X_{i,tr}})''^{(mirr)}$ with the mirroring strategy of Equation (20).
 43. Evaluate $(\overrightarrow{X_{i,tr}})''^{(mirr)}$ following the same steps described for the other trial designs to finally update population $\Pi_{\text{iter-final}}$.
 44. **end if**
 45. **end if**
 46. **end for**
 47. Step 4. Resort population, define new wolves α , β and δ , update $\overrightarrow{X_{opt}}$ and $\overrightarrow{X_{worst}}$
 48. Sort the updated population $\Pi_{\text{iter-final}}$ with respect to penalized cost function values W_p .
 49. Update positions $\overrightarrow{X_\alpha}$, $\overrightarrow{X_\beta}$ and $\overrightarrow{X_\delta}$ of wolves α , β and δ , respectively.
 50. Use elitist mirror strategy of Equation (21) to avoid stagnation of wolves α , β and δ . The optimizer is forced to define wolves that always are at least better than the δ -wolf.
 51. Update, if necessary, $\overrightarrow{X_\alpha}$, $\overrightarrow{X_\beta}$ and $\overrightarrow{X_\delta}$ or at least the worst two designs of population $\Pi_{\text{iter-final}}$.
 52. Set the α -wolf as the current best record $\overrightarrow{X_{opt}} \equiv \overrightarrow{X_\alpha}$ with penalized cost $W_{p,opt} = W_p(\overrightarrow{X_\alpha})$.
 53. Step 5. Check for convergence
 54. **If** the convergence criterion stated by Equation (22) is satisfied
 55. Go to line 61 to terminate optimization process.
 56. **else**
 57. Continue the optimization process.
 58. Update iteration number as $ITER = ITER + 1$.
 59. Go to line 5.
 60. **end if**
 61. Step 6. Terminate optimization process and output optimal solution
 62. Terminate optimization process.
 63. Output optimal design $\overrightarrow{X_{opt}}$ and optimal cost value $W(\overrightarrow{X_{opt}})$.
- END HALSGWJA.**
-

HALSGWJA actually represents an advanced grey wolf optimization algorithm. In particular, HALSGWJA updates the population, trying to improve the current best record or at least the largest part of the current population as possible in each iteration. Adding the use of approximate gradient information into the HALSGWJA formulation facilitates the optimization search process to a great extent. The JAYA schemes and the other elitist strategies included in HALSGWJA optimize the sequence of exploration and exploitation phases. Consequently, the proposed algorithm may increase diversity and it always selects high-quality trial designs, yet performing a limited number of analyses. HALSGWJA presents a master–slave architecture where the slave algorithm JAYA refines/corrects trial designs generated by the master algorithm GWO.

Similarly to the simple hybrid GWO/JAYA algorithms developed by the present authors in [6,7], HALSGWJA does not need any new internal parameters with respect to its component algorithms GWO and JAYA. Hence, population size N_{POP} and limit number of iterations $N_{itermax}$ remain the only hyper-parameters to be defined by the user. Interestingly, the 1.1 heuristic threshold used in [6,7] to accept/reject trial designs following the $W(\overrightarrow{X_{i,tr}}) \leq 1.1W(\overrightarrow{X_{best}})$ criterion is replaced in HALSGWJA by the more robust, flexible, and adaptive criterion $W(\overrightarrow{X_{i,tr}}) \leq \rho_w W(\overrightarrow{X_{best}})$. The inherent ability of HALSGWJA to generate high-

quality trial designs in each phase of the optimization process makes it possible to converge to the optimum design well before completing the $N_{itermax}$ iterations.

While most GWO/JAYA implementations require $N_{POP} \times N_{itermax}$ function/constraint evaluations (i.e., structural analyses for structural optimization problems), HALSGWJA defines all descent directions utilized in the search process without any new constraint evaluations. The elitist strategies and the approximate gradient information implemented in HALSGWJA allow the reliable evaluation of the quality of all trial designs based only on cost function values. Interestingly, approximate gradient information is obtained at the beginning of each iteration by evaluating the rate of change in the penalized cost function with respect to the distance of each search agent from the current best record. Since penalized cost function values are available for all candidate designs once the new population has been formed, the gradient evaluation performed by HALSGWJA does not entail any additional analysis.

It will be shown in Section 3 that the HALSGWJA's capacity to generate high-quality trial designs over the whole search process significantly reduces the computational cost of the optimization. The present algorithm does not need to work with many search agents because it can always generate high-quality designs that improve the current best record or at least many individuals of the population. Grey wolves hunt in nature in groups of 10–20 individuals (typical family pack includes 5–11 animals but it may also be formed by 2–3 families). For this reason, HALSGWJA utilized a population of 20 individuals. The validity of this setting was confirmed by sensitivity analysis.

The computational (i.e., time) complexity of the proposed HALSHGWJA algorithm results from the superposition of the complexities of different algorithmic phases: (i) initialization of candidate designs and population sorting; (ii) generation of preliminary trial designs $(\overrightarrow{X_{l, tr}})^{prel}$ with classical GWO; (iii) refinement of preliminary trial designs $(\overrightarrow{X_{l, tr}})^{prel}$ into $\overrightarrow{X_{l, tr}}$ using approximate gradient information; (iv) evaluation of trial designs $\overrightarrow{X_{l, tr}}$ and their modification into $(\overrightarrow{X_{l, tr}})'$ or $(\overrightarrow{X_{l, tr}})''$ with JAYA-based exploitation/exploration schemes or mirroring strategy; (v) preparation for the new iteration. The time complexity of each phase can be determined as follows:

- (i) The generation and sorting of initial population entail $(N_{POP} \times NDV)$ and $(N_{POP} \times \log N_{POP})$ operations, respectively; hence, the computational complexity of initialization phase is $O(N_{POP} \times (NDV + \log N_{POP}))$.
- (ii) The generation of preliminary trial designs $(\overrightarrow{X_{l, tr}})^{prel}$ with classical GWO entails $O(N_{POP} \times NDV)$ operations in each optimization iteration.
- (iii) The refinement of preliminary trial designs $(\overrightarrow{X_{l, tr}})^{prel}$ into trial designs $\overrightarrow{X_{l, tr}}$ involves the computation of approximate gradients $\gamma^{(i, k)}$; hence, $2 \times \binom{N_{POP}}{2}$ operations to determine $\Delta W_p^{(i, k)}$ and $\Delta S^{(i, k)}$, and N_{POP} operations to compute $\Delta W_p^{(i, k)} / \Delta S^{(i, k)}$. The computational complexity of this phase for each optimization iteration hence is $O(N_{POP} + 2 \times \binom{N_{POP}}{2})$.
- (iv) The evaluation of trial designs $\overrightarrow{X_{l, tr}}$ entails the verification of the elitist criterion $W(\overrightarrow{X_{l, tr}}) \leq \rho_w W(\overrightarrow{X_{opt}})$ (also defining threshold value ρ_w), the modification of $\overrightarrow{X_{l, tr}}$ into $(\overrightarrow{X_{l, tr}})'$ with JAYA-based exploitation/exploration schemes of Equation (16) and Equation (18), and the mirroring strategy to define $(\overrightarrow{X_{l, tr}})''$ if necessary. Hence, there are N_{POP} new operations for the elitist criterion (together with N_{POP} operations for defining ρ_w), $N_{POP} \times NDV$ new operations for the JAYA-based strategies (Equation (16) and Equation (18) may be used in alternative to perturb the trial designs $\overrightarrow{X_{l, tr}}$ that will be exploited or the trial designs for which re-exploration is necessary), and $2 \times NDV$ new operations for the mirroring strategy of Equation (20). The computational complexity of this phase for each optimization iteration hence is $O(2 \times (N_{POP} + NDV) + N_{POP} + NDV)$.

- (v) The preparation of the new iteration includes the re-sorting of the population and another mirroring strategy (Equation (21)) to avoid the stagnation of wolves α , β , and δ . The computational complexity of this phase for each iteration is hence $O(2 \times NDV + N_{POP} \times \log N_{POP})$.

By summing over the computational complexities of steps (i) through (v), the resulting total computational complexity of HALSGWJA over N_{iter} iterations performed in the optimization process is $O(N_{POP} \times (NDV + \log N_{POP}) + N_{iter} \times (N_{POP} + 2 \times \binom{N_{POP}}{2} + 3 \times N_{POP} + 2 \times N_{POP} \times NDV + 4 \times NDV + N_{POP} \times \log N_{POP}))$. For example, for $NDV = 30$ (i.e., the size of the largest test problem solved in this study) and $N_{POP} = 10$ (i.e., the population size used in the present optimization runs), the computational complexity of HALSGWJA would be $O(325 + N_{iter} \times 875)$ vs. at most $O(N_{iter} \times 400)$ recorded for typical GWO/JAYA formulations (see, for example, discussion in Ref. [6]). However, the inherent ability of HALSGWJA to generate very high-quality trial designs in each iteration allowed the present algorithm to efficiently explore/exploit search space. Consequently, HALSGWJA converged to the optimum in a few iterations, thus reducing the total number of operations entailed by the search process.

3. Test Problems and Optimization Results

The HALSGWJA algorithm developed in this study was tested on 20 real-world mechanical engineering problems. The first 19 problems belong to the CEC2020 suite of mechanical engineering problems (also indicated as RC15-RC33 in the literature). Table 1 presents the main features of the CEC2020 test problems, which include up to 30 design variables and 86 inequality constraints. The table lists the goal pursued in each test problem; all test cases are minimization problems, except problem 14, where the objective is to maximize the capacity of a rolling element bearing. Problems 4, 8, 12, 16, and 17 include either discrete or integer variables. Problems 8, 9, and 17 also include equality constraints. The full description of the CEC2020 test problems (including the number of design variables NDV , the number of inequality constraints G , and the number of equality constraints H for each problem) is given in Ref. [52] and recalled at different extents in Refs. [53–69] for the whole library, as well as in Refs. [6,14,32,70–87] for specific problems. Hence, for the sake of brevity, this paper presents in detail only the formulations of problems 2, 12, and 19 that include the largest numbers of optimization variables and constraints (i.e., 14, 22, and 30 and 15, 86, and 30, respectively). The formulation of problem 14 also is recalled because this test case is the only maximization problem in the test suite and it may have different optima. The target values of objective function to be reached in the nineteen CEC2020 mechanical engineering problems are listed in Table 1, considering 11 significant digits.

Table 1. Description of the CEC2020 real-world mechanical engineering test problems.

Problem	Description	Objective	NDV	G	H	Optimum
1	Speed reducer	Minimize weight	7	11	0	2.9944244658×10^3
2	Industrial refrigeration system design	Min. fabrication/operation costs	14	15	0	$3.2213000814 \times 10^{-2}$
3	Tension/compression spring design (1)	Minimize weight	3	3	0	$1.2665232788 \times 10^{-2}$
4	Pressure vessel design	Minimize fabrication cost	4	4	0	5.8853327736×10^3
5	Welded beam design	Minimize fabrication cost	4	5	0	1.6702177263×10^0
6	Three-bar truss design	Minimize weight	2	3	0	2.6389584338×10^2
7	Multiple disk clutch brake design	Minimize mass	5	7	0	$2.3524245790 \times 10^{-1}$
8	Planetary gear train design optimization	Minimize max errors in gear ratios	9	10	1	$5.2576870748 \times 10^{-1}$
9	Step-cone pulley problem	Minimize weight	5	8	3	1.6069868725×10^1
10	Robot gripper problem	Minimize difference max/min forces	7	7	0	2.5287918415×10^0
11	Hydrostatic thrust bearing design	Minimize bearing power loss	4	7	0	1.6161197651×10^3
12	Four-stage gearbox problem	Minimize weight	22	86	0	3.5359231973×10^1
13	Ten-bar truss design	Minimize weight	10	3	0	5.2445076066×10^2

14	Rolling element bearing	Maximize load carrying capacity	10	9	0	1.4614135715×10^4
15	Gas transmission compressor design	Minimize cost of gas delivery	4	1	0	2.9648954173×10^6
16	Tension/compression spring design (2)	Minimize steel wire volume	3	8	0	2.6138840583×10^0
17	Gear train design	Match target total gear ratio	4	1	1	0.0000000000×10^0
18	Himmelblau's function	Minimize quadratic model	5	6	0	$-3.0665538672 \times 10^4$
19	Topology optimization	Material layout for min compliance	30	30	0	2.6393464970×10^0

Problem 2 aims to minimize the fabrication and operation costs of industrial refrigeration systems with constraints on geometry and heat transfer characteristics. The design variables of this highly nonlinear mechanical engineering optimization problem are as follows: evaporator shell diameter (x_1), condenser shell diameter (x_2), evaporator shell thickness (x_3), condenser shell thickness (x_4), evaporator tubesheet thickness (x_5), condenser tubesheet thickness (x_6), evaporator fluid velocity (x_7), condenser fluid velocity (x_8), evaporator tube passes (x_9), condenser tube passes (x_{10}), evaporator tube length (x_{11}), condenser tube length (x_{12}), condenser effectiveness (x_{13}), and inlet/outlet fluid temperature difference (x_{14}). All design variables vary between 0.001 and 5. The optimization problem is stated as follows:

$$\begin{aligned}
 & \text{Minimize } W(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}) = \\
 & 63098.88 \cdot x_2 \cdot x_4 \cdot x_{12} + 5441.5 \cdot x_2^2 \cdot x_{12} + 115055.5 \cdot x_2^{1.664} \cdot x_6 + 6172.27 \cdot x_2^2 \cdot x_6 + 63098.88 \cdot x_1 \cdot x_3 \cdot x_{11} + 5441.5 \\
 & \cdot x_1^2 \cdot x_{11} + 115055.5 \cdot x_1^{1.664} \cdot x_5 + 6172.27 \cdot x_1^2 \cdot x_5 + 140.53 \cdot x_1 \cdot x_{11} + 281.29 \cdot x_1 x_3 + 70.26 \cdot x_1^2 + 281.29 \\
 & \cdot x_3 x_{11} + 281.29 \cdot x_3^2 + 14437 \cdot x_1^2 \cdot x_8^{1.8812} \cdot x_{12}^{0.3424} \cdot x_7 \cdot x_9^{-1} \cdot x_{10} \cdot x_{14}^{-1} + 20470.2 \cdot x_1^2 \cdot x_7^{2.893} \cdot x_{11}^{0.316} \\
 & \text{Subject to:} \\
 & \left\{ \begin{aligned} g_1(\bar{X}) &= 1.524 \cdot x_7^{-1} - 1 \leq 0 \\ g_2(\bar{X}) &= 1.524 \cdot x_8^{-1} - 1 \leq 0 \\ g_3(\bar{X}) &= 0.07789 \cdot x_1 - 2 \cdot x_7^{-1} \cdot x_9 - 1 \leq 0 \\ g_4(\bar{X}) &= 7.05305 \cdot x_1^2 \cdot x_2^{-1} \cdot x_8^{-1} \cdot x_9^{-1} \cdot x_{10} \cdot x_{14}^{-1} - 1 \leq 0 \\ g_5(\bar{X}) &= 0.0833 \cdot x_{13}^{-1} \cdot x_{14} - 1 \leq 0 \\ g_6(\bar{X}) &= 47.136 \cdot x_2^{0.333} \cdot x_{10}^{-1} \cdot x_{12} - 1.333 \cdot x_8 \cdot x_{13}^{2.1195} + 62.08 \cdot x_8^{0.2} \cdot x_{10}^{-1} \cdot x_{12}^{-1} \cdot x_{13}^{2.1195} - 1 \leq 0 \\ g_7(\bar{X}) &= 0.04771 \cdot x_8^{1.8812} \cdot x_{10} \cdot x_{12}^{0.3424} - 1 \leq 0 \\ g_8(\bar{X}) &= 0.0488 \cdot x_7^{1.893} \cdot x_9 \cdot x_{11}^{0.316} - 1 \leq 0 \\ g_9(\bar{X}) &= 0.0099 \cdot x_1 \cdot x_3^{-1} - 1 \leq 0 \\ g_{10}(\bar{X}) &= 0.0193 \cdot x_2 \cdot x_4^{-1} - 1 \leq 0 \\ g_{11}(\bar{X}) &= 0.0298 \cdot x_1 \cdot x_5^{-1} - 1 \leq 0 \\ g_{12}(\bar{X}) &= 0.056 \cdot x_2 \cdot x_6^{-1} - 1 \leq 0 \\ g_{13}(\bar{X}) &= 2 \cdot x_9^{-1} - 1 \leq 0 \\ g_{14}(\bar{X}) &= 2 \cdot x_{10}^{-1} - 1 \leq 0 \\ g_{15}(\bar{X}) &= x_{12} \cdot x_{11}^{-1} - 1 \leq 0 \end{aligned} \right. \quad (23)
 \end{aligned}$$

Problem 12 aims to minimize the structural weight of a four-stage gearbox. The design variables correspond to positions of the gears (x_{gi} , y_{gi}) and pinions (x_{pi} , y_{pi}), blank thickness (b_i), and number of teeth (N_{pi} , N_{gi}), with $i = 1, 2, 3, 4$. The variables can take the following discrete values: $x_{pi}, y_{pi}, x_{gi}, y_{gi} \in \{12.7, 25.4, 38.1, 50.8, 63.5, 88.9, 101.6, 114.3\}$; $b_i \in \{3.175, 5.715, 8.255, 12.7\}$; $7 \leq N_{pi}, N_{gi} \leq 76$ (integers). Design constraints refer to the pitch, kinematics, contact ratio, gear strengths, assembly, and size. The feasible region of search-space is only 0.01% of the total search space, thus leading to the presence of many local solutions. The optimization problem is stated as follows:

$$\text{Minimize } W(x_1, x_2, x_3, x_4 \dots x_{19}, x_{20}, x_{21}, x_{22}) = \left(\frac{\pi}{1000} \right) \sum_{i=1}^4 \frac{b_i c_i^2 (N_{pi}^2 + N_{gi}^2)}{(N_{pi} + N_{gi})^2}$$

Subject to:

$$\left\{ \begin{array}{l} g_1(\bar{X}) = \left(\frac{366000}{\pi \omega_1} + \frac{2c_1 N_{p1}}{N_{p1} + N_{g1}} \right) \left(\frac{(N_{p1} + N_{g1})^2}{4b_1 c_1^2 N_{p1}} \right) - \frac{\sigma_N J_R}{0.0167 W K_o K_M} \leq 0 \\ g_2(\bar{X}) = \left(\frac{366000 N_{g1}}{\pi \omega_1 N_{p1}} + \frac{2c_2 N_{p2}}{N_{p2} + N_{g2}} \right) \left(\frac{(N_{p2} + N_{g2})^2}{4b_2 c_2^2 N_{p2}} \right) - \frac{\sigma_N J_R}{0.0167 W K_o K_M} \leq 0 \\ g_3(\bar{X}) = \left(\frac{366000 N_{g1} N_{g2}}{\pi \omega_1 N_{p1} N_{p2}} + \frac{2c_3 N_{p3}}{N_{p3} + N_{g3}} \right) \left(\frac{(N_{p3} + N_{g3})^2}{4b_3 c_3^2 N_{p3}} \right) - \frac{\sigma_N J_R}{0.0167 W K_o K_M} \leq 0 \\ g_4(\bar{X}) = \left(\frac{366000 N_{g1} N_{g2} N_{g3}}{\pi \omega_1 N_{p1} N_{p2} N_{p3}} + \frac{2c_4 N_{p4}}{N_{p4} + N_{g4}} \right) \left(\frac{(N_{p4} + N_{g4})^2}{4b_4 c_4^2 N_{p4}} \right) - \frac{\sigma_N J_R}{0.0167 W K_o K_M} \leq 0 \\ g_5(\bar{X}) = \left(\frac{366000}{\pi \omega_1} + \frac{2c_1 N_{p1}}{N_{p1} + N_{g1}} \right) \left(\frac{(N_{p1} + N_{g1})^3}{4b_1 c_1^2 N_{g1} N_{p1}^2} \right) - \left(\frac{\sigma_H}{c_p} \right)^2 \frac{\sin \phi \cos \phi}{0.0334 W K_o K_M} \leq 0 \\ g_6(\bar{X}) = \left(\frac{366000 N_{g1}}{\pi \omega_1 N_{p1}} + \frac{2c_2 N_{p2}}{N_{p2} + N_{g2}} \right) \left(\frac{(N_{p2} + N_{g2})^3}{4b_2 c_2^2 N_{g2} N_{p2}^2} \right) - \left(\frac{\sigma_H}{c_p} \right)^2 \frac{\sin \phi \cos \phi}{0.0334 W K_o K_M} \leq 0 \\ g_7(\bar{X}) = \left(\frac{366000 N_{g1} N_{g2}}{\pi \omega_1 N_{p1} N_{p2}} + \frac{2c_3 N_{p3}}{N_{p3} + N_{g3}} \right) \left(\frac{(N_{p3} + N_{g3})^3}{4b_3 c_3^2 N_{g3} N_{p3}^2} \right) - \left(\frac{\sigma_H}{c_p} \right)^2 \frac{\sin \phi \cos \phi}{0.0334 W K_o K_M} \leq 0 \\ g_8(\bar{X}) = \left(\frac{366000 N_{g1} N_{g2} N_{g3}}{\pi \omega_1 N_{p1} N_{p2} N_{p3}} + \frac{2c_4 N_{p4}}{N_{p4} + N_{g4}} \right) \left(\frac{(N_{p4} + N_{g4})^3}{4b_4 c_4^2 N_{g4} N_{p4}^2} \right) - \left(\frac{\sigma_H}{c_p} \right)^2 \frac{\sin \phi \cos \phi}{0.0334 W K_o K_M} \leq 0 \\ g_{9-12}(\bar{X}) = -N_{pi} \sqrt{\frac{\sin 2\phi}{4} - \frac{1}{N_{pi}}} + \left(\frac{1}{N_{pi}} \right)^2 + N_{gi} \sqrt{\frac{\sin 2\phi}{4} + \frac{1}{N_{gi}}} \left(\frac{1}{N_{gi}} \right)^2 + \frac{\sin \phi (N_{pi} + N_{gi})}{2} + C R_{min} \pi \cos \phi \leq 0 \\ g_{13-16}(\bar{X}) = d_{min} - \frac{2c_i N_{pi}}{N_{pi} + N_{gi}} \leq 0 \\ g_{17-20}(\bar{X}) = d_{min} - \frac{2c_i N_{gi}}{N_{pi} + N_{gi}} \leq 0 \\ g_{21}(\bar{X}) = x_{p1} + \frac{(N_{p1} + 2)c_1}{N_{p1} + N_{g1}} - L_{max} \leq 0 \\ g_{22-24}(\bar{X}) = -L_{max} + \left(\frac{(N_{pi} + 2)c_i}{N_{pi} + N_{gi}} \right)_{i=2,3,4} + x_{g(i-1)} \leq 0 \\ g_{25}(\bar{X}) = -x_{p1} + \left(\frac{(N_{p1} + 2)c_1}{N_{p1} + N_{g1}} \right) \leq 0 \\ g_{26-28}(\bar{X}) = \left(\frac{(N_{pi} + 2)c_i}{N_{pi} + N_{gi}} - x_{g(i-1)} \right)_{i=2,3,4} \leq 0 \\ g_{29}(\bar{X}) = y_{p1} + \left(\frac{(N_{p1} + 2)c_1}{N_{p1} + N_{g1}} \right) - L_{max} \leq 0 \\ g_{30-32}(\bar{X}) = -L_{max} + \left(\frac{(N_{pi} + 2)c_i}{N_{pi} + N_{gi}} + y_{g(i-1)} \right)_{i=2,3,4} \leq 0 \\ g_{33}(\bar{X}) = -y_{p1} + \left(\frac{(N_{p1} + 2)c_1}{N_{p1} + N_{g1}} \right) \leq 0 \\ g_{34-36}(\bar{X}) = \left(\frac{(N_{pi} + 2)c_i}{N_{pi} + N_{gi}} - y_{g(i-1)} \right)_{i=2,3,4} \leq 0 \\ g_{37-40}(\bar{X}) = -L_{max} + \frac{(N_{gi} + 2)c_i}{N_{pi} + N_{gi}} + x_{gi} \leq 0 \\ g_{41-44}(\bar{X}) = -x_{gi} + \frac{(N_{gi} + 2)c_i}{N_{pi} + N_{gi}} \leq 0 \\ g_{45-48}(\bar{X}) = y_{gi} + \frac{(N_{gi} + 2)c_i}{N_{pi} + N_{gi}} - L_{max} \leq 0 \\ g_{49-52}(\bar{X}) = -y_{gi} + \frac{(N_{gi} + 2)c_i}{N_{pi} + N_{gi}} \leq 0 \\ g_{53-56}(\bar{X}) = (b_i - 8.255)(b_i - 5.715)(b_i - 12.70)(0.945c_i - N_{pi} - N_{gi})(-1) \leq 0 \\ g_{57-60}(\bar{X}) = (b_i - 8.255)(b_i - 3.175)(b_i - 12.70)(0.646c_i - N_{pi} - N_{gi}) \leq 0 \\ g_{61-64}(\bar{X}) = (b_i - 5.715)(b_i - 3.175)(b_i - 12.70)(0.504c_i - N_{pi} - N_{gi}) \leq 0 \\ g_{65-68}(\bar{X}) = (b_i - 5.715)(b_i - 3.175)(b_i - 8.255)(0c_i - N_{pi} - N_{gi}) \leq 0 \\ g_{69-72}(\bar{X}) = (b_i - 8.255)(b_i - 5.715)(b_i - 12.70)(N_{pi} + N_{gi} - 1.812c_i)(-1) \leq 0 \\ g_{73-76}(\bar{X}) = (b_i - 8.255)(b_i - 3.175)(b_i - 12.70)(N_{pi} + N_{gi} - 0.945c_i) \leq 0 \\ g_{77-80}(\bar{X}) = (b_i - 5.715)(b_i - 3.175)(b_i - 12.70)(N_{pi} + N_{gi} - 0.646c_i)(-1) \leq 0 \\ g_{81-84}(\bar{X}) = (b_i - 5.715)(b_i - 3.175)(b_i - 8.255)(N_{pi} + N_{gi} - 0.504c_i) \leq 0 \\ g_{85}(\bar{X}) = \omega_{MIN} - \frac{\omega_1 N_{p1} N_{p2} N_{p3} N_{p4}}{N_{g1} N_{g2} N_{g3} N_{g4}} \leq 0 \\ g_{86}(\bar{X}) = \frac{\omega_1 N_{p1} N_{p2} N_{p3} N_{p4}}{N_{g1} N_{g2} N_{g3} N_{g4}} - \omega_{MAX} \leq 0 \end{array} \right. \quad (24)$$

