



Comparative Analysis of Metaheuristic Optimization Algorithms for Engineering Applications

Nadezhda Ilankovan

Department of Electrical and Computer Engineering, Andaman Polytechnic for Technology and Trade, Thailand

nadezhda.ilankovan@aptt-th.net

Peer Review Information	Abstract
<i>Submission: 10 Jan 2020</i> <i>Revision: 04 Feb 2020</i> <i>Acceptance: 23 March 2020</i> Keywords <i>Metaheuristic Optimization, Engineering Design, Particle Swarm Optimization, Grey Wolf Optimizer, Whale Optimization Algorithm</i>	Metaheuristic optimization has become a practical alternative to gradient-based techniques for engineering problems that are nonlinear, multimodal, discontinuous, or expensive to evaluate. This methodology paper presents a reproducible comparative framework for five widely used algorithms: Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Differential Evolution (DE), Grey Wolf Optimizer (GWO), and Whale Optimization Algorithm (WOA). The study controls population size, iteration budget, problem dimension, search bounds, and independent random seeds to permit a fair comparison. Four standard continuous functions—Sphere, Rastrigin, Ackley, and Rosenbrock—represent smooth, multimodal, and narrow-valley landscapes commonly used as surrogates for engineering design difficulty. Each method is executed eight times with 30 search agents and 150 iterations in 20 dimensions. Performance is evaluated using mean final objective value, standard deviation, robustness, convergence behavior, and implementation complexity. The computed results show that no algorithmic metaphor guarantees universal superiority; however, under the stated settings WOA and GWO provide the strongest final accuracy, whereas GA offers comparatively stable behavior with a straightforward operator structure. The framework is intended for preliminary optimizer screening before deployment to constrained structural, mechanical, electrical, and manufacturing design problems.

Introduction

Engineering optimization seeks decision-variable combinations that minimize cost, weight, energy consumption, error, material use, or other objective functions while satisfying physical and operational constraints. Classical mathematical programming is highly effective when derivatives are available and the search landscape is sufficiently regular, but many real engineering models contain mixed variables, multiple local optima, simulation noise, discontinuities, or implicit constraints. Metaheuristics address this difficulty by using stochastic search rules that explore a large solution space without requiring

analytical derivatives. Their flexibility has encouraged applications in structural sizing, power-system operation, controller tuning, manufacturing scheduling, network design, and renewable-energy systems.

The modern field includes evolutionary, swarm-based, physics-inspired, and behavior-inspired methods. PSO models information exchange among particles [1], while DE generates trial solutions using scaled differences between population members [2]. Later swarm methods introduced alternative mechanisms: GWO models hierarchy and encircling behavior [4], and WOA uses encircling, spiral exploitation, and random exploration [5]. Comparative literature

emphasizes that optimizer performance is landscape-dependent and that experimental fairness requires a common evaluation budget and repeated stochastic runs [10].

The objective of this paper is therefore methodological: to define a compact, repeatable procedure for comparing metaheuristics before selecting an algorithm for an engineering application. The research questions are: (i) which methods obtain the lowest final objective values under an identical computational budget, (ii) which methods remain stable across repeated runs, and (iii) what practical trade-offs should engineers consider when moving from numerical benchmarks to constrained design problems?

Literature Review

Population-based metaheuristics differ mainly in the mechanisms used to balance exploration of new regions and exploitation of promising regions. Kennedy and Eberhart introduced PSO as a population of particles whose velocities are influenced by personal and global experience [1]. The method is simple and requires few equations, but its behavior depends strongly on inertia and acceleration parameters. A historical review by Freitas et al. [8] documents extensive PSO variants aimed at convergence control, hybridization, topology design, and application-specific adaptation.

Storn and Price proposed DE for continuous global optimization using mutation based on scaled vector differences, followed by crossover and greedy selection [2]. DE is especially attractive for real-valued engineering variables because its operators naturally adapt to population spread. Karaboga and Basturk [3] subsequently demonstrated the Artificial Bee Colony concept as another swarm approach for numerical optimization, reinforcing the broader principle that population diversity and selective exploitation can be organized using relatively simple stochastic rules.