where $c_i = \sqrt{(y_{gi} - y_{pi})^2 + (x_{gi} - x_{pi})^2}$, $d_{min} = 25$, $L_{max} = 127$, $K_o = 1.5$, $K_M = 1.6$, $J_R = 0.2$, $\phi = 120^\circ$, $W = 55.9$, $CR_{min} = 1.4$, $C_p = 464$, $\sigma_N = 2090$, $\sigma_H = 3290$, $\omega_1 = 5000$, $\omega_{min} = 245$, $\omega_{max} = 255$.

The goal of test problem 14 was to maximize the dynamic load capacity of a rolling element bearing. The optimization variables are as follows: pitch diameter D_m (x_1), balls diameter D_b (x_2), number of balls Z (x_3 , integer variable), inner raceway curvature coefficient f_i (x_4), outer raceway curvature coefficient f_o (x_5), minimum ball diameter limiter KD_{min} (x_6), maximum ball diameter limiter KD_{max} (x_7), outer ring strength parameter ε (x_8), mobility condition parameter e (x_9), and bearing width limiter ζ (x_{10}). The design variables can vary as follows: $125 \leq D_m \leq 150$; $10.5 \leq D_b \leq 31.5$; $4 \leq Z \leq 50$; $0.515 \leq f_i, f_o \leq 0.6$; $0.4 \leq KD_{min} \leq 0.5$; $0.6 \leq KD_{max} \leq 0.7$; $0.3 \leq \varepsilon \leq 0.4$; $0.02 \leq e \leq 0.1$; $0.6 \leq \zeta \leq 0.85$. The optimization problem is stated as

$$\text{Maximize } W(x_1, x_2, x_3, x_4, x_5) = \begin{cases} f_c Z^{2/3} D_b^{1.8} & , \text{ if } D_b \leq 25.4 \text{ mm} \\ 3.647 f_c Z^{2/3} D_b^{1.4} & , \text{ if } D_b > 25.4 \text{ mm} \end{cases}$$

Subject to:

$$\begin{cases} g_1(\bar{X}) = \frac{\phi_o}{2 \arcsin(\frac{D_b}{D_m})} - Z + 1 \geq 0 \\ g_2(\bar{X}) = 2D_b - KD_{min}(D - d) \geq 0 \\ g_3(\bar{X}) = KD_{max}(D - d) - 2D_b \geq 0 \\ g_4(\bar{X}) = \zeta B_w - D_b \leq 0 \\ g_5(\bar{X}) = D_m - 0.5(D + d) \geq 0 \\ g_6(\bar{X}) = (0.5 + e)(D + d) - D_m \geq 0 \\ g_7(\bar{X}) = 0.5(D - D_m - D_b) - \varepsilon D_b \geq 0 \\ g_8(\bar{X}) = f_i \geq 0.515 \\ g_9(\bar{X}) = f_o \geq 0.515 \end{cases} \quad (25)$$

The cost function W depends on the first five design variables only. The coefficient f_c included in the cost function W expression depends on the $\gamma = D_b/D_m$ ratio and the inner/outer raceway curvature coefficients f_i and f_o as

$$f_c = 37.91 \left[1 + \left\{ 1.04 \left(\frac{1 - \gamma}{1 + \gamma} \right)^{1.72} \left(\frac{f_i(2f_o - 1)}{f_o(2f_i - 1)} \right)^{0.41} \right\}^{10/3} \right]^{-0.3} \left[\frac{\gamma^{0.3}(1 - \gamma)^{1.39}}{(1 + \gamma)^{1/3}} \right] \left[\frac{2f_i}{2f_i - 1} \right]^{0.41}$$

The following parameters/expressions are utilized to define the constraint functions:

$$D = 160, \quad d = 90, \quad B_w = 30, \quad T = D - d - 2D_b, \quad \phi_o = 2\pi - \arccos\left(\frac{[(D-d)/2 - 3(T/4)]^2 + \{D/2 - T/4 - D_b\}^2 - \{d/2 + T/4\}^2}{2\{(D-d)/2 - 3(T/4)\}\{D/2 - T/4 - D_b\}}\right).$$

The objective of problem 19 is to optimize the material layout for a given set of loads, given the design search space and constraints related to system performance. A simply supported bar is discretized in 30 elements and loaded by a vertical force F at mid-bay. The optimizer minimizes the compliance of the structure (this is equivalent to maximize stiffness) by iteratively adjusting the material topology distribution within the design space. For that purpose, a “density” variable x_e is assigned to each element to assess the contribution of that element to the global compliance/stiffness of the structure. This contribution is expressed through a power law. The optimization problem is stated as follows:

$$\text{Minimize } W(x_1, x_2, x_3, x_4 \dots x_{27}, x_{28}, x_{29}, x_{30}) = \{u\}^T [K] \{u\} = \sum_{e=1}^{30} (x_e)^p u_e^T k_e u_e$$

Subject to:

$$\begin{cases} H_1(\bar{X}) = \frac{V(\bar{X})}{V_o} - 1 = 0 \\ H_2(\bar{X}) = [K]\{u\} - \{F\} = 0 \end{cases} \quad (26)$$

where the “density” variables must vary as $0 < \bar{x}_{min} \leq x_e \leq 1$. The p exponent of the density power law is set equal to 3. $[K]$ and $\{u\}$, respectively, are the global stiffness matrix and the nodal displacement vector, with the corresponding element displacement vector u_e and stiffness matrix k_e of the e -th element. The two constraints in Equation (25) state that the total volume of the structure must be equal to some target value V_o and that the nodal forces must match the applied loads.

The last mechanical engineering optimization problem solved in this study regarded the optimal crashworthiness design of a vehicle; structural weight must be minimized so that the vehicle can adsorb the side impact from a moving barrier. The vehicle’s structure must be optimized to safely protect passengers, minimizing weight and preserving the vehicle’s performance. Figure 2 (adapted from Ref. [7]) illustrates the vehicle frame parts to be optimized; the moving barrier and the direction of the lateral impact also are shown in the figure.

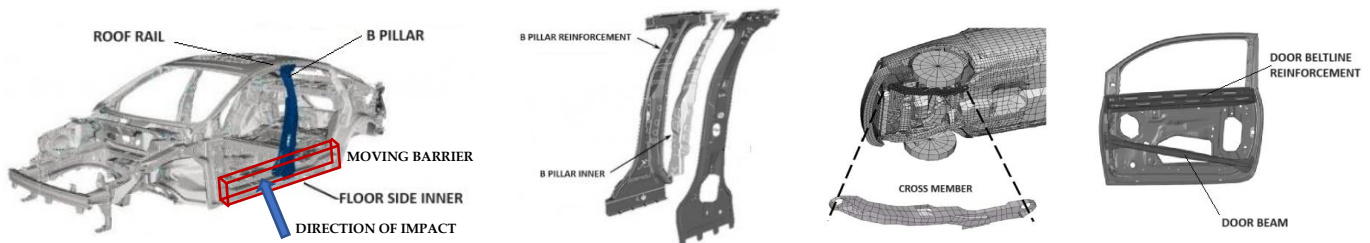


Figure 2. Schematic of vehicle parts (adapted from [7]) optimized in the car side impact problem.

The test problem has 11 design variables: inner thickness of B-pillar (x_1), reinforcement thickness of B-pillar (x_2), inner thickness of floor side (x_3), thickness of cross members (x_4), thickness of door beam (x_5), reinforcement of door beltline (x_6), thickness of roof rail (x_7); shape factor for the B-pillar inner (x_8) and shape factor for the floor side inner (x_9); height of the barrier (x_{10}), and the hitting position of the barrier with respect to the center of mass of the vehicle (x_{11}). All variables are continuous except x_8 and x_9 . The optimal crashworthiness design problem is formulated as

$$\text{Minimize } W(\vec{X}) = 1.98 + 4.90x_1 + 6.67x_2 + 6.98x_3 + 4.01x_4 + 1.78x_5 + 2.73x_7 \quad (27)$$

Subject to:

$$\left\{ \begin{array}{l}
G_1(\vec{X}) = 1.16 - 0.3717x_2x_4 - 0.00931x_2x_{10} - 0.484x_3x_9 + 0.01343x_6x_{10} \leq 1 \\
G_2(\vec{X}) = 0.261 - 0.0159x_1x_2 - 0.188x_1x_8 - 0.019x_2x_7 + 0.0144x_3x_5 + 0.0008757x_5x_{10} + \\
\quad 0.08045x_6x_9 + 0.00139x_8x_{11} + 0.00001575x_{10}x_{11} \leq 0.32 \\
G_3(\vec{X}) = 0.214 + 0.00817x_5 - 0.131x_1x_8 - 0.0704x_1x_9 + 0.03099x_2x_6 - 0.018x_2x_7 + \\
\quad 0.0208x_3x_8 + 0.121x_3x_9 - 0.00364x_5x_6 + 0.0007715x_5x_{10} - 0.0005354x_6x_{10} + \\
\quad 0.00121x_8x_{11} + 0.00184x_9x_{10} - 0.02x_2^2 \leq 0.32 \\
G_4(\vec{X}) = 0.074 - 0.61x_2 - 0.163x_3x_8 + 0.001232x_3x_{10} - 0.166x_7x_9 + 0.227x_2^2 \leq 0.32 \\
G_5(\vec{X}) = 28.98 + 3.818x_3 - 4.2x_1x_2 + 0.0207x_5x_{10} + 6.63x_6x_9 - 7.7x_7x_8 + 0.32x_9x_{10} \leq 32 \\
G_6(\vec{X}) = 33.86 + 2.95x_3 + 0.1792x_{10} - 5.057x_1x_2 - 11.0x_2x_8 - 0.0215x_5x_{10} - 9.98x_7x_8 + \\
\quad 22.0x_8x_9 \leq 32 \\
G_7(\vec{X}) = 46.36 - 9.9x_2 - 12.9x_1x_8 + 0.1107x_3x_{10} \leq 32 \\
G_8(\vec{X}) = 4.72 - 0.5x_4 - 0.19x_2x_3 - 0.0122x_4x_{10} + 0.009325x_6x_{10} + 0.000191x_{11}^2 \leq 4 \\
G_9(\vec{X}) = 10.58 - 0.674x_1x_2 - 1.95x_2x_8 + 0.02054x_3x_{10} - 0.0198x_4x_{10} + 0.028x_6x_{10} \leq 9.9 \\
G_{10}(\vec{X}) = 16.45 - 0.489x_3x_7 - 0.843x_5x_6 + 0.0432x_9x_{10} - 0.0556x_9x_{11} - 0.000786x_{11}^2 \leq 15.7
\end{array} \right. \quad (28)$$

The objective function $W(\vec{X})$ (structural weight of vehicle parts to be optimized) and constraints functions $G(\vec{X})$ were fitted in [88] using response surface models to deal with the high nonlinearity of the problem. More details on this test problem are available in Refs. [7,70,72,73,76,81,83–85,87–90]. The best solution ever reported in the optimization literature, $\{0.5; 1.21204; 0.5; 0.77908; 0.5; 1.49004; 0.5; 0.345; 0.345; -28.9781; 0.0001\}$, achieved a minimum structural weight of 21.38340 kg [7].

HALSGWJA was implemented in MATLAB Version 2025b as a standalone optimization code. Optimization runs were executed on a standard portable computer with 16 MB RAM memory and a 3.1 Mhz AMD processor. To gather statistically significant information on HALSGWJA's computational efficiency, each test problem was solved by executing 25 independent optimization runs from different initial populations. Population size and the limit number of iterations of HALSGWJA were always set to 10 and 5000, respectively, except for in the CEC2020 problems 2 and 11, where the limit number of iteration was increased to 10,000. Similarly to in [6,7], initial populations always include candidate solutions with up to 1000% constraint violation to check the ability of the present algorithm to quickly direct search towards the region of the optimum solution. It should be noted that HALSGWJA always required much fewer analyses than the theoretical computational budget of $N_{POP} \times N_{itermax}$ analyses. The constraint evaluations performed in the optimization search for evaluating the updated population in each iteration accounted for more than 90% of the total computational cost of the optimization process. This confirms that the fast line search approach implemented by the present algorithm is computationally inexpensive.

HALSGWJA utilized the static penalty function (SPF) approach stated by Equations (2,3) to handle optimization constraints. In SPF, penalty factors are independent of the iteration number and remain constant over the entire search process (see the classical literature surveys [91,92]). Here, the penalty factor p included in Equation (2) was set equal to 2. Several constraint handling techniques were used by HALSGWJA's competitors besides SPF: (i) the dynamic penalty function (DPF) method, where the penalty term increases over the optimization iterations; (ii) DPF method, where penalty to infeasible solutions increases when trial solutions move away from the feasible region and gradually decreases as search approaches the feasible region; (iii) hybrid augmented Lagrangian method with interior point penalty (the penalty term is small at points far from constraint boundaries and tends to infinity at points near constraint boundaries); (iv) linear combination of cost function and optimization constraints with or without variation in penalty

factors; (v) adaptive penalty function (APF) approach, where an individual is reinitialized if it violates constraints; (v) death penalty or dubbed death penalty approaches, where an infeasible trial solution is eliminated or it is set an objective value high (low) enough for this solution to be eliminated throughout the optimization process if minimization (maximization) is the goal; (vii) hierarchical sorting (HS) prioritizing trial points with lower constraint values regardless of W values (HS differentiates between feasible and infeasible points always setting the lowest cost feasible solution as current best record; among infeasible points, a lower constraint violation is preferred; among infeasible and feasible points, feasible points are preferred); (viii) dynamic constraint tolerance, where the tolerance limit ε decreases through the optimization process and constraints are more relaxed in the early optimization cycles to maintain population diversity and then restricted gradually to find truly feasible solutions (for example, inequality constraints to be satisfied become $G(\vec{X}) \leq \varepsilon$ with ε ranging from 0.01 in the initial iteration to 0.001 in the final iteration). As mentioned above, HALSGWJA adopted the static penalty function approach with $p = 2$. However, similarly to in [6], a sensitivity analysis was carried out for HALSGWJA by varying the penalty factor p in Equation (2) from 1 to 1,000,000 as 10^0 , 10^1 , 10^2 , 10^3 , 10^4 , 10^5 , and 10^6 . This simulated a dynamic scenario where the penalty depends on the distance from constraint boundaries. Furthermore, setting large values for the penalty factor amplifies constraint violation and forces the optimizer to generate very high-quality trial designs that actually satisfy design constraints. Remarkably, the largest deviation from the best optimized cost obtained by HALSGWJA for varying penalty factors always remained below 0.001%, thus confirming the insensitivity of HALSGWJA to constraint handling options.

3.1. Results of CEC 2020 Problems

The HALSGWJA algorithm developed in this study was compared with 18 other state-of-the art metaheuristic optimizers. HALSGWJA's competitors can be grouped in three categories: (a) CEC2020 winners, (b) advanced variants of GWO and JAYA already proven to be very competitive with the CEC2020 winners, and (c) physics-based and animal behavior-inspired methods released between 2023 and 2025 that have been proven to be superior over highly performing or/and well established metaheuristic optimizers. In particular, the following algorithms were compared with HALSGWJA:

- (i) EnMODE (Enhanced Multi-Operator Differential Evolution) [54].
- (ii) COLSHADE (Linearly reduced population size Success History-based Adaptive Differential Evolution for Constrained Optimization) [55].
- (iii) Success History-based Adaptive Differential Evolution with a gradient-based repair strategy (En(L)SHADE) [56].
- (iv) Improved Multi-Operator Differential Evolution with a knowledge-guided information sharing strategy (IMODE-KG) [57].
- (v) Differential Evolution with self-adaptive Spherical Search (DESS) [58].
- (vi) Self-Adaptive Spherical Search [53].
- (vii) SDDS-SABC algorithm [59] combining Split-Detect-Discard-Shrink (which uses partitions to identify promising regions of search space) and Sophisticated Artificial Bee Colony (with modifications in the initialization process, employed and scout bees searching strategy).
- (viii) Improved Young's Double-Slit Experiment (IYDSE) [60].
- (ix) Kepler Optimization Algorithm (KOA) [74].
- (x) Improved Kepler Optimization algorithm (CGKOA) [61], including the adaptive function strategy, the sinusoidal chaotic gravity strategy, the lateral crossover strategy, and the elite gold rush strategy.
- (xi) Multi-Strategy Fusion Enhanced Particle Swarm Optimization (MSFPSO) [62].
- (xii) Enhanced Snow Ablation Optimizer (ESAO) [63].

- (xiii) Improved Dwarf Mongoose Optimization (IDMO) [64].
- (xiv) The hybrid LMWOAGWO algorithm combining Lévy flight with modified Whale Optimization Algorithm (WOA) and GWO [65].
- (xv) Contact List Subpopulation Mixed Evolution Grey Wolf Optimizer (CSELGWO) [66]. This multi-strategy enhanced GWO variant obtains high-quality local information on search space, then generates a subpopulation that is updated with the main population through subpopulation mixed evolution, thus significantly improving population diversity and convergence accuracy. Levy Flight with archives and activation mechanisms serves to escape from local optima.
- (xvi) Marine Predators Social Group Optimization (MPSGO) [67].
- (xvii) Modified Artificial Hummingbird Algorithm (MAHA) [68].
- (xviii) Enhanced JAYA (EJAYA) [70]. This powerful JAYA variant updates designs based on current best and worst solutions (similar to classical JAYA), average solution, and historical solutions (these form an auxiliary population initially generated besides standard population and probabilistically permuted in the search process). Local exploitation utilizes upper/lower local attractors (weighted averages of best/worst and average solutions). Historical population guides global exploration.

The selected competitor algorithms make the evaluation of HALSGWJA's performance highly significant. In fact, EnMODE [54], COLSHADE [55], and SASS [53] were CEC2020 winners, being the top ranked challengers of the CEC competition on the real-world engineering problems, as reported in [56,58,59,65–68]. As reported in [56], En(L)SHADE achieved a better normalized score (i.e., 0.2849 vs. 0.3371 and 0.7162, respectively) and a higher feasibility rate (98.25% vs. 97.89% and 74.39%, respectively) than SASS and COLSHADE over all the independent optimization runs performed for the whole CEC2020 library of real-world engineering optimization problems. The normalized score was computed by weighing problem dimensionality with the normalized adjusted objective function values of the best, mean, and medium solutions. SDDS-ABC [59] achieved a better Friedman's test score than COLSHADE, specifically in the nineteen CEC2020 mechanical engineering problems considered in this study (2.32 vs. 2.37, see Ref. [59]).

IMODE-KG [57] outperformed other 12 metaheuristic optimizers including the winner algorithms EnMODE and SASS of the real-world mechanical engineering CEC2020 test suite. In fact, the Friedman's test score of IMODE-KG was 2.2 vs. 3.2 and 4.3667 achieved by EnMODE and SASS, respectively. Interestingly, these results were obtained with a computational budget of only 10,000 analyses, which is much smaller than the typical computational budget adopted in the CEC2020 competition, where the limit number of function evaluations is 100,000 analyses for $NDV \leq 10$ and 200,000 for $10 < NDV \leq 30$. For the same limited computational budget of 10,000 analyses, the advanced JAYA variant developed in [70]—EJAYA—was reported in [57] to be very competitive with EnMODE, SASS and IMODE-KG; in fact, it achieved a Friedman's test score value of 3.7, which was almost in the center of the 2.2 to 4.3667 range found for those three algorithms.

It should be noted that, in 2024, Ref. [69] presented some numerical experiments on CEC2020 real-world mechanical engineering problems, also reporting the precise number of analyses required by COLSHADE, SASS, and EnMODE in comparison to the Piranha Predation Optimization Algorithm (PPOA) developed in [69]. The theoretical computation budget was set to the very large value of 1,000,000 analyses. Interestingly, COLSHADE, SASS, and EnMODE could complete their optimizations within an average number of analyses lower than 10,000 in some optimization problems, respectively, from 220 to 8999 analyses in problems 6, 7, 15, 17, 18, and 19 for COLSHADE; from 360 to 7605 analyses in problems 9, 17, and 19 for SASS; and from 215 to 9948 analyses in problems 1, 2, 6, and 17 for EnMODE. However, the success rate of COLSHADE, SASS, and EnMODE in finding target solutions was found to be 0 in five optimization problems (i.e., 4, 8, 10,

14, and 16). This, openly in contrast with the findings of the other studies that re-run COLSHADE, SASS, and EnMODE optimizations for benchmark, might have been caused by the way in which the three CEC2020 winners were implemented in Ref. [69].

DESS [58] outperformed 18 other metaheuristic optimizers including the top-ranked CEC2020 real-world engineering problem solvers EnMODE, SASS, and COLSHADE. In particular, DESS achieved a normalized score of only 0.1230 vs. 0.1378, 0.2427, and 0.4514, respectively, achieved by the three above-mentioned algorithms. MPSGO [67] and MAHA [68] (two nature-inspired algorithms released in 2024 that reproduce animal behaviors that may also be integrated with human social behavior to solve complex problems) were also highly competitive against CEC2020 top solvers. In particular, MPSGO obtained the same best design as EnMODE and COLSHADE in 17 of the 19 CEC2020 mechanical engineering problems, while MAHA ranked second overall after SASS and above EnMODE and COLSHADE in the CEC2020 whole real-world engineering problems.

The hybrid LMWOAGWO algorithm [65] combining GWO and WOA, and the advanced GWO variant developed in [66]—CSELGWO—were also highly competitive against the CEC2020 winners EnMODE and COLSHADE, ranking immediately after these two algorithms in 17 of the 19 CEC2020 mechanical engineering problems. In [65], a score of 3.23 is reported for LMWOAGWO vs. 2.14 to 2.45, while in [66], the reported score for CSELGWO was 2.54 vs. 2.41 to 2.5.

IYDSE [60], KOA [74], GKOA [61], and ESAO [63] are four relevant physics-based metaheuristic algorithms released between 2023 and 2025 that ranked first amongst, respectively, 8, 9, 9, and 10 other metaheuristic optimizers, including, among others, PSO and improved PSO variants, GWO and improved GWO variants, WOA and enhanced WOA, starling murmuration optimizer (SMO), artificial hummingbird algorithm (AHA), and dwarf mongoose optimization (DMO). IDMO [64] is another algorithm reproducing the dwarf mongoose behavior that was released in 2023, developed to improve the capability of its mother algorithm DMO of solving complex problems; interestingly, IDMO outperformed 12 other metaheuristic algorithms including highly cited methods such as GWO and WOA. MSFPSO [62], released in 2025, outperformed other 10 metaheuristic algorithms including 5 other PSO variants.

Table 2 compares the optimization results obtained by HALSGWJA in the CEC2020 mechanical engineering problems with those reported in the literature for the selected competitors. The data reported in the table for HALSGWJA's competitors are relative to the best parameter settings indicated in the literature for each algorithm. For each test problem, the table reports the best (*B*), average (*A*), and worst (*W*) values of optimized cost and required number of analyses along with the corresponding values of standard deviation (*STD*) over the 25 independent runs. Statistical data on the computational cost of HALSGWJA's competitors (i.e., the number of analyses and corresponding standard deviations) are reported when available. For the sake of brevity, the average number of analyses and corresponding standard deviations are reported in the same row and indicated as (*A/D*); furthermore, the *A/B/W/STD* and *A/D* nomenclature is indicated only for HALSGWJA and the first test problem. The data relating to EnMODE, COLSHADE, SASS, IYDSE, KOA, and EJAYA were merged from the original studies presenting these algorithms for the first time and subsequent studies reporting further investigations of those methods; the best solutions for each test case are then reported in Table 2. The results obtained by HALSGWJA in the pressure vessel design problem 4 variant with continuous variables are indicated in italics in the table.

Table 2. Comparison of optimization results obtained by HALSGWJA and its competitors in the CEC2020 mechanical engineering problems.

Problem	HALSGWJA Present	EnMODE [54,65,68,69]♥	COLSHADE [55,65,68,69]♥	En(L)SHADE [56]	IMODE-KG [57]	DESS [58]	SASS [53,59,68,69]♥	SDDS-SABC [59]	IYDSE [57,60]
1 <i>Speed reducer</i>	(B) 2994.425	2994.4	2994.4	2.99×10^3	2994.425	2.89×10^3	2994.425	2963.911 ¹	2.99×10^3
	(A) 2994.425	2994.4	2994.4	2.99×10^3	2994.425	2.99×10^3	2994.425	2975.081 ¹	2.99×10^3
	(W) 2994.425	2994.4	2994.4	2.99×10^3	2994.425	N/A	2994.425	2982.083 ¹	3.10×10^3
	(STD) 0	4.6412×10^{-13}	4.5475×10^{-13}	0	1.1003×10^{-7}	N/A	4.641×10^{-13}	1.32691	0.216
	(B) 2326								
	(A/D) 2538 ± 116	100,000	100,000	100,000	10,000	100,000	100,000	20,000	30,000
2 <i>Industrial refrigeration sys- tem</i>	(W) 2728	{2277}♥	{12,930}♥				{51,430}♥		
	0.0322130	0.032213	0.032213	3.22×10^{-2}	N/A	3.22×10^{-2}	0.032213	0.0321969 ¹	3.29×10^{-2}
	0.0322132	0.032213	0.032213	3.22×10^{-2}	N/A	3.22×10^{-2}	0.032213	0.0334143	4.99×10^{-2}
	0.0322160	0.032213	0.032213	3.22×10^{-2}	N/A	N/A	0.032213	0.0335463	0.115
	6.7082×10^{-7}	3.1672×10^{-18}	0	0	N/A	N/A	1.416×10^{-17}	0.26439	0.0233
	4695								
3 <i>Tension/compres- sion spring (Case 1)</i>	5478 ± 1003	200,000	200,000	200,000	N/A	200,000	200,000	20,000	30,000
	6258	{9948}♥	{37,190}♥				{174,700}♥		
	0.012665	0.012665	0.012665	1.27×10^{-2}	0.0126742	1.27×10^{-2}	0.012665	0.0112423 ¹	1.27×10^{-2}
	0.012668	0.012710	0.012665	1.27×10^{-2}	0.0132706	1.27×10^{-2}	0.012665	0.0121864 ¹	1.27×10^{-2}
	0.012670	0.012719	0.012665	1.27×10^{-2}	0.0149735	N/A	0.012665	0.0136756	1.27×10^{-2}
	2.1152×10^{-6}	2.0138×10^{-5}	1.0625×10^{-7}	0	4.7759×10^{-4}	N/A	0	1.589×10^{-3}	1.34×10^{-7}
4 <i>Pressure vessel</i>	2103								
	3316 ± 1137	100,000	100,000	100,000	10,000	100,000	100,000	20,000	30,000
	4097	{47,560}♥	{14,630}♥				{43,890}♥		
	6059.714 (5885.331)	6059.7	6059.7	6.06×10^3	6059.714	6.06×10^3	6059.714	5979.985	6103.368
	6059.714 (5885.331)	6059.7	6062.2	6.06×10^3	6256.737	6.06×10^3	6059.714	5981.044	6385.287
	6059.714 (5885.331)	6059.7	6090.5	6.09×10^3	6410.087	N/A	6059.714	5985.026	6810.762
	0 (0)	9.2825×10^{-13}	8.3591	11.5	155.130	N/A	3.713×10^{-12}	2.96641	185.361
	3852 (3510)								
	5104 ± 1374	100,000	100,000	100,000	10,000	100,000	100,000	20,000	10,000

	(4672 ± 1203)								
	6193 (5713)	{N/A}♥	{N/A}♥				{N/A}♥		
5 Welded beam	1.670218	1.6702	1.6702	1.67	1.67022	1.67022	1.670218	1.452753 ¹	1.67
	1.670218	1.6702	1.6702	1.67	1.67022	1.67022	1.670218	1.580213 ¹	1.67
	1.670218	1.6702	1.6702	1.67	1.67022	N/A	1.670218	1.690036	1.67
	0	0	0	0	1.2677 × 10 ⁻⁹	N/A	2.266 × 10 ⁻¹⁶	0.96852	1.16 × 10 ⁻⁵
	2975								
	3224 ± 285	100,000	100,000	100,000	10,000	100,000	100,000	20,000	30,000
	3691	{22,930}♥	{16,060}♥				{67,660}♥		
6 Three-bar truss	263.8915	263.90	263.90	2.64 × 10 ²	N/A	2.64 × 10 ²	263.8958	263.6053 ¹	2.64 × 10 ²
	263.8915	263.90	263.90	2.64 × 10 ²	N/A	2.64 × 10 ²	263.8958	263.6724 ¹	2.64 × 10 ²
	263.8915	263.90	263.90	2.64 × 10 ²	N/A	N/A	263.8958	264.2198	2.64 × 10 ²
	0	0	0	0	N/A	N/A	5.802 × 10 ⁻¹⁴	1.275 × 10 ⁻³	4.12 × 10 ⁻¹⁴
	705								
	961 ± 168	100,000	100,000	100,000	N/A	100,000	100,000	20,000	30,000
	1224	{8581}♥	{5594}♥				{12,760}♥		
7 Multiple disk clutch brake	0.235243	0.23524	0.23524	0.235	0.23524	0.235	0.235242	0.218881 ¹	0.235
	0.235243	0.23524	0.23524	0.235	0.23524	0.235	0.235242	0.236467	0.235
	0.235243	0.23524	0.23524	0.235	0.23524	N/A	0.235242	0.251975	0.235
	0	1.1331 × 10 ⁻¹⁶	0	0	1.4021 × 10 ⁻¹⁶	N/A	2.833 × 10 ⁻¹⁷	2.656 × 10 ⁻²	3.37 × 10 ⁻¹¹
	792								
	900 ± 85	100,000	100,000	100,000	10,000	100,000	100,000	20,000	30,000
	1193	{13,040}♥	{4858}♥				{12,910}♥		
8 Planetary gear train	0.523250	0.52577	0.52577	0.526	0.526581	0.526	0.525967	0.523274	0.526
	0.525714	0.52691	0.54103	0.633	0.558675	0.532	1.001524	0.532176	0.528
	0.529188	0.53121	0.74667	1.300	0.674039	N/A	3.521656	0.538756	0.537
	1.2440 × 10 ⁻³	1.4402 × 10 ⁻³	0.042573	0.231	0.036539	N/A	0.725184	5.416 × 10 ⁻⁴	2.81 × 10 ⁻³
	2095								
	3260 ± 1039	100,000	100,000	100,000	10,000	100,000	100,000	20,000	30,000
	4364	{N/A}♥	{N/A}♥				{N/A}♥		
9 Step-cone Pulley	16.03601	16.070	16.070	16.1	N/A	16.1	16.06987	12.30213 ¹	16.1
	16.04033	16.070	16.070	16.1	N/A	16.1	16.06987	15.96217 ¹	16.1
	16.04529	16.070	16.070	16.1	N/A	N/A	16.06987	17.10746	16.1
	4.6437 × 10 ⁻³	3.3335 × 10 ⁻¹⁴	0	0	N/A	N/A	3.626 × 10 ⁻¹⁵	2.01784	4.30 × 10 ⁻⁶

	4278								
	4424 ± 322	100,000	100,000	100,000	N/A	100,000	100,000	20,000	30,000
	5000	{38,090}♥	{14,090}♥				{4826}♥		
10 <i>Robot gripper</i>	2.5437852	2.5438	2.5438	2.54	N/A	2.54	2.543786	2.673169	2.70
	2.5437854	2.5438	2.5438	2.54	N/A	2.54	2.543786	2.678953	3.14
	2.5437857	2.5438	2.5438	2.54	N/A	N/A	2.543786	2.681085	3.47
	2.8263 × 10 ⁻⁷	1.3501 × 10 ⁻¹²	0	0	N/A	N/A	0	0.264387	0.218
	3988								
	4645 ± 787	100,000	100,000	100,000	N/A	100,000	100,000	20,000	30,000
	5302	{N/A}♥	{N/A}♥				{N/A}♥		
11 <i>Hydrostatic thrust bearing</i>	1616.120	1616.1	1616.1	1.62 × 10 ³	1783.954	1.62 × 10 ³	1616.120	1839.641	2483.159
	1616.120	1616.1	1639.0	1.62 × 10 ³	1918.389	1.62 × 10 ³	1616.120	1841.907	3201.317
	1616.120	1616.1	2129.1	1.62 × 10 ³	2242.035	N/A	1616.123	1845.063	4174.048
	0	1.7751 × 10 ⁻¹¹	1007.3	0	115.871	N/A	9.425 × 10 ⁻⁴	1.42788	469.274
	7400								
	8624 ± 724	100,000	100,000	100,000	10,000	100,000	100,000	20,000	10,000
	9527	{43,930}♥	{66,620}♥				{64,410}♥		
12 <i>Four-stage gearbox</i>	35.29269	35.359	35.359	36.2	N/A	36.2	36.2504	43.14295	14.9 ²
	35.41012	35.728	36.611	39.6	N/A	39.5	38.5141	44.01259	33.9 ²
	35.65141	37.268	40.931	44.1	N/A	N/A	45.5064	45.36397	67.9
	0.1687	0.59911	1.3677	2.0	N/A	N/A	2.11482	0.49725	14.9
	591								
	1021 ± 318	200,000	200,000	200,000	N/A	200,000	200,000	20,000	30,000
	1531	{139,000}♥	{111,700}♥				{61,630}♥		
13 <i>Ten-bar truss</i>	524.5888	524.45	524.45	5.24 × 10 ²	524.480	5.24 × 10 ²	524.4576	522.9644 ¹	535.9481
	524.6004	524.45	524.45	5.24 × 10 ²	525.075	5.24 × 10 ²	524.4692	523.0724 ¹	545.6267
	524.6111	524.45	524.45	5.24 × 10 ²	528.814	N/A	524.4820	525.0326	559.1333
	0.0113519	3.7602 × 10 ⁻⁷	0	2.01 × 10 ⁻⁹	0.96481	N/A	6.620 × 10 ⁻³	3.572 × 10 ⁻²	5.8317
	1911								
	3321 ± 1092	100,000	100,000	100,000	10,000	100,000	100,000	20,000	10,000
	4801	{27,750}♥	{19,660}♥				{N/A}♥		
14 <i>Rolling element bearing</i>	16,958.202	16,958	16,958	1.70 × 10 ⁴	16,958.202	1.44 × 10 ⁴	14,614.136	14,811.085	16,981.126 ²
	16,958.202	16,958	16,958	1.70 × 10 ⁴	16,958.204	1.58 × 10 ⁴	14,614.136	14,835.501	17,045.036 ²
	16,958.202	16,958	16,958	1.70 × 10 ⁴	16,958.238	N/A	14,614.136	14,842.886	17,203.713 ²

<i>Step-cone pulley</i>	16.04033	8.5633 ³	16.59500	16.763	16.090	16.5	16.2	16.070	16.0699	N/A
	16.04529	8.5633 ³	17.13636	N/A	N/A	N/A	N/A	16.070	16.0699	N/A
	4.6437×10^{-3}	9.27×10^{-15}	0.52157	0.31569	3.8548×10^{-8}	0.209	0.2608	4.6376×10^{-7}	3.5527×10^{-15}	N/A
	4278 4424 ± 322 5000	50,000	N/A	15,000	50,000	100,000	100,000	100,000	100,000	N/A
10 <i>Robot gripper</i>	2.5437852	2.5455*	2.54379	2.5929	2.5528	N/A	N/A	2.5442	2.638958	2.601873
	2.5437854	2.6382	2.68250	2.8768	2.6979	N/A	N/A	2.5496	2.638958	3.135123
	2.5437857	2.7991	3.11950	N/A	N/A	N/A	N/A	2.5966	2.638958	3.883629
	2.8263×10^{-7}	0.071785	0.23285	0.25476	0.12596	N/A	N/A	0.010519	4.4409×10^{-16}	0.429461
	3988 4645 ± 787 5302	50,000	N/A	15,000	50,000	N/A	N/A	100,000	100,000	10,000
11 <i>Hydrostatic thrust bearing</i>	1616.120	2403.717 ⁺	137.08396 ⁴	1918.4	1640.9	N/A	N/A	1643.1	1624.990	1625.443
	1616.120	3168.443	143.78383 ⁴	3224.5	1744.5	1.83×10^3	1.616×10^3	2350.3	1625.431	1631.510
	1616.120	4551.426	270.73097 ⁴	N/A	N/A	N/A	N/A	2475.1	1625.450	1767.661
	0	514.898	29.8803	876.2	51.215	55.2	12.57	2.5456	0.0901429	26.272
	7400 8624 ± 724 9527	10,000	N/A	15,000	50,000	100,000	100,000	100,000	100,000	150,000
12 <i>Four-stage gearbox</i>	35.29269	36.6012*	98.69	41.697	37.461	N/A	N/A	35.359	34.63987 ⁵	N/A
	35.41012	43.8296	115448.92	4.8979×10^{14}	2.7239×10^{15}	N/A	N/A	35.699	39.19013	N/A
	35.65141	63.54	411255.69	N/A	N/A	N/A	N/A	37.258	48.81480	N/A
	0.1687	7.3255	126054.20	9.372×10^{14}	6.3419×10^{15}	N/A	N/A	0.56923	3.48167	N/A
	591 1021 ± 318 1531	50,000	N/A	15,000	50,000	N/A	N/A	200,000	200,000	N/A
13 <i>10 bar truss</i>	524.5888	529.4834 ⁺	524.50833	525.98	524.45	N/A	N/A	524.45	524.4095	525.0392
	524.6004	534.2848	527.44007	532.62	527.07	5.29×10^2	524.4	524.45	524.4591	528.3298
	524.6111	542.7236	532.35484	N/A	N/A	N/A	N/A	524.45	524.5082	535.6228
	0.0113519	3.62097	3.02564	3.5857	2.9632	2.32	4.989×10^{-3}	2.9397×10^{-3}	0.0257801	2.55928
	1911 3321 ± 1092 4801	10,000	N/A	15,000	50,000	100,000	100,000	100,000	100,000	10,000

14 <i>Rolling element bearing</i>	16,958.202	16,960.674 ⁺	16,958.20	16,958	16,958	N/A	N/A	16,958	14,614.136	16,958.202
	16,958.202	16,970.531	16,958.20	16,958	16,958	1.70×10^4	1.461×10^4	16,958	14,614.136	16,958.203
	16,958.202	16,986.506	16,958.20	N/A	N/A	N/A	N/A	16,958	14,614.136	16,958.206
	0	7.07606	0	0.013976	6.6483×10^{-8}	9.29	15.57	0	1.8190×10^{-12}	7.0516×10^{-4}
	2492									
	3117 ± 385	10,000	N/A	15,000	50,000	100,000	100,000	100,000	100,000	10,000
	3681									
15 <i>Gas transmission compressor</i>	2,964,893.937	2,965,375.3 ⁺	1,227,929.15 ⁴	2.9649×10^6	2.9649×10^6	N/A	N/A	2.9649×10^6	2,964,895.059	2,964,896.4
	2,964,893.937	2,967,464.3	1,227,929.15 ⁴	2.9790×10^6	2.9790×10^6	2.96×10^6	2.965×10^6	2.9649×10^6	2,964,895.340	2,964,916.3
	2,964,893.937	2,971,180.3	1,227,929.15 ⁴	N/A	N/A	N/A	N/A	2.9649×10^6	2,964,895.945	2,965,063.0
	0	1506.404	2.39×10^{-10}	16,072	8.33×10^{-10}	6.15	1.125×10^{-9}	2.5505×10^{-9}	0.183209	34.9261
	4377									
	5661 ± 932	10,000	N/A	15,000	50,000	100,000	100,000	100,000	100,000	10,000
	6587									
16 <i>Tension/compression spring (Case 2)</i>	2.658722	2.6586 [*]	2.65856	2.6586	2.6586	N/A	N/A	2.6586	2.658558	2.658560
	2.659550	2.6606	2.66879	2.9313	2.6586	2.71	2.666	2.6586	2.658559	2.662352
	2.660999	2.6995	2.69949	N/A	N/A	N/A	N/A	2.6586	2.658559	2.700421
	9.4671×10^{-4}	9.1532×10^{-3}	0.01819	0.2426	9.982×10^{-11}	0.0948	0.01644	1.6402×10^{-12}	2.7129×10^{-7}	0.0103828
	4223									
	4705 ± 325	50,000	N/A	15,000	50,000	100,000	100,000	100,000	100,000	10,000
	4996									
17 <i>Gear train</i>	2.3078×10^{-21}	0 [*]	0	0	6.7660×10^{-19}	N/A	N/A	0	0	1.350×10^{-18}
	3.5365×10^{-21}	6.57×10^{-22}	0	0	2.3015×10^{-16}	9.53×10^{-18}	7.917×10^{-15}	0	0	1.315×10^{-13}
	4.9608×10^{-21}	6.90×10^{-21}	0	N/A	N/A	N/A	N/A	0	0	1.444×10^{-12}
	1.1169×10^{-21}	2.02×10^{-21}	0	0	3.7492×10^{-16}	1.54×10^{-17}	3.838×10^{-14}	0	0	3.559×10^{-13}
	860									
	1170 ± 456	50,000	N/A	15,000	50,000	100,000	100,000	100,000	100,000	10,000
	1870									
18 <i>Himmelblau</i>	-30,664.873	-30,663.305 ⁺	-30,665.54	-30,666	-30,666	N/A	N/A	-30,666	-30,665.539	-30,665.54
	-30,664.848	-30,652.862	-30,665.54	-30,666	-30,666	-3.07×10^4	-3.067×10^4	-30,666	-30,665.539	-30,665.54
	-30,664.834	-30,625.231	-30,665.54	N/A	N/A	N/A	N/A	-30,666	-30,665.539	-30,665.53
	0.0195687	8.63379	0	9.9915×10^{-5}	5.4957×10^{-8}	0.318	1.109×10^{-11}	5.8638×10^{-8}	1.4552×10^{-11}	9.6829×10^{-4}
	4494									
	4694 ± 219	10,000	N/A	15,000	50,000	100,000	100,000	100,000	100,000	10,000

19 Topology Optimization	5001									
	2.63935	2.6393 [*]	2.63935	2.6393	2.6393	N/A	N/A	2.6393	2.639347	2.639364
	2.63935	2.6393	2.63935	2.6393	2.6583	2.64	2.639	2.6393	2.639347	2.661795
	2.63935	2.6393	2.63935	N/A	N/A	N/A	N/A	2.6393	2.639347	2.747870
	0	1.39×10^{-15}	9.11×10^{-16}	2.8925×10^{-10}	5.4519×10^{-2}	1.36×10^{-15}	3.392×10^{-15}	0	4.4409×10^{-16}	0.0281439
	2613									
	3021 ± 237	50,000	N/A	15,000	50,000	200,000	200,000	200,000	200,000	10,000
	3526									

¹ The optimized design corresponding to this solution was not shown in Ref. [59]. ² The optimized design corresponding to this solution was not shown in Ref. [60]. ³ The optimized design corresponding to this solution was not shown in Ref. [61]. ⁴ The optimized design corresponding to this solution was not shown in Ref. [62]. ⁵ The optimized design corresponding to this solution was not shown in Ref. [68]. ♥ Average number of analyses for the successful optimization runs of Ref. [69]. * The reported data are taken from Ref. [74]. + The reported data are taken from Ref. [57]. * The reported data are taken from Ref. [61].