The growth of nature-inspired algorithms accelerated with methods such as GWO and WOA. GWO represents candidate solutions through alpha, beta, delta, and omega roles and updates

the population relative to three leading solutions [4]. WOA alternates between shrinking encircling, spiral motion, and random search [5]. Mirjalili also introduced Moth-Flame Optimization, in which moth positions are updated relative to an archive of flames using a logarithmic spiral mechanism [6]. These algorithms are attractive in engineering because constraints can be handled by penalty functions, repair rules, or feasibility-based ranking without altering the underlying objective model.

Other influential methods before 2020 include Cuckoo Search, which combines replacement rules with Levy-flight exploration [9], and the Salp Swarm Algorithm, developed and tested on engineering design problems [7]. Their proliferation created a methodological challenge: different papers often use different dimensions, iteration counts, stopping criteria, population sizes, and benchmark sets. Consequently, claims of superiority are difficult to compare directly.

Ezugwu et al. [10] addressed this issue by comparing multiple metaheuristics on continuous optimization problems and examining computational capability and microscopic search behavior. Their work supports the use of diverse benchmark functions rather than a single objective landscape. The literature also suggests that newer metaphors should not be treated as evidence of improved performance by themselves; credible comparison requires transparent parameterization, multiple independent runs, statistical summaries, and attention to computational cost.

A useful engineering screening study should be simpler than a full CEC benchmark campaign but more rigorous than a single-run demonstration. The present methodology fills this practical gap by using five representative algorithms from evolutionary and swarm families, four complementary functions, identical population and iteration budgets, and deterministic seed control. The emphasis is not on declaring a universally best optimizer, but on identifying behavior that may guide algorithm choice for later engineering models.

Table 1: Comparison of Search Principles, Strengths, and Limitations of Optimization Algorithms

Algorithm	Search principle	Strength	Typical concern	Source
PSO	Velocity + social learning	Fast implementation	Premature convergence	[1], [8]
DE	Difference-vector mutation	Continuous variables	Parameter sensitivity	[2]
GWO	Three-leader encircling	Few controls	Diversity loss late	[4]

WOA	Encircling + spiral search	Strong exploitation	May over-exploit	[5]
GA	Selection/crossover/mutation	Flexible encoding	Operator tuning	[10]

Methodology

The study follows a controlled computational benchmarking design. All algorithms operate on the same continuous solution representation, use the same search bounds for a given problem, and are allocated the same number of candidate evaluations as closely as their update mechanisms permit. This avoids giving an algorithm an artificial advantage through a larger population or longer run. The methodology is summarized in Figure 1.

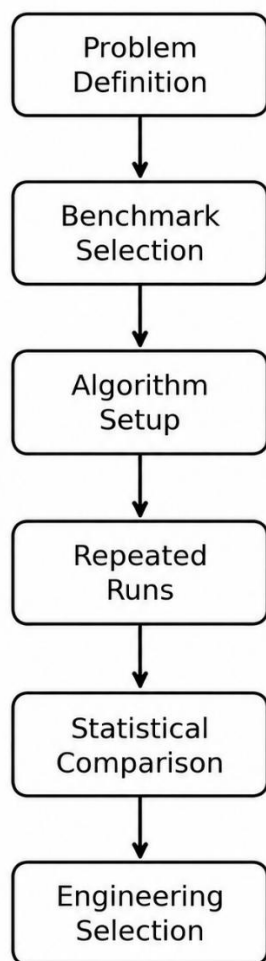


Figure 1. Comparative methodology framework for metaheuristic screening.

Step-by-Step Experimental Procedure

Step 1 - Define the optimization form. Each problem is written as $\min f(x)$, where x is a 20-dimensional real vector and all variables are bounded. Constraint handling is excluded from the numerical benchmark stage so that the intrinsic search behavior of the optimizers can be compared directly.