It can be seen from Table 2 that HALSGWJA was the best optimizer overall in test problems 1 (speed reducer design), 2 (industrial refrigeration system), 3 (tension-compression spring design case 1), 4 (pressure vessel design), 5 (welded beam design), 6 (three-bar truss design), 7 (multiple disk clutch brake design), 8 (planetary gear train design), 9 (step-cone pulley design), and 11 (hydrostatic thrust bearing design), converging to the target optima of these problems and requiring fewer analyses than its 18 competitors to complete the optimization process. The target optimum costs indicated in Table 1 for the CEC2020 real-world mechanical engineering problems were also either reached or slightly improved by HALSGWJA in test cases 12 (four-stage gearbox design), 14 (rolling element bearing design), 15 (gas transmission compressor design), and 19 (topology optimization), again requiring fewer analyses than its 18 competitors to complete the optimizations.

It should be noted that the pressure vessel problem 4 actually presents two variants with either continuous or mixed optimization variables. Table 2 reports the corresponding optimization results; the (italics) notation is used for the HALSGWJA's data relative to the continuous problem variant. The vessel comprises a cylindrical segment closed at its ends by two spherical caps. The four design variables are the thickness of the cylindrical segment (x_1), the thickness of the spherical caps (x_2), the radius of the curvature of spherical caps (x_3), and the length of the cylindrical segment (x_4). The 5885.333 optimum cost indicated in Table 1 refers to the problem variant with continuous variables. In the problem variant with mixed variables, thickness variables x_1 and x_2 are multiple integers of the available thickness of rolled steel plates (see Ref. [6]); the corresponding optimum cost is 6059.714. The mixed variable version of the problem was solved by all of the algorithms compared in Table 2, except for CGKOA [61], MAHA [68], and EJAYA [57,70], which instead solved the continuous variables problem variant. Regardless of the problem variant, HALSGWJA always converged to the target solution of this problem, requiring between 3510 and 6193 analyses only vs. at least 10,000 analyses required by its competitors.

Another interesting issue regards the rolling element bearing design problem 14, where the objective is to maximize the load bearing capacity. The target optimum cost of 14,614.136, given in Table 1, is obtained for the optimum design vector $\overline{X}_{opt,1} = \{131.2; 18; 4; 0.6; 0.6; 0.4; 0.6; 0.3; 0.025; 0.6\}$. The third design variable, the number of balls Z , must take an integer value and it is hence rounded in the optimization process; in this case, it was reduced from 4.476 to 4. Another solution reported in the literature for two nature-inspired metaheuristic algorithms, such as the multi-strategy enhanced northern goshawk optimization algorithm (MENGO) [77] and the chaos-based mayfly algorithm with opposition-based learning and Levy flight (COLMA) [78], had an optimized cost of 16,958.2; the number of balls was rounded from 4.51 to 5 and the final design was $\overline{X}_{opt,2} = \{131.2; 18; 5; 0.6; 0.6; 0.4071; 0.6437; 0.3; 0.0632; 0.6\}$. While designs $\overline{X}_{opt,1}$ and $\overline{X}_{opt,2}$ shared the same values of design variables x_1 , x_2 , x_4 , and x_5 , the corresponding optimized costs were different because the problem cost function W stated by Equation (25) also depends on the rounded value of x_3 (Z), which changed from 4 to 5, passing from $\overline{X}_{opt,1}$ to $\overline{X}_{opt,2}$. The differences in variables x_6 to x_{10} observed between solutions $\overline{X}_{opt,1}$ and $\overline{X}_{opt,2}$ obviously did not affect the cost function values. Both $\overline{X}_{opt,1}$ and $\overline{X}_{opt,2}$ can be considered as target solutions for this problem; the 14,614.136 optimized cost was reached by SASS [53,58,59], DESS [58], SDDS-SABC [59], and MAHA [68], while the 16,958.2 optimized cost was reached by HALSGWJA and the remaining competitors. The high computational efficiency of the present algorithm was confirmed by its ability to converge to the 16,958.2 cost in all independent optimization runs.

Indeed, there are yet other solutions indicated in the literature for problem 14. For example, Ref. [75] reports, for the energy valley optimizer (a physics-based method), the optimized design $\overline{X}_{opt,3} = \{125.7191; 21.4256; 11; 0.515; 0.515; 0.4632; 0.6999; 0.3; 0.06343; 0.6042\}$,

yielding a cost of 81,859.8. Teaching–learning-based optimization (TLBO) [14] also converged practically to the same cost, 81,859.74, with the corresponding optimum design $\overline{X}_{opt,4} = \{125.7191; 21.42559; 11; 0.515; 0.515; 0.424266; 0.633948; 0.3; 0.068858; 0.799498\}$. In Ref. [84], it is reported that for the artificial lemming algorithm (a nature-inspired method), the optimal design is $\overline{X}_{opt,5} = \{125.7191; 21.4256; 11; 0.515; 0.515; 0.4014; 0.6138; 0.3; 0.0307; 0.6001\}$, yielding a cost of 85,549.2. The same optimized cost value of 85,549.2 was also reported in Ref. [70] for EJAYA, with the corresponding optimal design $\overline{X}_{opt,6} = \{125.7191; 21.4256; 11; 0.515; 0.515; 0.4009; 0.6228; 0.3; 0.1; 0.6023\}$. However, since $\overline{X}_{opt,3}$, $\overline{X}_{opt,4}$, $\overline{X}_{opt,5}$, and $\overline{X}_{opt,6}$ present the same values for x_1 , x_2 , x_3 , x_4 , and x_5 , the corresponding optimized values of W must coincide; it can be proven that the correct value of the optimized cost is 81,859.8, not 85,549.2. Interestingly, HALSGWJA always converged to the designs $\overline{X}_{opt,3}$ and $\overline{X}_{opt,4}$, and the corresponding optimized cost of 81,859.8, with 0 standard deviation, by simply restarting the search process from the initial populations obtained by perturbing $\overline{X}_{opt,2}$ with the side constraint $x_3 = Z > 5$. The cascade optimization process comprising the initial optimization towards $\overline{X}_{opt,2}$ and the subsequent perturbation towards $\overline{X}_{opt,3}$, $\overline{X}_{opt,4}$ was always completed by HALSGWJA within 8000 analyses, much less than the 10,000 to 20,000 analyses required by the algorithms of Refs. [14,70,75,84].

In the robot gripper design problem 10, HALSGWJA converged to the best optimized cost of 2.5437852, which is larger than the 2.528792 target value listed in Table 1, but perfectly consistent with the best solution of 2.5438 reported in the literature for the CEC2020 winners. Similarly, the best solution obtained by HALSGWJA for problem 16 (tension/compression spring design case 2), corresponding to a minimum steel wire volume of 2.658722, was different from the 2.61388 target volume of Table 1 but totally consistent with all other solutions reported in the literature for this test case that always ranged from 2.658558 to 2.684243 (see Table 2). Remarkably, HALSGWJA was also the fastest optimizer in these two test cases.

In test problems 13 (ten-bar truss design) and 18 (Himmelblau's function), HALSGWJA's best optimized cost was, respectively, 524.5885 vs. the 524.4508 target (only 0.0263% cost penalty) and −30,664.873 vs. the −30,665.539 target (only 0.00217% cost penalty). Again, HALSGWJA ranked first for computational speed in both of these problems.

The optimized designs found by HALSGWJA in the CEC2020 problems are reported in Appendix A. In summary, the present algorithm practically never missed the target optimal design in any test problem, as the largest deviation from target optimum was only 0.0263% in the ten-bar truss design problem 13. Such excellent performance was achieved only by EnMODE [54,65,68] and COLSHADE [55,65,68], remarkably the two winners of the CEC2020 real-world mechanical engineering competition. Conversely, En(L)SHADE [56], IMODE-KG [57], DESS [58], SASS [53,59,68], and MSFPSO [62] missed the target optimum only in problem 12 on the weight minimization of a four-stage gearbox; their best optimized weights ranged between 36.2504 and 98.69 vs. the 35.359 target optimum (cost penalty hence ranged between 2.52% and 179%). MPSGO [67], MAHA [68], and EJAYA [57,70] also missed the target optimum only in one test problem: (i) MPSO in problem 11 (optimized cost of 1643.1 instead of the 1616.120 target, 1.7% cost penalty); MAHA and EJAYA, in problem 10, respectively, obtained optimized costs of 2.638958 and 2.601873 vs. the actual best cost of 2.5438 reported in the CEC2020 literature (cost penalty hence ranged between 2.28% and 3.74%). SSDS-ABC [59] missed the target optimum in two test problems: specifically, 4 (5979.985 vs. the 5885.333 target, 1.61% cost penalty) and 12 (43.14295 vs. the 35.359 target, 22.02% cost penalty), with a maximum penalty on optimized cost of 22.02%.

KOA/CGKOA [57,61] missed the target optimum in three test problems: specifically, 11 (2403.717 vs. the 1616.120 target, 48.73% cost penalty), 12 (36.6012 vs. the 35.359 target,

3.51% cost penalty), and 13 (529.4834 vs. the 524.45 target, 0.96% cost penalty), with a maximum penalty on optimized cost of 48.73%. IDMO [64] also missed the target design in three test problems: specifically, 2 (0.033960 vs. the 0.032213 target, 5.42% cost penalty), 11 (1640.9 vs. the 1616.120 target, 1.53% cost penalty), and 12 (37.461 vs. the 35.359 target, 5.945% cost penalty), with a maximum penalty on optimized cost of 5.945%. IYDSE [57,60] missed the target design in four test problems: specifically, 2 ($3.29 \cdot 10^{-2}$ vs. the 0.032213 target, 2.13% cost penalty), 10 (2.70 vs. the 2.5438 target, 6.14% cost penalty), 11 (2483.159 vs. the 1616.120 target, 53.65% cost penalty), and 13 (535.9481 vs. the 524.45 target, 2.19% cost penalty), with a maximum penalty on optimized cost of 53.65%. ESAO [63] also missed the target design in four test problems: specifically, 8 (0.53571 vs. the 0.52577 target, 1.89% cost penalty), 10 (2.5929 vs. the 2.5438 target, 1.93% cost penalty), 11 (1918.4 vs. the 1616.120 target, 18.7% cost penalty), and 12 (41.697 vs. the 35.359 target, 17.9% cost penalty), with a maximum penalty on optimized cost of 18.7%.

CSELGWO [66] missed the target average solution in test problems 2 and 8, with a maximum penalty on the average optimized cost of 1.077%, while LMWOAGWO [65] obtained worse average costs than the corresponding target values in test problems 2, 4, 9, 11, 13, and 16, with a maximum penalty on the average optimized cost of 24.5%. Data on best and worst optimized solutions were not given in Refs. [65,66].

It should be noted that the best optimization runs of SDDS-SABC were reported in [59] to obtain, in several test problems, significantly better solutions than the target ones listed in Table 1. The most relevant improvements would have occurred for (i) test problem 9, a step-cone pulley design (12.30213 vs. 16.07, that is 25.139% reduction); (ii) test problem 5, a welded beam design (1.452753 vs. 1.670218, that is 13.02% reduction); (iii) test problem 3, a tension–compression spring design (case 1) (0.0112423 vs. 0.012665, 11.233%); (iv) test problem 7, a multiple disk clutch brake design (0.21881 vs. 0.23524, that is 6.954% reduction); (v) test problem 15, a gas transmission compressor design (2,851,884.4 vs. 2,964,895.4, that is 3.812% reduction); and (vi) test problem 1, a speed reducer design (2963.911 vs. 2994.425, that is 1.019% reduction). However, these designs were not provided, and their feasibility was not verified in Ref. [59]. Furthermore, the worst optimized designs of the SDDS-SABC independent runs for test problems 3, 5, 7, 9, and 15 were worse than those obtained by HALSGWJA. The average optimized designs of SDDS-SABC in test problems 7 and 15 were also worse than those found by the present algorithm. Hence, SDDS-SABC was significantly less robust than HALSGWJA.

IYDSE [57,60] was reported to obtain a significantly better solution than the target one in the four-stage gearbox design problem 12: only 14.9 vs. 35.359. However, the corresponding design and constraint margins (to check for any violations) were not specified in the above-mentioned references. Furthermore, the largest optimized cost and standard deviation on optimized cost recorded in the independent optimization runs were significantly worse than those found by HALSGWJA. IYDSE also found a slightly better design than the target solution in the rolling element bearing problem 14: 16,981.126 vs. 16,958.2 (also obtained by HALSGWJA). While the optimized cost of 16,981.126 reported in Ref. [57] for IYDSE improved the suboptimal cost of $5.60 \cdot 10^3$ originally reported in Ref. [60] by a great extent, no evidence on the feasibility of the 16,981.126 solution was given in [57]. Furthermore, the reported average and worst optimized costs of 17,045.036 and 17,203.713 were not consistent with the fact that problem 14 is a maximization problem.

CGKOA [61] was reported to obtain a much lower structural weight than the target solution in the step-cone pulley design problem 9: only 8.5633 vs. 16.0699. The corresponding optimal design of CGKOA was not given in [61]. A deeper investigation into the optimized solutions of problem 9 reveals that the best feasible design ever reported in the literature has the minimum weight of 8.18 and was obtained in 2024 by the Secretary Bird Optimization Algorithm (SBOA) developed in Ref. [82]; the design vector for this solution

was $d_1 = 17$ mm, $d_2 = 28.3$ mm, $d_3 = 50.8$ mm, $d_4 = 84.5$ mm, and $w = 90$ mm, where $d_{1-2-3-4}$ and w , respectively, denote the pulley diameters and width. The best design obtained by HALSGWJA was $d_1 = 38.477$ mm; $d_2 = 53.101$ mm; $d_3 = 95.346$ mm; $d_4 = 158.332$ mm; and $w = 47.913$ mm, with a weight of 16.036, which is much closer to the classical solution of $d_1 = 38.4$ mm; $d_2 = 52.9$ mm; $d_3 = 70.5$ mm; $d_4 = 84.5$ mm; and $w = 90$ mm, yielding the target pulley structural weight of 16.097. In fact, since the cost function of problem 9 is expressed as $W = 7200 \times w \times \{d_1^2[1 + (750/350)^2] + d_2^2[1 + (450/350)^2] + d_3^2[1 + (250/350)^2] + d_4^2[1 + (150/350)^2]\}$, that is $W = 7200 \times w \times \{5.5918 d_1^2 + 2.6531 d_2^2 + 1.5102 d_3^2 + 1.1837 d_4^2\}$, cost function is mostly sensitive to the variations of d_1 and d_2 . Interestingly, by removing the $d_{1-2-3-4} > 35$ mm side constraints originally included in the HALSGWJA optimizations, the present algorithm was able to converge to the 8.18 minimum weight design quoted in Ref. [82]. This was accomplished within about 10,000 analyses, hence in a much faster way than the computational budgets of 30,000 to 50,000 analyses indicated in [59,61,82] for the structural weights of 8.18, 8.5633, and 12.30213, respectively, obtained by SBOA, CGKOA, and SDDS-SABC.

The best optimized costs found by MSFPSO [62] in test problems 11 (hydrostatic thrust bearing design) and 15 (gas compressor transmission design) appear significantly lower than the corresponding target values reported in Table 1: only 137.08396 vs. 1616.120 and only 1,227,929.15 vs. 2,964,895.4, respectively. MSFPSO also obtained a slightly better optimized cost than the target value (i.e., 6032.55 vs. 6059.714) in the pressure vessel design problem 4 variant solved with mixed variables. However, the corresponding optimized designs and constraint margins of problems 11, 15, and 4 were not indicated in [62] and hence it is not possible to verify the feasibility of these designs. Furthermore, in problem 4, MSFPSO obtained a higher cost in its worst optimization run and a larger standard deviation on optimized cost than the present algorithm. MAHA [68] slightly improved the target optimum cost of the four-stage gearbox problem, achieving a minimum weight of 34.63987 vs. the 35.359 target, but the corresponding optimal design and information on constraint margins were not provided in Ref. [68]. Furthermore, the average and worst optimized weights found by MAHA were, respectively, about 10.7% and 36.9% higher than those of HALSGWJA, thus indicating a significantly lower robustness of MAHA with respect to the present algorithm.

The relevance of the present results is highlighted by the fact that the 18 selected competitors of HALSGWJA were very efficient metaheuristic algorithms that either (i) won the CEC2020 competition for real-world mechanical engineering problems (i.e., EnMODE, COLSHADE, SASS); (ii) implemented improved or hybrid variants of the HALSGWJA's component algorithms GWO and JAYA (i.e., LMWOAGWO, CSELGWO, EJAYA) that effectively competed with the CEC2020 winners; (iii) were not based on GWO/JAYA but could, however, compete with the CEC2020 winners (i.e., En(L)SHADE, IMODE-KG, DESS, SDDS-ABC, MPSGO, MAHA); or (iv) outperformed highly cited methods like PSO, GWO, and WOA (i.e., IYDSE, KOA, GKOA, MSFPSO, ESAO, IDMO). Each selected competitor of HALSGWJA was, in turn, proven to outperform from 4 to 21 other metaheuristic algorithms.

For the sake of brevity, detailed comparisons between HALSGWJA and other GWO/JAYA variants and hybrid GWO-JAYA formulations that solved only some of the problems included in the CEC2020 library will not be made in this study. However, it should be noted that HALSGWJA significantly outperformed the Simple Hybrid Grey Wolf JAYA (SHGWJA) algorithm originally developed in 2024 by the present authors by hybridizing GWO and JAYA [6]. SHGWJA is the starting base of the HALSGWJA formulation, and it was proven in Ref. [6] to be very competitive (or even superior) against state-of-the-art metaheuristic algorithms, including CEC2020 winners, as well as with advanced

variants of GWO and JAYA, the component algorithms selected in [6] and here for developing the hybrid metaheuristic formulations. SHGWJA was tested in seven engineering problems, also including four indicative CEC2020 test problems: (i) industrial refrigeration system design problem 2; (ii) tension-compression spring design (case 1) problem 3; (iii) pressure vessel design problem 4 (both variants with continuous and mixed variables); and (iv) welded beam design problem 5. The following facts should be highlighted.

- In problem 2, both HALSGWJA and SHGWJA converged to the target optimum cost of 0.032213 but the present algorithm always completed the 20 independent optimization runs within only 4695 to 6258 analyses (the fastest optimization run of SHGWJA was completed within 5165 analyses), requiring, on average, only 5478 analyses (with a standard deviation of 1003 analyses) vs. the 8517 analyses required, on average, by SHGWJA (with a standard deviation of 2370 analyses). Furthermore, average and worst optimized costs and corresponding standard deviation were better for HALSGWJA being, respectively, 0.0322132 vs. 0.032215, 0.032216 vs. 0.032219, and 6.7082×10^{-7} vs. 3.671×10^{-6} .
- In problem 3, HALSGWJA converged to the target optimum cost of 0.012665 while SHGWJA converged to a slightly higher optimized cost, 0.0126665. The average and worst optimized costs and corresponding standard deviation recorded for HALSGWJA again were better than those of SHGWJA, being, respectively, 0.012668 vs. 0.012682, 0.012670 vs. 0.012692, and 2.1152×10^{-6} vs. 1.11×10^{-5} . Furthermore, HALSGWJA required fewer analyses than SHGWJA, respectively, only 2103 vs. 2247 for the fastest optimization run and only 3316 vs. 6347 on average (with a standard deviation of only 1137 vs. 2246 analyses).
- In both variants of problem 4, HALSGWJA and SHGWJA converged to the target optimum costs of 5885.331 (continuous problem) and 6059.714 (mixed problem), always reaching 0 standard deviation. However, HALSGWJA required fewer analyses than SHGWJA: (i) for the continuous problem, only 3510 vs. 6732 in the fastest optimization run, only 4672 vs. 9773 on average (with a standard deviation of only 1203 analyses vs. 2561 analyses); (ii) for the mixed problem, only 3852 vs. 7604 in the fastest optimization run, only 5104 vs. 9712 on average (with a standard deviation of only 1374 analyses vs. 2730 analyses). The number of analyses recorded for the slowest optimization runs of HALSGWJA was even lower than its counterpart recorded for the fastest optimization runs of SHGWJA: only 5713 vs. 6732 analyses for the continuous variables problem variant; only 6193 vs. 7604 analyses for the mixed variables problem variant.
- In problem 5, HALSGWJA and SHGWJA converged to the target optimum cost of 1.670218 with 0 standard deviation. HALSGWJA again required fewer analyses than SHGWJA: only 2975 vs. 3691 for the fastest optimization run and only 3224 vs. 4438 on average (with a standard deviation of only 285 analyses vs. 424 analyses). Remarkably, the slowest optimization run of HALSGWJA required fewer analyses than the fastest optimization run of SHGWJA: only 3691 vs. 3791 analyses.

Although the data presented in Table 2 are very composite as they followed from the various settings chosen for the optimization runs of the 18 metaheuristic methods compared with HALSGWJA, the proposed algorithm is clearly superior to its competitors in terms of its capability to converge to the optimal solutions of the CEC2020 real-world mechanical engineering problems. Such a superiority becomes even more evident if one considers the very small number of analyses required by HALSGWJA in the optimization process. In fact, Table 2 also shows the number of analyses required by the fastest optimization run of HALSGWJA, converging to the best optimized cost of each test problem. The present algorithm always completed the optimization process between only 591

(four-stage gear-box design problem 12) and 7400 (hydrostatic thrust bearing design problem 11) analyses vs. at least 10,000 analyses required by its fastest competitors, such as IMODE-KG [57], IYDSE [57,60], KOA [57,74], and EJAYA [57,70].

Interestingly, IMODE-KG converged in 10,000 analyses to worse designs than HALSGWJA in (i) problem 3 (tension/compression spring design case 1), with an optimized cost of 0.0126742 vs. the 0.012665 target optimum reached by HALSGWJA in only 2013 analyses; (ii) problem 8 (planetary gear-train design), with an optimized cost of 0.526581 vs. the 0.523250 optimum cost reached by HALSGWJA in only 2095 analyses; and (iii) problem 11 (hydrostatic thrust bearing design), with an optimized cost of 1783.954 vs. the 1616.120 optimum cost reached by HALSGWJA within only 7400 analyses.

IYDSE also converged within 10,000 analyses to worse designs than HALSGWJA in (i) problem 4 (pressure vessel design variant with mixed variables), with an optimized cost of 6103.368 vs. the 6059.714 target optimum reached by HALSGWJA in only 3852 analyses; (ii) problem 11, with an optimized cost of 2483.159 vs. the 1616.120 optimum cost reached by HALSGWJA in only 7400 analyses; (iii) problem 13, with an optimized cost of 535.5885 vs. the 524.5888 to 524.6111 optimized costs obtained by HALSGWJA within 1911 to 4801 analyses only; and (iv) problem 18, with an optimized cost of −30,660.325 vs. the −30,664.873 optimum cost reached by HALSGWJA within only 4694 analyses.

KOA also converged within 10,000 analyses to worse designs than HALSGWJA in (i) problem 1 (speed reducer design), with an optimized cost of 2995.351 vs. the 2994.425 target optimum reached by HALSGWJA within only 2326 analyses; (ii) problem 11, with an optimized cost of 2403.717 vs. the 1616.120 optimum cost reached by HALSGWJA within only 7400 analyses; (iii) problem 13, with an optimized cost of 529.4834 vs. the 524.5888 to 524.6111 optimized costs obtained by HALSGWJA within 1911 to 4801 analyses only; (iv) problem 15, with an optimized cost of 2,965,375.3 vs. the 2,964,893.937 optimum cost reached by HALSGWJA within only 4377 analyses; and (iv) problem 18, with an optimized cost of −30,663.305 vs. the −30,664.873 optimum cost reached by HALSGWJA within only 4694 analyses.

EJAYA (a very advanced JAYA variant) also converged within 10,000 analyses to worse designs than HALSGWJA in (i) problem 8, with an optimized cost of 0.527347 vs. the 0.523250 optimum cost reached by HALSGWJA within only 2095 analyses; (ii) problem 10, with an optimized cost of 2.601873 vs. the 2.5437852 optimum cost reached by HALSGWJA within only 2095 analyses; and (iii) problem 13, with an optimized cost of 525.0392 vs. the 524.5888 to 524.6111 optimized costs obtained by HALSGWJA within 1911 to 4801 analyses only.

Another proof of the very high computational efficiency of HALSGWJA is that the optimization data listed in Table 2 for the CEC2020 top algorithms EnMODE [54,65,68], COLSHADE [55,65,68], and SASS [53,59,68] were relative to the classical computational budgets of (i) 100,000 analyses in test problems 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, and 18, or (ii) 200,000 analyses in test problems 2, 12, and 19. In summary, the CEC2020 winners required at least one order of magnitude more analyses than the present algorithm to complete their optimizations.

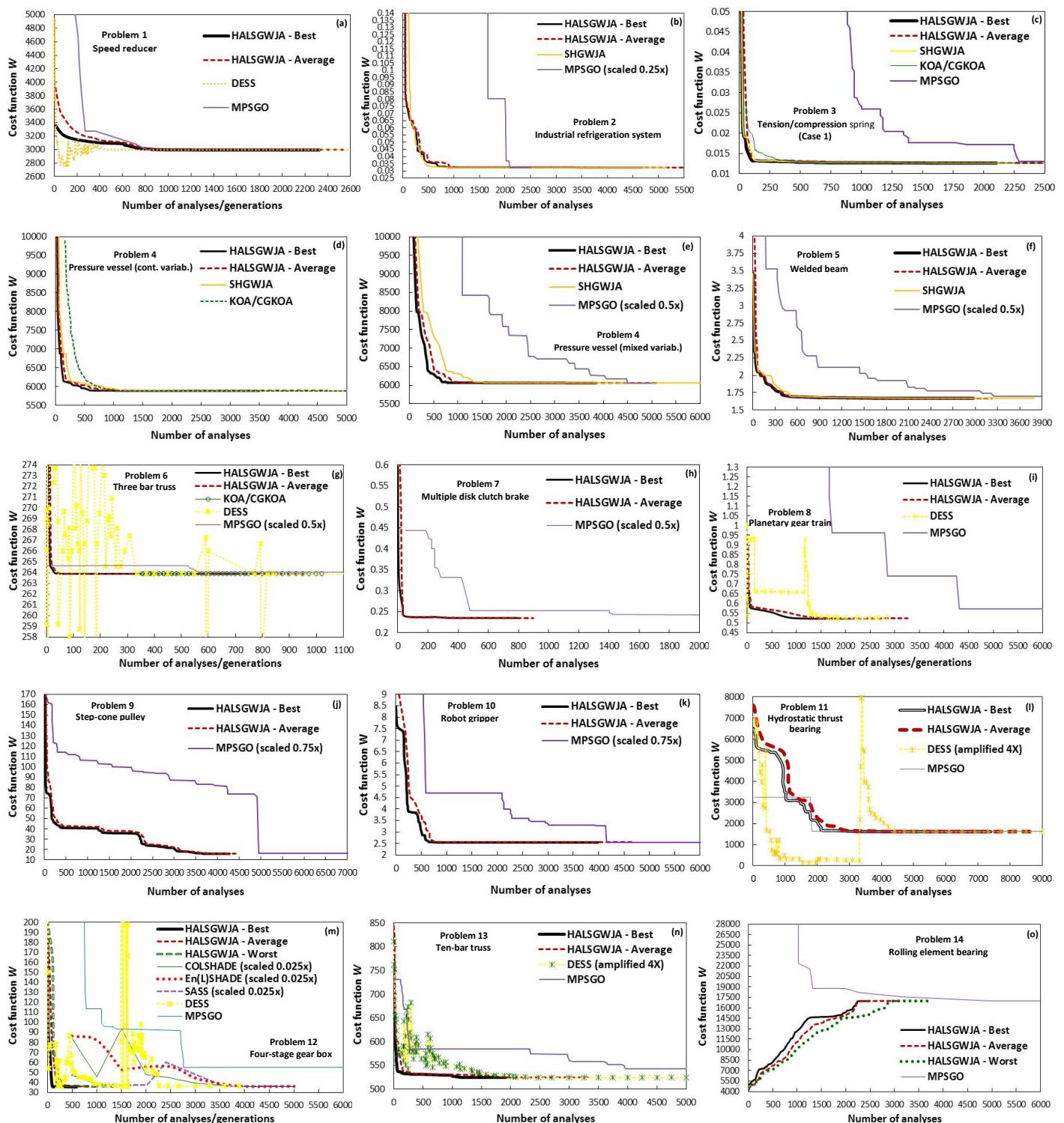
Whilst Table 2 reports the computational cost of the best and worst optimization runs as well as the average number of analyses and the corresponding standard deviation recorded for HALSGWJA, similar data were not available for its competitors, except for the information on the average number of analyses required in the optimizations given in Ref. [69] for the CEC2020 winners EnMODE, COLSHADE, and SASS. The data provided in [69] actually served to dispose of a relevant comparison basis for the Piranha Predation

Optimization Algorithm (PPOA) developed in [69]. While PPOA could converge to feasible designs only in 11 of the 19 test problems in spite of the theoretical computational budget of 1,000,000 analyses (requiring on average at least 26,900 analyses except for problems 1, 7, 17 and 19; however, PPOA was on average faster than HALSGWJA only in problem 17), according to the data reported in [69], EnMODE was on average faster than HALSGWJA in problems 1 (only 2277 analyses vs. 2538 analyses required by the present algorithm) and 17 (only 215 analyses vs. 1170 analyses required by the present algorithm). COLSHADE was on average faster than HALSGWJA in problems 17 and 18 (respectively, only 220 analyses vs. 1170 analyses, and only 2267 analyses vs. 4694 analyses). SASS was on average faster than HALSGWJA only in problem 17 (respectively, only 360 analyses vs. 1170 analyses). Since the number of analyses required by EnMODE, COLSHADE, and SASS in their best and worst optimization runs and the corresponding standard deviations were not specified in Ref. [69], it is not possible to directly compare the computational costs of EnMODE/COLSHADE/SASS evaluated in that study with those of HALSGWJA. In summary, the following aspects should be underlined for the results of Ref. [69]: (i) the CEC2020 winners converged in [69] to the target solutions of only 14–15 test problems, in open contrast with the documented results for their original implementations; (ii) the gap between HALSGWJA and each CEC2020 winner occurred only in one or two test problems out of 14–15 test problems successfully solved in [69]; (iii) the gap was just 11% in problem 1; (iv) the order of magnitude of the average number of analyses recorded in [69] for the CEC2020 winners always was comparable with its counterpart recorded for HALSGWJA.

HALSGWJA was also very robust in terms of optimized cost values and the required number of analyses in the independent optimization runs. In fact, HALSGWJA reached the 0 standard deviation on optimized cost in test problems 1, 4, 5, 6, 7, 11, 14, 15, and 19. For all other test problems, the ratio between the standard deviation in optimized cost and average optimized cost never exceeded 0.476%. Interestingly, HALSGWJA also achieved the lowest standard deviation overall in the test problem 12, yielding the above-mentioned largest standard deviation to the average optimized cost ratio of 0.476%. In fact, HALSGWJA obtained a standard deviation of only 0.1687 (optimized cost values from 35.29269 to 35.65141; all optimization runs required between 531 and 1591 analyses) vs. 0.49725 of SDDS-ABC [59] (optimized cost values from 43.14925 to 45.36397, within 20,000 analyses) and 0.56923 of MPSGO [67] (optimized cost values from 35.359 to 37.258, within 200,000 analyses). The CEC2020 winners EnMODE [54,65,68], COLSHADE [55,65,68], and SASS [53,59,68], respectively, obtained standard deviations on optimized cost equal to 0.59911 (i.e., 1.677% of the average optimized cost 35.728), 1.3677 (i.e., 3.736% of the average optimized cost 36.611), and 2.11484 (i.e., 5.491% of the average optimized cost 38.5141). MSFPSO [62], ESAO [63], and IDMO [64] exhibited the largest values of standard deviation on optimized cost for test problem 12, being, respectively, 126,054.2, 9.372×10^{14} (within 15,000 analyses), and 6.3419×10^{15} within 50,000 analyses. The data reported for problem 12 confirm the difficulty of many metaheuristic algorithms in converging to the target solution of this highly nonlinear problem, regardless of the selected initial population in each optimization run. Conversely, HALSGWJA also obtained the best standard deviation amongst all the algorithms compared in this study in this difficult problem, even disposing of a much lower computational budget.

As far as concerns HALSGWJA's robustness in terms of computational cost, the ratio between the standard deviation in required analyses and the average number of analyses required by the present algorithm never exceeded 38.97%, with an average of 18.07% and a median of 16.7%. The robust convergence behavior of HALSGWJA is confirmed by Figure 3, which shows the optimization histories recorded in all CEC2020 problems for the present algorithm and its competitors. It can be seen that the convergence curves of best

and average optimization runs of HALSGWJA practically coincided on average at about 49.2% of the optimization history of the best run. The overlap of HALSGWJA's best and average convergence curves occurred after completing more than 75% of the best run optimization history only in four cases: the planetary gear train design problem 8 (i.e., 76.4%); the step-cone pulley problem 9 (i.e., 81.8%); the ten-bar truss design problem 13 (i.e., 78.5%); and the gear train design problem 17 (i.e., 87.2%, to reach cost function values below the 10^{-15} double precision limit). However, this was well balanced by four other cases where the overlap occurred before completing 25% of the best run optimization history: industrial refrigeration system problem 2 (i.e., 23.4%); three-bar truss design problem 6 (i.e., 8.51%); four-stage gearbox design problem 12 (i.e., 16.9%); and gas transmission compressor problem 15 (i.e., 11.4%).



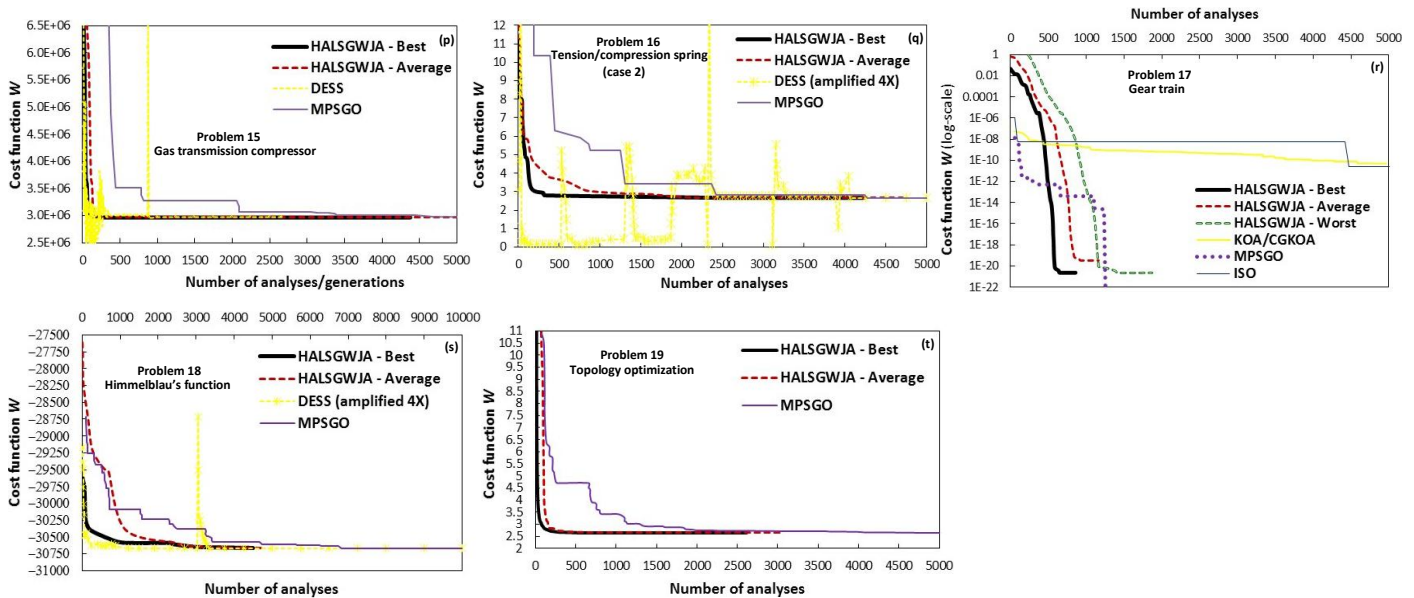


Figure 3. Convergence curves recorded for HALSGWJA and its competitors (SHGWJA [6], COLSHADE [55,65,68], En(L)SHADE [56], DESS [58], SASS [53,59,68], KOA/GKOA [57,61,74], MPSGO [67], ISO [79]) in the CEC2020 real-world mechanical engineering problems. Each sub-illustration included in (a–t) refers to a test case of the CEC2020 library. In the vertical axes of each sub-illustration's plot, the “,” notation indicates the decimal signs.

Figure 3 clearly shows the superior convergence behavior of HALSGWJA with respect to its competitors. It should be noted that convergence curves for the whole CEC2020 real-world mechanical engineering problems library were available only for the DESS [58] and MPSGO [67] algorithms. However, since some convergence curves presented in Ref. [58] for the DESS algorithm either were out of scale or did not have enough resolution, they are not included in Figure 3 for the sake of clarity. Since the optimization histories of DESS and MPSGO were given in Refs. [58,67] with respect to the number of generations (also indicated as optimization iterations), while Figure 3 referred the curves to the number of analyses, the data of Refs. [58,67] were converted as follows.

DESS is a differential evolution-based algorithm that linearly reduces the population size from the initial value $6 \times NDV^2$ as generations progress and distributes search agents over four sub-populations. Hence, in each generation, DESS would perform at least four new analyses if each sub-population included at most one search agent. Interestingly, in test problems 1, 6, 8, 12, and 15 (see Figure 3a,g,i,m,p), the HALSGWJA algorithm developed in this research directly converged to the target optima within fewer analyses than the generations required by DESS; hence, even in the ideal scenario where DESS performed only one new analysis per generation, it would, however, require more analyses than HALSGWJA. The number of generations of the DESS convergence curves shown in Figure 3l,n,q,s for test problems 11, 13, 16, and 18 was instead amplified by a factor of 4; the corresponding DESS curves were denoted as “amplified 4X” in the figure. This arrangement made the comparison of convergence curves more realistic. In Figure 3, the horizontal axis of convergence curves is hence labeled as “Number of analyses/generations” if the actual number of DESS generations is considered, or labeled as “Number of analyses” if the number of DESS generations is amplified.

It should be noted that the selected amplification factor of 4 for DESS generations was, however, much smaller than the initial population of $6 \times NDV^2$ search agents used by DESS in its first generation, being, respectively, 96 (i.e., 6×4^2), 600 (i.e., 6×10^2), 54 (i.e., 6×3^2), and 150 (i.e., 6×5^2) for test problems 11, 13, 16, and 18. It can be seen from Figure

3l,n,q,s that DESS practically converged to the optimal solutions of test problems 11, 13, 16, and 18 within 1100, 555, 1000, and 800 generations, respectively; hence, by linearly decreasing population size from the corresponding values 96, 600, 54, and 150, the computational cost of DESS would be about 63,850, 75,270, 59,230, and 60,050 analyses, respectively, vs. only 7400, 1911, 4223, and 4494 analyses required by HALSGWJA.

Unlike HALSGWJA, which monotonically converged to the optimum solution in all test problems, DESS always exhibited a significantly oscillatory behavior often characterized by the presence of infeasible intermediate designs. This occurred because the spherical search operator implemented in DESS explores a spherical region around each population member using gradient repair to guide the solutions towards the feasible region, while HALSGWJA always relates approximate gradient information to the best candidate solutions currently stored in the population. The JAYA-based perturbation schemes and mirror strategies always force the present algorithm to generate high-quality trial designs and move in the opposite direction to any perturbation that leads to an increasing cost function.

The number of analyses relative to the MPSGO algorithm [67] was determined by multiplying the population size $N_{POP} = 50$ indicated in Ref. [67] by the corresponding number of optimization iterations. This was done because MPSGO hybridizes the Marine Predators Algorithm (MPA) and the Social Group Optimization (SGO) algorithm, and both component algorithms perform N_{POP} analyses in each optimization iteration, evaluating the cost function and constraints of all search agents. Interestingly, MPSGO was reported in [67] to always practically converge to the optimal solutions of CEC2020 real-world mechanical engineering problems within 100–200 optimization iterations. Hence, the number of analyses covered in Figure 3 for MPSGO's convergence curves ranged between 5000 and 10,000. To make convergence curves more readable, the number of analyses of MPSGO was scaled by (i) 0.25 in the case of test problem 2 (see Figure 3b); (ii) 0.5 in the case of test problems 4 (see Figure 3e for the mixed variable problem variant), 5 (see Figure 3f), 6 (see Figure 3g), 7 (see Figure 3h), and 9 (see Figure 3j); and (iii) 0.75 in the case of test problem 10 (see Figure 3k). The scaled optimization histories of MPSGO are, respectively, denoted as "scaled 0.25×", "scaled 0.5×", and "scaled 0.75×" in Figure 3.

It can be seen that MPSGO was much slower than HALSGWJA in all test problems. This occurred even when MPSGO's convergence curves were scaled. Furthermore, MPSGO's optimization histories always presented several steps yielding marginal improvements in the cost function. This occurred over the whole optimization process. Such a behavior may be explained as follows. MPSGO divides the optimization search process into five phases: an inspiration phase dominated by SGO, where search agents try to imitate the best candidate design of the population; Brownian exploration dominated by MPA; a self-awareness phase (typical of modified SGO), where the search process shifts from exploration to exploitation; combined exploration and exploitation, characterized by Levi's and Brownian random walks; and a high exploitation phase, where the optimizer focusses on local search to converge to the global optimum. However, two facts should be considered. Firstly, the five phases of MPSGO mentioned above all cover the same fraction of the search process, 1/5. Secondly, MPSGO adopts the classical approach of metaheuristic optimization: exploring first and exploiting later. Conversely, HALSGWJA dynamically mixes exploration and exploitation based on the current trend of optimization search. This allows HALSGWJA to monotonically converge to the optimum. Furthermore, the line search strategies implemented in HALSGWJA allow the regions of design space hosting the best solutions, and in all likelihood the global optimum, to be approached quickly.

Although Figure 3l might indicate that MPSGO converged to the optimal solution of the hydrostatic thrust bearing problem 11 before the present algorithm, it should be noted

that MPSGO obtained a suboptimal solution with a higher cost than the target optimum cost achieved by HALSGWJA: 1643.1 vs. only 1616.120. HALSGWJA recovered the initial gap in cost function with respect to MPSGO (i.e., about 6520 vs. about 3251) within the first 1100 analyses and continued to smoothly reduce the cost function until reaching the target optimum cost at about 3000 analyses. Conversely, MPSGO did not change the cost function significantly over the first 1815 analyses and then drastically reduced W between 1816 and 1836 analyses, finally remaining trapped at the local minimum, 1643.1. In the rolling element bearing problem 14 (see Figure 3o), MPSGO converged to the target optimum cost of 16,958, always reducing the cost function, while HALSGWJA progressively increased it until reaching the target solution. The latter behavior certainly was more consistent with the nature of problem 14, which aimed to maximize the cost function. Interestingly, MPSGO was not even able to generate any better intermediate design than those elaborated in the worst optimization run of HALSGWJA.