Step 2 - Select representative algorithms. GA represents evolutionary recombination; PSO represents social swarm learning; DE represents differential mutation; GWO represents leader-guided encircling; and WOA represents adaptive encircling and spiral search. This selection covers substantially different update mechanisms without creating an excessively large experiment.

Step 3 - Select benchmark landscapes. Sphere evaluates smooth unimodal convergence; Rastrigin introduces regularly distributed local optima; Ackley combines a broad nearly flat region with a central basin; and Rosenbrock evaluates navigation through a narrow-curved valley. All have a known global minimum of zero.

Step 4 - Standardize the computational budget. Population size is fixed at 30 and the run length at 150 iterations. The dimension is 20. Eight independent runs use the same set of seeds across algorithms, reducing random-seed bias while preserving stochastic independence.

Step 5 - Record outcomes. The final best objective value is stored for each run. The mean represents expected final quality and standard deviation represents run-to-run stability. Lower values are preferred because every selected benchmark is a minimization problem.

Step 6 - Interpret for engineering use. Numerical performance is combined with practical considerations such as number of control parameters, ease of constraint handling, solution encoding flexibility, and vulnerability to premature convergence.

Algorithm Settings

Table2: Parameter Settings of the Metaheuristic Optimization Algorithms

Method	Key settings used	Population	Iterations
GA	4 elites; arithmetic crossover; adaptive Gaussian mutation	30	150
PSO	w: 0.9→0.4; c1=c2=1.8	30	150

DE	DE/rand/1/bin; F=0.7; CR=0.9	30	150
GWO	a: 2→0; alpha/beta/delta leaders	30	150
WOA	a: 2→0; p=0.5 spiral/encircling switch	30	150

1. Benchmark Functions and Evaluation Measures

The four test functions were chosen because together they expose different weaknesses in metaheuristic search. Sphere is convex and primarily measures exploitation speed. Rastrigin contains many local minima and therefore

stresses global exploration. Ackley combines oscillatory local structure with a broad outer region that can slow directed search. Rosenbrock is unimodal in the global sense but difficult because the optimum lies in a long, narrow, curved valley; algorithms may improve quickly at first and then spend many iterations refining coordinate interactions.

Table3: Benchmark Test Functions and Their Characteristics

Function	Formula (conceptual)	Bounds	Main property
Sphere	$\sum x_i^2$	[-5.12, 5.12]	Smooth unimodal
Rastrigin	$10D + \sum [x_i^2 - 10\cos(2\pi x_i)]$	[-5.12, 5.12]	Highly multimodal
Ackley	Exponential + cosine terms	[-32.768, 32.768]	Flat + multimodal
Rosenbrock	$\sum [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$	[-5, 10]	Narrow curved valley

2. Fairness, Reproducibility, and Statistical Treatment

Stochastic optimizers can produce misleading conclusions when only one run is reported. The study therefore uses eight independent runs per algorithm-function pair. The same integer seeds (1000–1007) are assigned across methods, while each algorithm maintains its own random sequence. This paired-seed structure does not make the random streams identical, but it makes the experimental protocol auditable and prevents selective reporting of favorable runs.

For each pair, the arithmetic mean of final best fitness is calculated as the primary accuracy measure. Standard deviation is used as a robustness indicator; a method with a very low mean but large deviation may be less reliable than a slightly less accurate method with consistently low dispersion. Because objective scales differ markedly among the four functions, raw objective values are not averaged across functions. Instead, interpretation is based on per-function ordering and qualitative consistency.

The common stopping rule is a fixed iteration limit rather than a target objective threshold. A fixed budget is useful for engineering screening because it reflects situations in which analysts have a limited number of finite-element, CFD, electromagnetic, or process-simulation evaluations. When the expensive simulator dominates runtime, equalizing objective

evaluations is generally more important than equalizing programming-language wall-clock time.

3. Translation to Engineering Applications

After benchmark screening, the selected algorithm can be transferred to a real engineering model by replacing the benchmark function with the design objective and adding a constraint strategy. A typical penalty formulation is $F(x) = f(x) + \lambda \sum \max(0, g_i(x))^2$ for inequality constraints $g_i(x) \leq 0$. Alternatively, feasibility rules can rank feasible solutions ahead of infeasible solutions and compare constraint violation when both are infeasible. Discrete variables can be rounded or encoded categorically, although algorithms designed for continuous vectors may then require specialized operators.