Some representative convergence curves plotted in Figure 3 were also available for (i) COLSHADE [55,65,68], En(L)SHADE [56], and SASS [53,59,68] in the case of the four-stage gearbox problem 12 (see Ref. [56]); (ii) KOA [57,74] and CGKOA [61] in the case of test problems 3 (tension/compression spring design case 1), 4 (pressure vessel design with continuous variables), 6 (three-bar truss design), and 17 (gear train design); (iii) SHGWJA [6] in the case of test problems 2 (industrial refrigeration system), 3 (tension/compression spring case 1), and 4 (pressure vessel design with continuous or mixed variables); and (iv) the Attributes-based Information-learning Search Optimization (ISO) algorithm (mimicking people's psychological assessment and learning behaviors for information resources, emphasizing psychological assessment, learning and recovering for information loss) (see Ref. [79]) in the case of the gear train design problem 17.

Figure 3m compares the convergence curves of the four-stage gearbox problem 12, the test case of CEC2020 real-world mechanical engineering library, including the largest number of optimization variables (22) and nonlinear constraints (86). The optimization histories reported in the literature for COLSHADE [55,65,68], En(L)SHADE [56], and SASS [53,59,68] were scaled in this study by a factor of 0.025 to fit their computational budget of 200,000 analyses into the plot of Figure 3m. Remarkably, the CEC2020 winner COLSHADE [55,65,68] converged to the target optimum cost of 35.359 within 3990 scaled analyses, while the best optimization run of the HALSGWJA algorithm developed here converged to the best cost overall of 35.294 within only 591 analyses. Furthermore, the worst optimization run of HALSGWJA required about 470 analyses only to find a cost function value of 36.265, similar to (i) the 36.2 optimized cost value finally reached by DESS within about 2265 generations (DESS actually obtained the 36.2 cost function value already after about 200, 245, 275, 1495, 1540, 1610, and 1630 generations; however, it always exhibited oscillatory behavior and shifted between feasible and infeasible intermediate designs in all these phases of optimization history); (ii) the 36.2 optimized cost value finally reached by En(L)SHADE within about 3990 scaled analyses; and (iii) the 36.25 optimized cost value finally reached by SASS within about 3575 scaled analyses. The superior convergence behavior of HALSGWJA is highlighted by the fact that COLSHADE, En(L)SHADE, and SASS could never find better intermediate designs than the present algorithm over the whole optimization history.

The best optimization runs of KOA [57,74] and CGKOA [61], shown in Figure 3c,d,g for problems 3, 4 (solved with continuous variables), and 6, were even outperformed by the average optimization runs of HALSGWJA when the present algorithm started its search from better designs than KOA/CGKOA. In the gear train design problem 17 (see Figure 3r), all optimization runs of HALSGWJA started from initial designs that were significantly worse than their counterparts generated for the KOA/CGKOA optimizations (i.e., the initial cost function values of HALSGWJA ranged from 0.0464 to 2.046 vs. the

initial cost of only 1.4×10^{-8} recorded for KOA/GKOA); the ISO algorithm [79] also started from a very low cost value, only 10^{-6} . However, the present algorithm required between 435 and 900 analyses only to recover such significantly large initial gaps in cost function and generate better intermediate designs than those of KOA/CGKOA and ISO. The multiple strategies implemented in KOA and CGKOA to balance exploration and exploitation, increase population diversity, eliminate poor quality individuals, accelerate the output of high-quality population, and explore elite groups from multiple perspectives were overcome by the inherent capability of HALSGWJA to generate new trial designs that are always likely to improve the current best record.

It should be noted that in the test problem 17, ISO [79] outperformed the other 12 state-of-the-art optimizers, including HALSGWJA components (i.e., GWO), components of CEC2020 winners (i.e., differential evolution), highly cited methods (i.e., particle swarm optimization, whale optimization algorithm, genetic algorithms, harmony search optimization), and a variety of other techniques (i.e., bee swarm algorithm, equilibrium optimizer, etc.). ISO was selected for comparison because its algorithmic structure is somewhat similar to GWO; there are three leading individuals that hierarchically share information with the rest of the population such as occurs in GWO, where wolves α , β , and δ guide the other wolves towards the prey. However, gradient information and line search made HALSGWJA definitely more effective than ISO in avoiding stagnation (and the presence of large steps in the optimization history) and repairing trial solutions that either may not improve the current best record or improve $\overline{X}_{best}$ just marginally.

Interestingly, Figure 3b–f shows that SHGWJA [6] was rather competitive with HALSGWJA in test problems 2, 3, 4, and 5. In fact, its convergence curves were much closer than those of MPSGO and KOA/CGKOA to the corresponding optimization histories recorded for the present algorithm. Such a behavior occurred because, similar to HALSGWJA, the SHGWJA algorithm hybridized GWO and JAYA, utilizing elitist strategies to accept/reject trial designs. This confirms the inherent efficiency of generating new trial designs that maximize the probability of improving the current best record.

3.2. Performance Ranking of HALSGWJA and Its Competitors in the CEC2020 Problems

HALSGWJA and its 18 competitors were also ranked with respect to five performance indicators: respectively, the best, average, and worst optimized cost values obtained in the independent optimization runs; the corresponding standard deviation on optimized cost (*STD*); and the number of analyses (*No. analyses*) required in the optimization process. Table 3 lists the score obtained by HALSGWJA for each performance category in the 19 CEC2020 real-world mechanical engineering problems solved in this study; “1” and “19”, respectively, denote that HALSGWJA was the best or the worst algorithm in a specific category. The table also reports how far HALSGWJA was from its best competitors when it could not rank amongst the top four optimizers (i.e., when HALSGWJA’s performance did not fall within the first quartile, defined in the present case by $4/19=0.2105$, 21% of the compared algorithms) for some of the performance indicators listed above.

Table 3. HALSGWJA ranks obtained in the CEC2020 mechanical engineering problems.

Test Problem	Best	Average	Worst	STD	No. Analyses
Speed reducer	1	1	1	1	1
Industrial refrigeration system design	1	7 (0.0322132 vs. 0.032213)	5 (0.0322160 vs. 0.032213)	6 (6.7082 $\times 10^{-7}$ vs. 0)	1

		7	5	8	
Tension/compression spring design (1)	1	(0.012668 vs. 0.012665)	(0.012670 vs. 0.012665)	(2.1152×10^{-6} vs. 0)	1
Pressure vessel design	1	1	1	1	1
Welded beam design	1	1	1	1	1
Three-bar truss design	1	1	1	1	1
Multiple disk clutch brake design	1	1	1	1	1
Planetary gear train design optimization	1	1	1	4	1
Step-cone pulley problem	1	1	1	10 (4.6437×10^{-3} vs. 0)	1
Robot gripper problem	1	1	1	6 (2.8263×10^{-7} vs. 0)	1
Hydrostatic thrust bearing design	1	1	1	1	1
Four-stage gearbox problem	2	2	1	1	1
	12	10	7	7	
Ten-bar truss design	(524.5885 vs. 522.9644)	(524.6004 vs. 523.0724)	(524.6111 vs. 524.45)	(0.0113519 vs. 0)	1
Rolling element bearing	2	2	2	1	1
Gas transmission compressor design	2	1	1	1	1
Tension/compression spring design (2)	12 (2.658722 vs. 2.658558)	6 (2.659550 vs. 2.658559)	5 (2.660999 vs. 2.658559)	7 (9.4671×10^{-4} vs. 0)	1
Gear train design	13 (2.3078×10^{-21} vs. 0)	8 (3.5365×10^{-21} vs. 0)	6 (4.9608×10^{-21} vs. 0)	7 (1.1169×10^{-20} vs. 0)	1
Himmelblau's function	14 (−30,664.873 vs. −30,665.54)	15 (−30,664.848 vs. −30,665.54)	10 (−30,664.834 vs. −30,665.53)	13 (0.0195687 vs. 0)	1
Topology optimization	2	2	1	1	1

It can be seen from Table 3 that in the nineteen CEC2020 real-world mechanical engineering problems, HALSGWJA ranked first overall (i) 11 times for best optimized cost, (ii) 10 times for average optimized cost, (iii) 12 times for worst optimized cost, (iv) 10 times for standard deviation on optimized cost, (v) 19 times for required number of analyses. Furthermore, HALSGWJA ranked second overall four times for the best optimized cost, three times for the average optimized cost, one time for the worst optimized cost. Lastly, HALSGWJA ranked fourth overall another time for the standard deviation on optimized cost.

In summary, HALSGWJA performed very well in the CEC2020 real-world mechanical engineering test suite because it ranked in the first quartile 15, 13, 13, and 11 times, respectively, for the best/average/worst solutions, standard deviation on optimized cost, and it was always the fastest optimizer in all test problems. It should be noted that even when HALSGWJA ranked below the fourth position in some category, its performance should be considered satisfactory. In fact, while HALSGWJA ranked 12th in test problems 13 (10 bar truss design) and 16 (tension/compression spring design case 2) and 13th in test problem 17 (gear train design) and 14th in test problem 18 (Himmelblau's function) in

terms of cost function's best values, the best solutions found by the present algorithm were only 0.311% (problem 13), 0.00617% (problem 16), and 0.00218% (problem 18) worse than those found by the top algorithms for this item, and the best cost of 2.3078×10^{-21} found by HALSGWJA in problem 17 was much below the double precision limit of 10^{-15} . Furthermore, while HALSGWJA ranked between the 6th and 15th position in problems 2 (industrial refrigeration system), 3 (tension/compression spring design case 1), 13, 16, and 18 in terms of average optimized cost, the gap with respect to best average solutions obtained by the top algorithms for this item only ranged between $6.209 \times 10^{-4}\%$ and 0.292%; the average cost of 3.5635×10^{-21} found by HALSGWJA in problem 17 again was much below the 10^{-15} double precision limit. Moreover, HALSGWJA ranked between the 5th and the 10th position in problems 2, 3, 13, 16, 17, and 18 in terms of worst optimized cost but the gap with respect to the top competitors for this item only ranged between $2.270 \times 10^{-3}\%$ and 0.0918%; the worst cost of 4.9608×10^{-21} found by HALSGWJA in problem 17 once again was much below the 10^{-15} double precision limit. Finally, while HALSGWJA ranked between the 6th and the 13th position in problems 2, 3, 9 (step-cone pulley), 10 (robot gripper), 13, 16, 17, and 18 in terms of standard deviation on optimized cost, either the standard deviation value recorded for HALSGWJA in problem 17 was lower than the 10^{-15} double precision limit, and the ratio between standard deviation on optimized cost and target optimum cost ranged from only $1.11 \times 10^{-5}\%$ to 0.0167% for problems 2, 3, 9, 10, 13, 16, and 18.

The very good performance of HALSGWJA is confirmed by Table 4, which shows the number of times that the present algorithm and its 18 competitors ranked first with respect to the selected performance indicators for each CEC2020 test problem. The table column labeled as "B+A+W" reports the sum of the scores relative to the best, average, and worst optimized costs categories. The "B+A+W+STD" column reports the sum of the scores relative to the best/average/worst optimized costs and the standard deviation on optimized cost categories. The "TOTAL" column reports the sum of the scores recorded for all performance indicators including also the number of analyses required in the optimization process. Remarkably, HALSGWJA achieved a total score of 62, ranking first overall ahead of En(L)SHADE [56], which totaled 60 points in this study and had been proven in [56] to be very competitive against CEC2020 winners. HALSGWJA also outperformed CEC2020 winners such as EnMODE [54,65,68], COLSHADE [55,65,68], and SASS [53,59,68] in view of its significantly higher computational speed that always allowed the present algorithm to converge to the target optima or at least to obtain very close solutions to target optima. Even considering Ref. [69], which reported the lower average values of analyses required by EnMODE, COLSHADE, and SASS with respect to HALSGWJA in problem 17 (i.e., 215, 220, and 360 vs. 1170) and for COLSHADE with respect to HALSGWJA in problem 18 (i.e., 2267 vs. 4694), the total score of HALSGWJA would be 60 points, remaining the best overall together with En(L)SHADE.

Table 4. Number of first rankings achieved by HALSGWJA and its competitors in the CEC2020 real-world mechanical engineering problems.

Algorithm	Best	Average	Worst	STD	No. Analyses	B + A + W	B + A + W + STD	TOTAL
HALSGWJA-Present	11	10	12	10	19	33	43	62
EnMODE [54,65,68]	14	13	14	3	0	41	44	44
COLSHADE [55,65,68]	13	12	13	11	0	38	49	49
En(L)SHADE [56]	16	14	16	14	0	46	60	60
IMODE-KG [57]	10	8	9	0	2	27	27	29
DESS [58]	14	13	N/A	N/A	0	27	27	27
SASS [53,59,68]	12	13	12	3	0	37	40	40

SDDS-SABC [59]	8	6	1	0	0	15	15	15
IYDSE [57,60]	10	5	5	0	0	20	20	20
KOA [57,74]—CGKOA [61]	10	8	6	2	0	24	26	26
MSFPSO [62]	12	9	7	4	0	28	32	32
ESAO [63]	11	5	N/A	1	0	16	17	17
IDMO [64]	12	7	N/A	0	0	19	19	19
LMWOAGWO [65]	N/A	8	N/A	0	0	8	8	8
CSELGWO [66]	N/A	9	N/A	0	0	9	9	9
MPSGO [67]	13	13	12	3	0	38	41	41
MAHA [68]	14	12	10	2	0	36	38	38
EJAYA [57,70]	10	5	3	0	1	18	18	19

Without considering standard deviation and computational speed, HALSGWJA ranked seventh overall after En(L)SHADE, EnMODE, COLSHADE, MPSGO, SASS, and MAHA with respect to the “B+A+W” score based on the cost function values obtained in the independent optimization runs. In particular, HALSGWJA achieved a score of 33 points vs. 36 to 46 points scores obtained by the above-listed algorithms. This occurred because HALSGWJA could neither rank first nor second overall in at least two of the best/average/worst optimized cost categories for test problems 2, 3, 13, 16, 17, and 18 (see also Table 3). In order to analyze this issue, Wilcoxon signed-rank tests with a significance level of 0.05 were executed for problems 2, 3, 13, 16, 17, and 18, comparing two datasets, including the best/average/worst optimized costs recorded for HALSGWJA and one competitor. As expected, HALSGWJA always achieved (i) positive signed-rank values with respect to En(L)SHADE, EnMODE, COLSHADE, MPSGO, SASS, and MAHA and (ii) negative signed-rank values with respect to the other competitors. However, since this Wilcoxon test operated on data pairs, each of which included only three values, another Wilcoxon signed-rank test was executed, taking the ensemble of best/average/worst optimized costs obtained in all nineteen CEC2020 problems. Hence, the new datasets included at most $19 \times 3 = 57$ values per pair of compared algorithms (i.e., the pairs formed by HALSGWJA and one competitor at a time). Since target optima significantly varied in magnitude, ranging from -3.07×10^4 to 2.965×10^6 , the best/average/worst optimized costs recorded for each test problem were normalized with respect to the corresponding target solutions listed in Table 1. Again, HALSGWJA achieved positive signed-ranks with respect to En(L)SHADE, EnMODE, COLSHADE, MPSGO, SASS, and MAHA only. These results were fully consistent with the “B+A+W” ranking stated in Table 4. The detailed results of the Wilcoxon signed-rank tests carried out in this study are not shown in the article for the sake of brevity.

HALSGWJA increased its rank to the fourth position in the “B+A+W+STD” score ranking after En(L)SHADE, COLSHADE, and EnMODE, with 43 points vs. 44 to 60 points. However, while the data related to EnMODE and COLSHADE were obtained by combining the optimization results of optimization reported in multiple studies, HALSGWJA’s performance evaluation relies only on the data gathered in the present study. Furthermore, the better performance of HALSGWJA’s competitors with respect to the “B+A+W” and “B+A+W+STD” scores was a direct consequence of their significantly higher computational budget. For example, in problem 2, En(L)SHADE, EnMODE, COLSHADE, MPSGO, SASS, and MAHA could complete their optimization runs within 200,000 analyses, while HALSGWJA completed all optimizations with between only 4695 and 6258 analyses. In order to gather further evidence of the HALSGWJA computational efficiency, another cycle of numerical trials was performed by restarting the optimization runs that previously missed the 0.0322130 target solution from initial populations of individuals in

the neighborhood of the suboptimal solutions. Remarkably, HALSGWJA always converged to the target optimum cost of 0.0322130 (thus also achieving also standard deviation on optimized cost) within a total of 9000 analyses for all repaired cases, thus remaining well below the number of analyses required by En(L)SHADE, EnMODE (even considering the average number of 9948 analyses indicated in [69] for this algorithm), COLSHADE, MPSGO, SASS, and MAHA in their optimization runs. Hence, HALSGWJA should be considered as also ranking first for the average/worst costs and standard deviation on optimized cost; this would increase its score by 2 and 3 points, respectively, in the “B + A + W” and “B + A + W + STD” categories. Similarly, in problem 3, HALSGWJA completed all primary optimization runs with between 2103 and 4097 analyses, and it then repaired all suboptimal runs, converging to the target optimum cost of 0.012665 within a total number of analyses of 6000, well below the computational cost of its competitors. Hence, HALSGWJA would again increase its “B+A+W” and “B+A+W+STD” scores by 2 and 3 points, respectively. By summing the above-mentioned improvements that could be achieved in problems 2 and 3, the “B+A+W” and “B+A+W+STD” scores of HALSGWJA would rise to 37 and 49, respectively. This would put the present algorithm at in fifth place (after En(L)SHADE, EnMODE, COLSHADE, and MPSGO, but at the same level as SASS) and third (after En(L)SHADE and COLSHADE) rank overall in the “B+A+W” and “B+A+W+STD” categories, respectively. The total score, including the computational cost, would instead rise to 68 (or 66 considering Ref. [69]), thus outperforming En(L)SHADE by 8 (or 6) points.

Besides the ranking analysis described above, the classical ranking-based Friedman test was executed for each performance indicator considered in this study: best/average/worst optimized costs, standard deviation on optimized cost, and number of analyses. The results of the Friedman test are presented in Table 5, which shows the average ranks achieved by HALSGWJA and its competitors over the 19 CEC2020 problems for each performance indicator, as well as the corresponding cumulated average rank Σ_{AR} (defined as the sum of average ranks for all performance indicators) and the mean cumulated average rank (defined as $\Sigma_{AR}/5$). The table also shows the partially cumulated average rank for the best/average/worst optimized cost categories (indicated as “*Partial Aver A+B+W*” in the table). For the sake of brevity, Table 5 shows only the data relative to the seven best algorithms—HALSGWJA, EnMODE [54,65,68], COLSHADE [55,65,68], En(L)SHADE [56], SASS [53,59,68], MPSGO [67], and MAHA [68]—that resulted from Table 4. Ranks were determined obviously considering HALSGWJA and its 18 competitors, increasing from “1” (best algorithm) to “19” (worst algorithm). Average ranks were obtained by dividing the sum of ranks obtained in the test problems by 19. It should be noted that the ranking-based Friedman test is usually carried out to rank algorithms with respect to their best or average optimized solutions. However, since HALSGWJA and its competitors always converged to target optima or very close solutions to target optima, it was decided to exhaustively compare algorithms by investigating in detail all aspects related to their robustness and computational cost.

Table 5. Results of ranking-based Friedman test carried out for all performance indicators.

Algorithm	Best	Average	Worst	STD	No. Analyses	Cumul. Aver. Rank	Mean Cumul. Aver. Rank $\Sigma_{AR}/5$
						Σ_{AR} (<i>Partial Aver B + A + W</i>)	
HALSGWJA-Present	3.684	3.632	2.947	4.105	1	15.368 (10.263)	3.074
EnMODE [54,65,68]	1.368	2.947	2.211	5.421	6.632	18.579 (6.526)	3.716
COLSHADE [55,65,68]	1.579	3.684	2.842	4.579	5.474	18.158 (8.105)	3.632
En(L)SHADE [56]	2.052	2.789	1.947	3.053	9.316	19.157 (6.788)	3.831

SASS [53,59,68]	2.842	3.421	2.947	5.421	7.263	21.894 (9.210)	4.379
MPSGO [67]	2.614	3.052	2.895	5.631	9.316	23.508 (8.561)	4.702
MAHA [68]	2.895	3.316	3.473	5.789	9.316	24.789 (9.684)	4.958

The results of the ranking-based Friedman test carried out in this study confirmed the mutual ranking of algorithms stated in Table 4. As far as concerns the best/average/worst optimized cost categories, the partially cumulated average rank increased from 6.526 for EnMODE and 6.788 for En(L)SHADE to 10.263 for HALSGWJA. However, the present algorithm obtained the second best average rank with respect to standard deviation on optimized cost (i.e., 4.105 vs. 3.053 of En(L)SHADE, the best algorithm for this category) and was the very best optimizer in terms of computational speed, achieving an average rank of 1 (i.e., the fastest algorithm in all test problems) vs. 5.474 obtained by COLSHADE (the second best algorithm for this category). Consequently, HALSGWJA obtained the lowest value of cumulated average rank Σ_{AR} : only 15.368 vs. 18.158 of COLSHADE, 18.579 of EnMODE, 19.157 of En(L)SHADE, 21.894 of SASS, 23.508 of MPSGO, and 24.789 for MAHA. In summary, HALSGWJA was confirmed to be the best optimizer overall and outperformed CEC2020 winners like COLSHADE, EnMODE, and En(L)SHADE; its mean cumulated average rank was only 3.074 vs. the 3.632 to 3.831 mean cumulated ranks obtained by the other algorithms. These four algorithms were superior to SASS, MPSGO, and MAHA, which ranked on average between 4.379 and 4.958.

3.3. Results of the Car Side Impact Problem

The optimization results obtained in the car side impact problem by the HALSGWJA algorithm developed in this study were compared with those obtained by 25 other state-of-the-art metaheuristic optimizers. HALSGWJA's competitors can be grouped in eight categories: (a) hybrid GWO-JAYA algorithms including elitist strategies such as the FHGWJA developed in [7]; (b) advanced GWO variants such as multi-strategy fusion improved gray wolf optimization (IGWO) [72], chaotic GWO, and adaptive dynamic self-learning grey wolf optimization algorithm (ASGWO) [73]; (c) advanced JAYA variants such as EJAYA [70]; (d) particle swarm optimization (PSO) [93] and comprehensive learning particle swarm optimizer (CLPSO) [94]; (e) efficient animal behavior-inspired methods such as cuckoo search optimization (CSO) [95], tunicate swarm algorithm (TSA) [96], starling murmuration optimizer (SMO) [76], improved continuous ant colony optimization algorithm (LIACOR) [97], krill herd algorithm (KH) [98], best variant of artificial bee colony (ABC) [99], whale optimization algorithm (WOA) [100], horned lizard optimizer algorithm (HLOA) [101], artificial lemming algorithm (ALA) [84], Harris hawks optimizer (HHO) [102], dwarf mongoose optimization (DMO) [103], starfish optimization algorithm (SFOA) [85], and firefly algorithm (FA) [87]; (f) efficient human behavior-inspired methods such as the attributes-based information-learning search optimization (ISO) algorithm [79] and the dream optimization algorithm (DOA) [83]; (g) efficient mathematics-inspired methods such as the exponential-trigonometric optimization algorithm (ETO) [81] and the arithmetic optimization algorithm (AOA) [104]; (h) high-performance optimizers such as the hybrid LSHADE-SPACMA algorithm [105], which combines history-based adaptive differential evolution with linear population size reduction and covariance matrix adaptation evolution strategy.

Table 6 reports the optimized designs and the corresponding structural weights obtained by HALSGWJA and its competitors. The solutions reported in the table were clustered according to those reported in the literature. In particular, IGWO, chaotic GWO, CSO, and TSA were compared in Ref. [72]; SMO, LIACOR, KH, Best ABC, CLPSO, and WOA were compared in Ref. [76]; DOA and HLOA were compared in Ref. [83]; ALA, HHO, AOA, DMO, and LSHADE-SPACMA were compared in Ref. [84]; and FA and PSO

were compared in Ref. [87]. The (X,Y) notation used in the table indicates the referenced study “X” proposing a main HALSGWJA’s competitor and the study “Y” from which it was originally taken the algorithm compared to the main algorithm developed in “X”. Similarly to the presentation of CEC2020 results, data reported in Table 6 for HALSGWJA’s competitors are relative to the best parameter settings indicated in the literature for each algorithm.

Table 6. Optimization results obtained by HALSGWJA and its competitors in the car side impact problem.

Algorithms	X1 (mm)	X2 (mm)	X3 (mm)	X4 (mm)	X5 (mm)	X6 (mm)	X7 (mm)	X8	X9	X10 (mm)	X11 (mm)	Structural weight (kg)
HALSGWJA Present	0.50000	1.21301	0.50000	0.77611	0.50000	1.48975	0.50000	0.345	0.345	−29.1035	0.00011	21.37798
FHGWJA [7]	0.50000	1.21204	0.50000	0.77908	0.50000	1.49004	0.50000	0.345	0.345	−28.9781	0.00010	21.38340
EJAYA [70]	0.50000	1.11631	0.50000	1.30228	0.50000	1.50000	0.50000	0.345	0.32680	−19.5709	0.00838	22.84297
IGWO [72]	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 [72,95]	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 [72,96]	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 [73]	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 [76]	0.5000	1.11634	0.5000	1.30224	0.5000	1.50000	0.5000	0.345	0.345	−19.566	0.000001	22.84298
LIACOR [76,97]	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 [76,98]	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 [76,99]	0.5000	1.30539	0.5000	1.10312	0.5000	0.50000	0.5000	0.345	0.345	14.213	20.3306	22.88605
CLPSO [76,94]	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 [76,100]	0.5000	1.09276	0.5000	1.41233	0.5000	1.45497	0.5000	0.345	0.192	−24.038	−3.1789	23.12717
ISO [79]	0.5000	1.11642	0.5000	1.30211	0.5000	1.5000	0.5000	0.345	0.345	−19.5525	−0.00152663	22.843
ETO [81]	0.50282	1.2414	0.51604	1.2201	0.60334	1.3878	0.5000	0.74832	0.06747	2.2526	−7.2818	23.2574
DOA [83]	0.5081	1.2021	0.5318	1.3052	0.5719	1.4954	0.5557	0.3030	0.2585	−24.8171	3.4047	23.9682
HLOA [83,101]	0.5000	1.0669	0.8016	1.0704	0.5040	1.4873	0.5000	0.192	0.192	−29.9786	3.2119	23.6956
ALA [84]	0.5000	1.1163	0.5000	1.3020	0.5000	1.5000	0.5000	1.0000	0.2391	−19.5747	−0.0022	22.84238
HHO [84,102]	0.5000	1.1137	0.5000	1.3065	0.5000	1.5000	0.5000	1.0000	0.5997	−20.0430	0.0010	22.84277
AOA [84,104]	0.5000	1.1179	0.5000	1.3005	0.5000	1.5000	0.5000	0.8525	0.1021	−19.2807	−1.3098	22.84756
DMO [84,103]	0.5000	1.1077	0.5000	1.3175	0.5000	1.5000	0.5000	0.6647	0.5054	−21.1276	−0.2616	22.84771
LSHADE-SPACMA [84,105]	0.5000	1.1059	0.5000	1.3213	0.5000	1.5000	0.5000	0.7355	0.9547	−21.4276	0.4018	22.84992
SFOA [85]	0.5000	1.2340	0.5000	1.1870	0.8750	0.8920	0.4000	0.345	0.192	1.5000	0.5720	23.56158
FA [87]	0.50000	1.36000	0.50000	1.20200	0.50000	1.12000	0.50000	0.345	0.192	8.87307	−18.9981	22.84298
PSO [87,93]	0.50000	1.11670	0.50000	1.30208	0.50000	1.50000	0.50000	0.345	0.192	−19.5494	−0.00431	22.84474

It can be seen from Table 6 that HALSGWJA converged to the best solution overall, corresponding to the lowest weight of 21.37798 kg. This design was slightly better than the 21.38340 kg obtained by FHGWJA [7] and the 21.39473 kg obtained by IGWO [72]. It should be noted that while the optimized designs of HALSGWJA and FHGWJA actually converged to the discrete value of 0.345 indicated in the problem statement for variables x_8 and x_9 , the corresponding optimal values found by IGWO and chaotic GWO did not coincide with the discrete values $\{0.192; 0.345\}$ set for this test problem. Instead, IGWO- and GWO-optimized designs became infeasible as x_8 was set equal to 0.192 or 0.345, violating constraints G_6 , G_7 , G_8 , and G_9 from 13.9 to 23.7%.

Chaotic GWO, CSO, and TSA also converged to very good structural weights, ranging between 21.46164 and 22.70296 kg, although their optimized solutions did not include the prescribed discrete values of variables x_8 and x_9 . Most of HALSGWJA’s competitors

obtained optimized weights between 22.84238 and 22.88605 kg, ranking as follows: ALA, HHO, EJAYA, SMO, FA, LIACO_R, ISO, PSO, AOA, DMO, LSHADE-SPACMA, ASGWO, KH, and Best ABC. Again, ALA, HHO, EJAYA, AOA, DMO, LSHADE-SPACMA, and ASGWO could not converge to the prescribed discrete values of x_8 and x_9 . The last six competitors obtained optimized weights between 23.06244 and 23.9682 kg, ranking as follows: CLPSO, WOA, ETO, SFOA, HLOA and DOA; ETO and DOA missed the target discrete values of x_8 and x_9 .

The optimized weights obtained by the 26 optimizers compared in Table 6 varied by 12%, from the lightest design overall being 21.37798 kg, obtained by HALSGWJA, to the heaviest design overall being 23.9682 kg, obtained by DOA. The high nonlinearity of this test problem and its particular formulation explain why the different algorithms found so many local optima. It should be noted that the cost function $W(\vec{X})$ stated by Equation (14) depends only on six optimization variables (i.e., x_1 , x_2 , x_3 , x_4 , x_5 , and x_7), while only x_{10} enters in all constraint functions (see Equation (15)). The search space of the car side impact problem therefore includes many competitive solutions for which optimized values of design variables may be significantly different. This is confirmed by the fact that in spite of recording only 12% variation in optimized cost amongst the various optimized solutions, the first nine variables listed in Table 6 for these solutions varied up to 30.4%, while x_{10} and x_{11} , respectively, varied by 1446.1% and 762.9% to satisfy all constraints. However, the optimization variables x_1 , x_2 , x_3 , x_4 , x_5 , and x_7 , included in the cost function $W(\vec{X})$, changed by 8.71%, 19.61%, 6.03%, 15.93%, 13.05%, and 5.88%, respectively, with an average dispersion of 11.54%. This was consistent with the 12% variation in optimized cost, considering that the cost function of the car side impact problem was fitted by a linear response surface. The inherent ability of HALSGWJA to generate high-quality trial designs that are always likely to improve the current best record allowed the present algorithm to avoid local minima.

Table 7 compares the best (indicated as “B–W” in the table), average (indicated as “A–W” in the table), and worst (indicated as “W–W” in the table) optimized weights and standard deviations on optimized weight (indicated as “W–W” in the table) obtained by HALSGWJA and its competitors in the different runs as well as the corresponding information on the number of structural analyses required in the optimization process (respectively, indicated as “B–NSA”, “A–NSA”, “W–NSA”, and “B–NSA” in the table). Statistical data on optimized weight were available only for FHGWJA [7], EJAYA [70], ISO [79], DOA, and HLOA, compared in Ref. [83], and SFOA [85], FA, and PSO, compared in [87]. No statistical data were available for the computational cost except for FHGWJA [7].

Table 7. Comparison of statistical data on optimized weight and computational cost for HALSGWJA and its competitors in the car side impact problem.

Algorithms	B–W (kg)	A–W (kg)	W–W (kg)	STD–W (kg)	B–NSA	A–NSA	W–NSA	STD–NSA
HALSGWJA Present	21.37798	21.37798	21.37798	0	1240	1291	1387	85
FHGWJA [7]	21.38340	21.3905	21.4145	0.007125	1367	1309	1606/1110	343
EJAYA [70]	22.84297	22.94398	23.26191	0.17098	N/A	27,000	N/A	N/A
IGWO [72]	21.39473	N/A	N/A	N/A	N/A	30,000	N/A	N/A
Chaotic GWO	21.46164	N/A	N/A	N/A	N/A	30,000	N/A	N/A
CSO [72,95]	22.00444	N/A	N/A	N/A	N/A	30,000	N/A	N/A
TSA [72,96]	22.70296	N/A	N/A	N/A	N/A	30,000	N/A	N/A
ASGWO [73]	22.87188	N/A	N/A	N/A	N/A	15,000	N/A	N/A
SMO [76]	22.84298	N/A	N/A	N/A	N/A	30,000	N/A	N/A
LIACO _R [76,97]	22.84299	N/A	N/A	N/A	N/A	30,000	N/A	N/A

KH [76,98]	22.88596	N/A	N/A	N/A	N/A	30,000	N/A	N/A
Best ABC [76,99]	22.88605	N/A	N/A	N/A	N/A	30,000	N/A	N/A
CLPSO [76,94]	23.06244	N/A	N/A	N/A	N/A	30,000	N/A	N/A
WOA [76,100]	23.12717	N/A	N/A	N/A	N/A	30,000	N/A	N/A
ISO [79]	22.843	22.8569	23.2612	0.0764	N/A	15,000	N/A	N/A
ETO [81]	23.2574	N/A	N/A	N/A	N/A	30,000	N/A	N/A
DOA [83]	23.9682	25.0673	25.9506	0.5111	N/A	2000	N/A	N/A
HLOA [83,101]	23.6956	28.4803	34.5482	2.5047	N/A	2000	N/A	N/A
ALA [84]	22.84238	N/A	N/A	N/A	N/A	15,000	N/A	N/A
HHO [84,102]	22.84277	N/A	N/A	N/A	N/A	15,000	N/A	N/A
AOA [84,104]	22.84756	N/A	N/A	N/A	N/A	15,000	N/A	N/A
DMO [84,103]	22.84771	N/A	N/A	N/A	N/A	15,000	N/A	N/A
LSHADE- SPACMA [84,105]	22.84992	N/A	N/A	N/A	N/A	15,000	N/A	N/A
SFOA [85]	23.56158	23.56160	23.56203	8.23×10^{-5}	N/A	50,000	N/A	N/A
FA [87]	22.84298	22.89376	24.06623	0.16667	N/A	20,000	N/A	N/A
PSO [87,93]	22.84474	22.89429	23.21354	0.15017	N/A	20,000	N/A	N/A

It can be seen from Table 7 that HALSGWJA always converged to the global optimum weight of 21.37798 kg in all optimization runs and completed the search process within the lowest number of structural analyses. The present algorithm required from 1240 to 1387 structural analyses to converge to the global optimum design with an average computational cost of only 1291 analyses. FHGWJA [7] was the second fastest optimizer, completing its best optimization run within 1367 structural analyses, just 10% more than the 1240 analyses required by the present algorithm. Interestingly, HALSGWJA was on average slightly faster than FHGWJA (1291 vs. 1309 analyses) but while the present algorithm always converged to the global optimum structural weight of 21.37798 kg in all runs, the latter algorithm converged two times to its worst optimized structural weight of 21.4145, completing these “worst” runs within 1110 and 1606 structural analyses, respectively. Such a large variation also explains why the average computational cost of FHGWJA (i.e., 1309 analyses) was lower than the computational cost recorded for the best optimization run of FHGWJA (i.e., 1367). As mentioned before, FHGWJA used elitist strategies and refined or repaired trial solutions while HALSGWJA directly used line search augmented by gradient information to generate trial designs. The average gradients defined by HALSGWJA as $\Delta W_{p(i,k)} / \Delta S^{(i,k)}$ in perturbing each design $\vec{X}_k$ stored in the population coincide with the actual gradients of a linear cost function if $\vec{X}_k$ is feasible. This property became very useful in the case of the car side impact problem, where the objective function was right-linear.

DOA and HLOA ranked third in computational speed, completing their optimizations within 2000 structural analyses, but obtained the worst designs overall, with structural weights of between 23.6956 and 23.9682 kg (see Ref. [83] for details). All other algorithms completed the optimization process within 15,000 and 50,000 analyses.

Besides achieving 0 standard deviation on optimized cost, HALSGWJA was very robust also in terms of computational cost. In fact, the ratio between standard deviation on the number of structural analyses and the corresponding average value required in the optimization runs was only 6.58%, in line with the best values obtained for the CEC2020 real-world mechanical engineering problems. The excellent convergence behavior exhibited by HALSGWJA is confirmed by Figure 4, which compares the optimization histories recorded for the proposed algorithm in the best, average, and worst runs with those of its

competitors FHGWJA [7], ISO [79], and ETO [81]. Since ISO and ETO, respectively, required 15,000 (i.e., the product of $N_{POP}=30$ by 500 iterations) and 30,000 (i.e., the product of $N_{POP}=30$ by 1000 iterations) structural analyses to complete their optimizations, the corresponding numbers of analyses were scaled by 0.1 and 0.05 to fit in the plot of Figure 4; similarly to in Figure 3, the notations “(scaled 0.1 ×)” and “(scaled 0.05 ×)” denote the optimization histories of ISO and ETO.

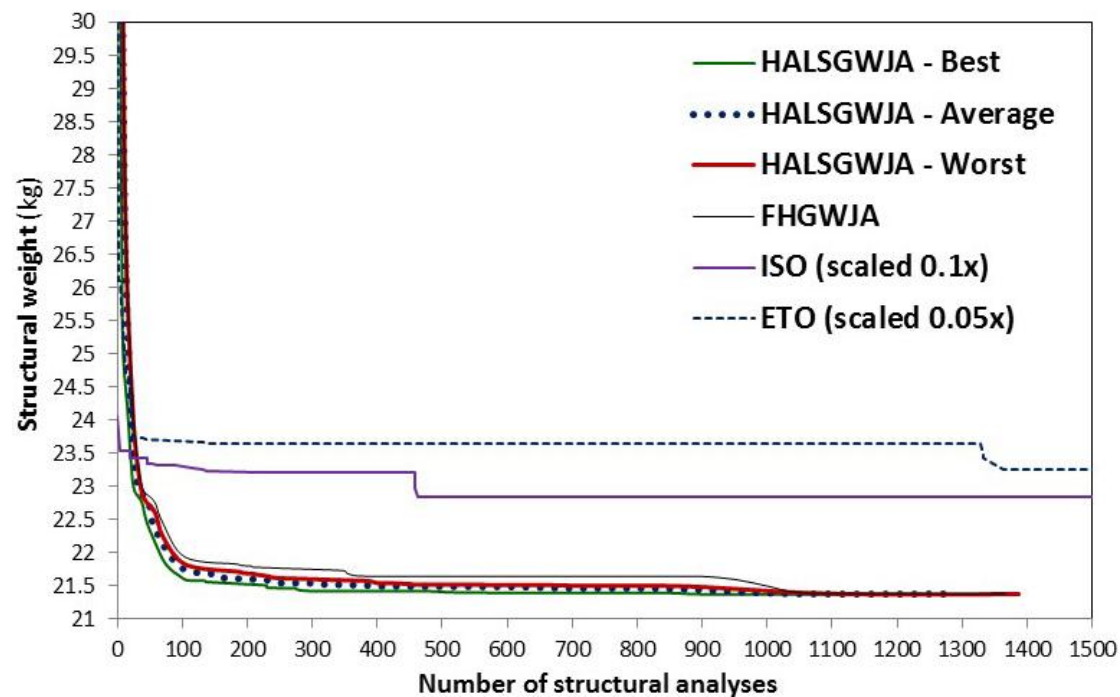


Figure 4. Convergence curves recorded for HALSGWJA and its competitors (FHGWJA [7], ISO [79], ETO [81]) in the car side impact problem. In the vertical axis of the plot, the “,” notation indicates the decimal signs.

The superiority of HALSGWJA over its competitors was evident in spite of the fact that the present algorithm started its optimizations from rather conservative designs. For example, the structural weight of the initial designs of HALSGWJA was 1.5 times larger than its counterpart for ISO (i.e., about 40 kg vs. about 24 kg) but the present algorithm always recovered this gap between only 20 and 35 structural analyses regardless of considering the best or worst optimization runs. ETO started its optimization search from a best search agent weighing about 24 kg but then reduced structural weight very slowly. Furthermore, HALSGWJA required, respectively, (i) less than 110 structural analyses to find a feasible intermediate design weighing about 21.6 kg, just 1% heavier than the global optimum weight of 21.37798 kg; (ii) only 17 structural analyses to find a feasible intermediate design weighing less than 23.9682 kg, the optimized weight of DOA; and (iii) only 18 structural analyses to find a feasible intermediate design weighing less than 23.6956 kg, the optimized weight of HLOA. The convergence curves of the best and average optimization runs of HALSGWJA practically coincided after only 1000 structural analyses, about 80% of the best run’s optimization history. Remarkably, the worst optimization run of HALSGWJA always generated better intermediate designs than their counterparts generated in the best optimization runs of FHGWJA, which was the only algorithm to be close enough to HALSGWJA in terms of convergence speed.

Similarly to what was observed in the case of CEC2020 real-world mechanical engineering problems, HALSGWJA outperformed its competitors in terms of its ability to converge to the global optimum in a very fast and robust way.

4. Conclusions

This study presented the novel hybrid metaheuristic algorithm, HALSGWJA, combining the grey wolf optimizer (GWO) and JAYA methods, enhanced by approximate line search. HALSGWJA utilizes line search augmented by approximate gradient information to generate trial designs that are always likely to be better than the current best record. This goal is reached at low computational cost because the generation and selection of trial design relies on cost function evaluations rather than on optimization constraints.

HALSGWJA was successfully tested on 20 real-world mechanical engineering optimization problems: (i) 19 test cases included in the CEC2020 library on real-world mechanical engineering and (ii) an example of optimal crashworthiness design. The extensive comparisons with 18 other state-of-the-art metaheuristic optimizers in the CEC2020 problems and 25 other optimizers in the car side impact problem proved without a shadow of a doubt the superiority of HALSGWJA over its competitors. In fact, the present algorithm practically converged to target optima in all test problems (the largest deviation from target cost was only 0.0263% for problem 13 of the CEC2020 library) and achieved very often 0 or nearly 0 standard deviation on optimized cost. Furthermore, HALSGWJA always ranked first in terms of computational speed, requiring fewer analyses than its competitors and very often exhibiting a moderate dispersion on the required number of analyses. Statistical analysis based on Wilcoxon signed-rank and ranking-based Friedman tests confirmed the efficiency of the proposed algorithm.

The results presented in this article support the conclusion that HALSGWJA is a powerful tool for real-world mechanical engineering optimization. Remarkably, this goal was achieved by developing a rather simple optimization formulation. The proposed algorithm does not present any theoretical limitation that may affect its search capability beyond the inherent limitation applied by the No Free Lunch Theorem, for which no metaheuristic algorithm can always outperform all other metaheuristic algorithms in all optimization problems. New investigations are currently evaluating the behavior of HALSGWJA in (i) large-scale problems with hundreds of variables and thousands of non-linear constraints, (ii) discrete structural optimization problems with frequency constraints, and (iii) inverse problems such as mechanical characterization of anisotropic hyperelastic materials. The preliminary results obtained in problem types (i), (ii), and (iii) also seem to confirm the computational efficiency of HALSGWJA in optimization problems that include very high levels of nonlinearity and non-convexity.

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: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding authors.

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

Appendix A—HALSGWJA's optimized designs for CEC2020 real-world mechanical engineering problems

PROBLEM 1—Speed reducer design (7 optimization variables)

$\vec{X}_{opt} = \{3.50000; 0.70000; 17; 7.30000; 7.71533; 3.35054; 5.28665\}$

PROBLEM 2—Industrial refrigeration system (14 optimization variables)

$\overrightarrow{X_{opt}} = \{0.001; 0.001; 0.001; 0.001; 0.001; 0.001; 1.524; 1.524; 5; 2; 0.001; 0.001; 0.0072934; 0.0875\}$
 56
 PROBLEM 3—Tension/compression spring design (Case 1) (3 optimization variables)
 $\overrightarrow{X_{opt}} = \{0.051689; 0.35673; 11.2883\}$
 PROBLEM 4—Pressure vessel design (4 optimization variables)
 $\overrightarrow{X_{opt}} = \{0.81205; 0.4375; 42.0984; 176.6372\}$ Mixed variables problem variant;
 $\overrightarrow{X_{opt}} = \{0.77817; 0.38465; 40.31962; 200\}$ Continuous variables problem variant.
 PROBLEM 5—Welded beam design (4 optimization variables)
 $\overrightarrow{X_{opt}} = \{0.19883; 3.33736; 9.19202; 0.19883\}$
 PROBLEM 6—Three-bar truss design (2 optimization variables)
 $\overrightarrow{X_{opt}} = \{0.78865; 0.40824\}$
 PROBLEM 7—Multiple disk clutch brake design (5 optimization variables)
 $\overrightarrow{X_{opt}} = \{70; 90; 1; 995.4261; 2\}$
 PROBLEM 8—Planetary gear train design (9 optimization variables)
 $\overrightarrow{X_{opt}} = \{40; 21; 14; 19; 14; 69; 5; 2.25; 2.5\}$
 PROBLEM 9—Step-cone pulley (5 optimization variables)
 $\overrightarrow{X_{opt}} = \{38.47631; 53.10083; 95.34589; 158.33150; 47.91321\}$
 PROBLEM 10—Robot gripper (7 optimization variables)
 $\overrightarrow{X_{opt}} = \{150; 149.88285; 200; 1.26622 \times 10^{-13}; 150; 100.94276; 2.29741\}$
 PROBLEM 11—Hydrostatic thrust bearing design (4 optimization variables)
 $\overrightarrow{X_{opt}} = \{5.95551; 5.38872; 5.3587 \times 10^{-6}; 2.25664\}$
 PROBLEM 12—Four-stage gearbox design (22 optimization variables)
 $\overrightarrow{X_{opt}} = \{7; 7; 7; 26; 9; 7; 9; 31; 8.255; 3.175; 5.715; 3.175; 50.8; 50.8; 63.5; 101.6; 25.4; 88.9; 101.6; 76.2; 101.6; 76.2\}$
 PROBLEM 13—Ten-bar truss design (10 optimization variables)
 $\overrightarrow{X_{opt}} = \{0.0035200; 0.0014242; 0.0035191; 0.0015087; 6.4500 \times 10^{-5}; 0.00045689; 0.0022838; 0.0024685; 0.0012649; 0.0012079\}$
 PROBLEM 14—Rolling element bearing (10 optimization variables)
 $\overrightarrow{X_{opt}} = \{131.2; 18; 5; 0.6; 0.6; 0.4071; 0.6427; 0.3; 0.0632; 0.6\}$
 PROBLEM 15—Gas transmission compressor design (4 optimization variables)
 $\overrightarrow{X_{opt}} = \{50; 1.17828; 24.59258; 0.38835\}$
 PROBLEM 16—Tension/compression spring design (Case 2) (3 optimization variables)
 $\overrightarrow{X_{opt}} = \{9; 1.22311; 0.283\}$
 PROBLEM 17—Gear train design (4 optimization variables)
 $\overrightarrow{X_{opt}} = \{13; 30; 53; 51\}$
 PROBLEM 18—Himmelblau's function (5 optimization variables)
 $\overrightarrow{X_{opt}} = \{78; 33; 29.99701; 44.98974; 36.77737\}$
 PROBLEM 19—Topology optimization (30 optimization variables). The solution shows all significant digits for the double precision limit adopted in the computations to demonstrate the convergence of optimal topology towards the uniform density of 1.
 $\overrightarrow{X_{opt}} = \{0.99999999999973; 0.99999999999979; 0.99999999999999; 0.99999999999995; 7; 1; 0.99999999999996; 1; 0.99999999999996; 1; 1; 0.99999999999951; 0.99999999999976; 0.99999999999991; 0.99999999999986; 1; 0.99999999999998; 0.99999999999999; 1; 1; 0.999999999999846; 0.999999999999931; 0.999999999999918; 0.999999999999744; 0.999999999999993; 1; 1; 1; 1; 1\}$

References

1. Mendes Platt, G.; Yang, X.-S.; Silva Neto, A.J. *Computational Intelligence, Optimization and Inverse Problems with Applications in Engineering*; Springer: Cham, Switzerland, 2019.
2. Elaziz, M.A.; Elsheikh, A.H.; Oliva, D.; Abualigah, L.; Lu, S.; Ewees, A. Advanced metaheuristic techniques for mechanical design problems: Review. *Arch. Comput. Methods Eng.* **2022**, *29*, 695–716.
3. Turgut, O.E.; Turgut, M.S.; Kirtepe, E. A systematic review of the emerging metaheuristic algorithms on solving complex optimization problems. *Neural Comput. Appl.* **2023**, *35*, 14275–14378.
4. Abualigah, L. *Metaheuristic Optimization Algorithms—Optimizers, Analysis, and Applications*, 1st ed.; Elsevier: Amsterdam, The Netherlands, 2024.
5. Cuevas, E.; Chavarin-Fajardo, A.; Ascencio-Piña, C.; Garcia-De-Lira, S. *Optimization Strategies: A Decade of Metaheuristic Algorithm Development*; Springer: Berlin/Heidelberg, Germany, 2025.
6. Furio, C.; Lamberti, L.; Pruncu, C. Mechanical and civil engineering optimization with a very simple hybrid GreyWolf—JAYA metaheuristic optimizer. *Mathematics* **2024**, *12*, 3464.
7. Furio, C.; Lamberti, L.; Pruncu, C. An efficient and fast hybrid GWO-JAYA algorithm for design optimization. *Appl. Sci.* **2024**, *14*, 9610.
8. 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.
9. Hwang, S.F.; He, R.S. Improving real-parameter genetic algorithm with simulated annealing for engineering problems. *Adv. Eng. Softw.* **2006**, *37*, 406–418.
10. Lamberti, L.; Pappalettere, C. Weight optimization of skeletal structures with multipoint simulated annealing. *Comput. Model. Eng. Sci.* **2007**, *18*, 183–221.
11. Degertekin, S.O. Optimum design of steel frames using harmony search algorithm. *Struct. Multidiscip. Optim.* **2008**, *36*, 393–401.
12. 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.
13. Hasancebi, O.; Erdal, F.; Saka, M.P. Adaptive harmony search method for structural optimization. *ASCE J. Struct. Eng.* **2010**, *136*, 419–431.
14. 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.
15. Yang, X.-S.; Gandomi, A.H. Bat algorithm: A novel approach for global engineering optimization. *Eng. Comput.* **2012**, *29*, 464–483.
16. 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.
17. Liu, Z.; Atamturktur, S.; Juang, C.H. Reliability based multi-objective robust design optimization of steel moment resisting frame considering spatial variability of connection parameters. *Eng. Struct.* **2014**, *76*, 393–403.
18. Kaveh, A.; Bakhshpoori, T.; Azimi, M. Seismic optimal design of 3D steel frames using cuckoo search algorithm. *Struct. Des. Tall Spec. Build.* **2015**, *24*, 210–217.
19. Kaveh, A.; Bakhshpoori, T. A new metaheuristic for continuous structural optimization: Water evaporation optimization. *Struct. Multidiscip. Optim.* **2016**, *54*, 23–43.
20. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp swarm algorithm: A bioinspired optimizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191.
21. Farshchin, M.; Maniat, M.; Camp, C.V.; Pezeshk, S. School-based optimization algorithm for design of steel frames. *Eng. Struct.* **2018**, *171*, 326–335.
22. Keshtegar, B.; Hao, P.; Wang, Y.; Hu, Q. An adaptive response surface method and Gaussian global-best harmony search algorithm for optimization of aircraft stiffened panels. *Appl. Soft Comput.* **2018**, *66*, 196–207.
23. Jafari, S.; Nikolaidis, T. Meta-heuristic global optimization for aircraft engines modelling and controller design; A review, research challenges and exploring the future. *Prog. Aerosp. Sci.* **2019**, *104*, 40–53.
24. Kaveh, A.; Biabani Hamedani, K.; Milad Hosseini, S.; Bakhshpoori, T. Optimal design of planar steel frame structures utilizing meta-heuristic optimization algorithms. *Structures* **2020**, *25*, 335–346.
25. Korkmaz, F.F.; Subran, M.; Yildiz, A.R. Optimal design of aerospace structures using recent meta-heuristic algorithms. *Mater. Test.* **2021**, *11*, 1025–1031.

26. Ficarella, E.; Lamberti, L.; Degertekin, S.O. Comparison of three novel hybrid metaheuristic algorithms for structural optimization problems. *Comput. Struct.* **2021**, *244*, 106395.
27. Gupta, S.; Abderazek, H.; Yildiz, B.S.; Yildiz, A.R.; Mirjalili, S.; Said, S.M. Comparison of metaheuristic algorithms for solving constrained mechanical design optimization problems. *Exp. Syst. Appl.* **2021**, *183*, 115351.
28. Abualigah, L.; Elaziz, M.A.; Khasawneh, A.M.; Alshinwan, M.; Ibrahim, R.A.; Al-ganess, M.A.A.; Mirjalili, S.; Sumari, P.; Gandomi, A.H. Meta-heuristic optimization algorithms for solving real-world mechanical engineering design problems: A comprehensive survey, applications, comparative analysis, and results. *Neural Comput. Appl.* **2022**, *34*, 4081–4110.
29. 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.
30. 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.
31. Houssein, E.H.; Gafar, M.H.A.; Fawzy, N.; Sayed, A.Y. Recent metaheuristic algorithms for solving some civil engineering optimization problems. *Sci. Rep.* **2025**, *15*, 7929.
32. Wang, J.; Ouyang, H.; Li, S.; Ding, W.; Gao, L. Equilibrium optimizer-based harmony search algorithm with nonlinear dynamic domains and its application to real-world optimization problems. *Artif. Intell. Rev.* **2024**, *57*, 166.
33. Akcay, O.; Ilkilic, C. Light-weight design of aerospace components using genetic algorithm and dandelion optimization algorithm. *Int. J. Aeronaut. Space Sci.* **2025**, *26*, 2898–2909.
34. Phuekpan, K.; Khammee, R.; Panagant, N.; Bureerat, S.; Pholdee, N.; Wansasueb, K. A comparison of modern metaheuristics for multi-objective optimization of transonic aeroelasticity in a tow-steered composite wing. *Aerospace* **2025**, *12*, 101.
35. Hossain, M.S.; Chao, O.Z.; Ismail, Z.; Noroozi, S.; Khoo, S.Y. Artificial neural networks for vibration based inverse parametric identifications: A review. *Appl. Soft Comput.* **2017**, *52*, 203–219.
36. Ficarella, E.; Lamberti, L.; Degertekin, S.O. Mechanical identification of materials and structures with optical methods and metaheuristic optimization. *Materials* **2019**, *12*, 2133.
37. Cuevas, E.; Gálvez, J.; Avalos, O. *Recent Metaheuristics Algorithms for Parameter Identification*; Springer: Cham, Switzerland, 2020.
38. Hassan, H.; Tallman, T.Y. A Comparison of metaheuristic algorithms for solving the piezoresistive inverse problem in self-sensing materials. *IEEE Sens. J.* **2021**, *21*, 659–666.
39. Di Lecce, M.; Onaizah, O.; Lloyd, P.; Chandler, J.H.; Valdastrì, P. Evolutionary inverse material identification: Bespoke characterization of soft materials using a metaheuristic algorithm. *Front. Robot. AI* **2022**, *8*, 790571.
40. Zhai, H.; Hao, H.; Yeo, J. Benchmarking inverse optimization algorithms for materials design. *APL Mater.* **2024**, *12*, 021107.
41. Oliveira, M.A.; Inman, D.J. Performance analysis of simplified fuzzy ARTMAP and probabilistic neural networks for identifying structural damage growth. *Appl. Soft Comput.* **2017**, *52*, 53–63.
42. Li, H.; Zhang, Q.; Qin, X.; Yuantao, S. Raw vibration signal pattern recognition with automatic hyper-parameter-optimized convolutional neural network for bearing fault diagnosis. *Proc. Inst. Mech. Eng. C J. Mech. Eng. Sci.* **2019**, *234*, 343–360.
43. da Silva Lopes Alexandrino, S.; Ferreira Gomes, G.; Simões Cunha Jr, S. A robust optimization for damage detection using multiobjective genetic algorithm, neural network and fuzzy decision making. *Inverse Probl. Sci. Eng.* **2020**, *28*, 21–46.
44. 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.
45. Benaissa, B.; Hocine, N.A.; Khatir, S.; Riahi, M.K.; Mirjalili, S. YUKI algorithm and POD-RBF for elastostatic and dynamic crack identification. *J. Comput. Sci.* **2021**, *55*, 101451.
46. Hassani, S.; Dackermann, U. Optimization-based damage detection in composite structures using incomplete measurements. *Structures* **2023**, *56*, 104825.
47. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61.
48. Sharma, I.; Kumar, V.; Sharma, S. A comprehensive survey on grey wolf optimization. *Recent Adv. Comput. Sci. Commun.* **2022**, *15*, 323–333.
49. 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.
50. Tsai, H.-C.; Shi, J.-Y. Potential corrections to grey wolf optimizer. *Appl. Soft Comp.* **2024**, *161*, 111776.
51. Rao, R.V. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *Int. J. Ind. Eng. Comput.* **2016**, *7*, 19–34.
52. Kumar, A.; Wu, G.; Ali, M.Z.; Mallipeddi, R.; Suganthan, P.N.; Das, S. A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. *Swarm Evol. Comput.* **2020**, *56*, 100693.

53. Kumar, A.; Das, S.; Zelinka, I. A self-adaptive spherical search algorithm for real-world constrained optimization problems. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion, Cancún, Mexico, 8–12 July 2020; pp. 13–14.
54. Sallam, K.M.; Elsayed, S.M.; Chakraborty, R.K.M.; Ryan, M.J. Multi-operator differential evolution algorithm for solving real-world constrained optimization problems. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020; pp. 1–8.
55. Gurrola-Ramos, J.; Hernández-Aguirre, A.; Dalmau-Cedeño, O. COLSHADE for real-world single-objective constrained optimization problems. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020; pp. 1–8.
56. Tang, H.; Lee, J. Adaptive initialization LSHADE algorithm enhanced with gradient-based repair for real-world constrained optimization. *Knowl.-Based Syst.* **2022**, *246*, 108696.
57. Yuan, Z.; Peng, L.; Dai, G.; Wang, M.; Zhang, W.; Zhou, Q.; Zheng, W. An improved multi-operator differential evolution via a knowledge-guided information sharing strategy for global optimization. *Exp. Syst. Appl.* **2025**, *269*, 126403.
58. Lee, J.; Mendoza, R.; Mendoza, V.M.P.; Jung, E. Differential evolution with spherical search algorithm for nonlinear engineering and infectious disease optimization problems. *Appl. Soft Comp.* **2025**, *168*, 112446.
59. Sharma, D.; Jabeen, S.D. Hybridizing interval method with a heuristic for solving real-world constrained engineering optimization problems. *Structures* **2023**, *56*, 104993.
60. Hu, G.; Guo, Y.; Zhong, J.; Wei, G. IYDSE: Ameliorated Young's double-slit experiment optimizer for applied mechanics and engineering. *Comput. Methods Appl. Mech. Engrg.* **2023**, *412*, 116062.
61. Hu, G.; Gong, C.; Li, X.; Xu, Z. CGKOA: An enhanced Kepler optimization algorithm for multi-domain optimization problems. *Comput. Methods Appl. Mech. Engrg.* **2024**, *425*, 116964.
62. Shu, B.; Hu, G.; Cheng, M.; Zhang, C. MSFPSO: Multi-algorithm integrated particle swarm optimization with novel strategies for solving complex engineering design problems. *Comput. Methods Appl. Mech. Engrg.* **2025**, *437*, 117791.
63. You, G.; Hu, Y.; Yang, Z.; Li, Y. Enhanced snow ablation optimizer using dynamic tangential flight and elite guidance strategy. *Sci. Rep.* **2025**, *15*, 10036.
64. Fu, S.; Huang, H.; Ma, C.; Wei, J.; Li, Y.; Fu, Y. Improved dwarf mongoose optimization algorithm using novel nonlinear control and exploration strategies. *Exp. Syst. Appl.* **2023**, *233*, 120904.
65. 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.
66. Wang, Z.; Dai, D.; Zeng, Z.; He, D.; Chan, S. Multi-strategy enhanced Grey Wolf Optimizer for global optimization and real world problems. *Clust. Comput.* **2024**, *27*, 10671–10715.
67. Naik, A. Marine predators social group optimization: A hybrid approach. *Evol. Intell.* **2024**, *17*, 2355–2386.
68. Kaur, S.; Kaur, L.; Lal, M. Artificial hummingbird algorithm with chaotic-opposition-based population initialization for solving real-world problems. *Neural Comput. Appl.* **2024**, *in press*.
69. Zhang, C.; Li, H.; Long, S.; Yue, X.; Ouyang, H.; Chen, Z.; Li, S. Piranha predation optimization algorithm (PPOA) for global optimization and engineering design problems. *Appl. Soft Comput.* **2024**, *165*, 112085.
70. 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.
71. Ankrah, J.C.; Bofo Effah, F.; Twumasi, E. An enhanced semisteady-state Jaya algorithm with a control coefficient and a self-adaptive multipopulation strategy. *J. Electr. Comput. Eng.* **2025**, *1*, 3036909.
72. 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.
73. 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.
74. Abdel-Basset, M.; Mohamed, R.; Azeem, S.A.A.; Jameel, M.; Abouhawwash, M. Kepler optimization algorithm: A new metaheuristic algorithm inspired by Kepler's laws of planetary motion. *Knowl.-Based Syst.* **2023**, *268*, 110454.
75. Azizi, M.; Aickelin, U.; Khorshidi, H.A.; Shishehgharkhaneh, M.B. Energy valley optimizer: A novel metaheuristic algorithm for global and engineering optimization. *Sci. Rep.* **2023**, *13*, 226.
76. 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.

77. Li, K.; Huang, H.; Fu, S.; Ma, C.; Fan, Q.; Zhu, Y. A multi-strategy enhanced northern goshawk optimization algorithm for global optimization and engineering design problems. *Comput. Methods Appl. Mech. Engrg.* **2023**, *415*, 116199.
78. Zhao, Y.; Huang, C.; Zhang, M.; Lv, C. COLMA: A chaos-based mayfly algorithm with opposition-based learning and Levy flight for numerical optimization and engineering design. *J. Supercomput.* **2023**, *79*, 19699–19745.
79. Wang, K.; Guo, M.; Dai, C.; Li, Z. A novel heuristic algorithm for solving engineering optimization and real-world problems: People identity attributes-based information-learning search optimization. *Comput. Methods Appl. Mech. Engrg.* **2023**, *416*, 116307.
80. Wu, L.; Wu, J.; Wang, T. The improved grasshopper optimization algorithm with Cauchy mutation strategy and random weight operator for solving optimization problems. *Evol. Intell.* **2024**, *17*, 1751–1781.
81. Luan, T.-M.; Khatir, S.; Tran, M.-T.; De Baets, B.; Cuong-Le, T. Exponential-trigonometric optimization algorithm for solving complicated engineering problems. *Comput. Methods Appl. Mech. Engrg.* **2024**, *432*, 117411.
82. Fu, Y.; Liu, D.; Chen, J.; He, L. Secretary bird optimization algorithm: A new metaheuristic for solving global optimization problems. *Artif. Intell. Rev.* **2024**, *57*, 123.
83. Lang, Y.; Gao, Y. Dream Optimization Algorithm (DOA): A novel metaheuristic optimization algorithm inspired by human dreams and its applications to real-world engineering problems. *Comput. Methods Appl. Mech. Engrg.* **2025**, *436*, 117718.
84. Xiao, Y.; Cui, H.; Khurma, R.A.; Castillo, P.A. Artificial lemming algorithm: A novel bionic meta-heuristic technique for solving real-world engineering optimization problems. *Artif. Intell. Rev.* **2025**, *58*, 84.
85. Zhong, C.; Li, G.; Meng, Z.; Li, H.; Yildiz, A.R.; Mirjalili, S. Starfish optimization algorithm (SFOA): A bio-inspired metaheuristic algorithm for global optimization compared with 100 optimizers. *Neural Comput. Appl.* **2025**, *37*, 3641–3683.
86. Cai, X.; Wang, W.; Wang, Y. Multi-strategy enterprise development optimizer for numerical optimization and constrained problems. *Sci. Rep.* **2025**, *15*, 10538.
87. Gandomi, A.H.; Yang, X.-S.; Alavi, A.H. Mixed variable structural optimization using Firefly Algorithm. *Comput. Struct.* **2011**, *89*, 2325–2336.
88. Youn, B.D.; Choi, K.K. A new response surface methodology for reliability-based design optimization. *Comput. Struct.* **2004**, *82*, 241–256.
89. 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.
90. 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.
91. Deb, K. An efficient constraint handling method for genetic algorithms. *Comput. Methods Appl. Mech. Eng.* **2000**, *186*, 311–338.
92. Coello Coello, C.A. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. *Comput. Methods Appl. Mech. Eng.* **2002**, *191*, 1245–1287.
93. Eberhart, R.C.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995.
94. Liang, J.J.; Qin, A.K.; Suganthan, P.N.; Baskar, S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. Evol. Comput.* **2006**, *10*, 281–295.
95. Yang, X.S.; Deb, S. Engineering optimisation by cuckoo search. *Int. J. Math. Model.* **2010**, *1*, 330–343.
96. Kaur, S.; Awasthi, L.K.; Sangal, A.L.; Dhiman, G. Tunicate Swarm Algorithm: A new bio-inspired based metaheuristic paradigm for global optimization. *Eng. Appl. Artif. Intell.* **2020**, *90*, 103541.
97. Omran, M.G.; Al-Sharhan, S. Improved continuous Ant Colony Optimization algorithms for real-world engineering optimization problems. *Eng. Appl. Artif. Intell.* **2019**, *85*, 818–829.
98. Gandomi, A.H.; Alavi, A.H. Krill herd: A new bio-inspired optimization algorithm. *Commun. Nonlinear Sci. Numer. Simul.* **2012**, *17*, 4831–4845.
99. Banharnsakun, A.; Achalakul, T.; Sirinaovakul, B. The best-so-far selection in artificial bee colony algorithm. *Appl. Soft Comput.* **2011**, *11*, 2888–2901.
100. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67.
101. Peraza-Vazquez, H.; Pena-Delgado, A.; Merino-Trevino, M.; Morales-Cepeda, A.B.; Sinha, N. A novel metaheuristic inspired by horned lizard defense tactics. *Artif. Intell. Rev.* **2024**, *57*, 59.
102. 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.

103. Agushaka, J.O.; Ezugwu, A.E.; Abualigah, L. Dwarf mongoose optimization algorithm. *Comput. Methods Appl. Mech. Eng.* **2022**, *391*, 114570.
104. Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A.H. The arithmetic optimization algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609.
105. Mohamed, A.W.; Hadi, A.A.; Fattouh, A.M.; Jambi, K.M. LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems. In Proceedings of the 2017 IEEE Congress on Evolutionary Computations, Donostia-San Sebastian, Spain, 5–8 June 2017; pp. 145–152.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.