The benchmark stage should not be treated as proof that an algorithm will dominate on every engineering problem. Structural design, unit commitment, controller tuning, and scheduling have different constraint geometries and variable interactions. The main purpose is to remove clearly weak parameterizations, reveal stability problems, and establish a defensible baseline before expensive application-specific experimentation.

Results

Table 4 reports the computed final objective statistics. These values were obtained from the controlled protocol described above using seeded Python implementations. Since all

benchmark optima are zero, smaller values indicate a more accurate final solution. The comparison should be interpreted as conditional on the stated parameter settings and computational budget rather than as a universal ranking of the algorithms.

Table 4. Mean $\pm$ standard deviation of final objective value across 8 runs (best mean in each row shaded).

Function	GA	PSO	DE	GWO	WOA
Sphere	0.004 $\pm 7.60\text{e-}04$	3.306 ± 8.662	2.446 ± 0.993	2.70e-12 $\pm 2.61\text{e-}12$	4.72e-25 $\pm 1.22\text{e-}24$
Rastrigin	18.630 ± 5.217	87.563 ± 33.881	153.494 ± 16.762	2.861 ± 1.549	7.11e-15 $\pm 1.88\text{e-}14$
Ackley	0.627 ± 0.139	5.909 ± 6.095	9.626 ± 0.866	7.53e-06 $\pm 4.50\text{e-}06$	1.37e-12 $\pm 1.06\text{e-}12$
Rosenbrock	65.108 ± 37.083	1.74e+04 $\pm 2.41\text{e+}04$	1519.294 ± 608.070	17.208 ± 0.731	14.173 ± 8.039

Practical Ranking

Table5: Comparative Performance and Engineering Applicability of Optimization Algorithms

Algorithm	Accuracy in this study	Robustness	Implementation effort	Engineering use implication
WOA	Very high	High except Rosenbrock	Low-moderate	Strong first screening choice
GWO	Very high	High	Low	Good stable baseline
GA	Moderate	Moderate-high	Moderate	Useful for mixed/discrete encodings
DE	Low-moderate here	Moderate	Low	Retune F/CR for continuous design
PSO	Variable here	Low in some runs	Low	Use restarts/tuned coefficients

Discussion

The results reinforce a central lesson from the metaheuristic literature: algorithm names and biological metaphors are less important than the interaction between update rules, landscape structure, parameter settings, and evaluation budget. GWO and WOA perform very strongly in this experiment because their update mechanisms progressively reduce exploratory movement and concentrate the population around leading solutions [4], [5]. This behavior is advantageous on the selected smooth and moderately multimodal continuous functions, but it can become a weakness if an engineering problem has disconnected feasible regions or requires sustained diversity late in the search. GA behaves differently because crossover and mutation continually create new combinations. Its final accuracy is lower than WOA/GWO on

three functions, yet it avoids the extreme failure pattern seen in the tested PSO configuration. GA is also attractive when engineering variables include integer choices, component selections, sequences, or logical decisions because encoding and custom operators can be tailored to the design representation. For purely real-valued design, DE remains theoretically and practically important; its difference-vector mutation is self-scaled to population geometry [2]. The relatively weak DE values here indicate that a short fixed-budget experiment can be sensitive to F, CR, population size, and the chosen mutation strategy.

PSO illustrates why repeated runs are essential. Its median performance can be acceptable even when its mean is inflated by a small number of unsuccessful runs. Engineering decisions based only on a best run would hide this reliability

issue. The literature documents many PSO adaptations involving inertia scheduling, constriction, neighborhood topology, and hybridization [8]. Therefore, the correct conclusion is not that canonical PSO is unsuitable, but that the tested parameterization needs additional safeguards if consistent performance is required.

The methodology also has limitations. First, four analytic benchmark functions cannot reproduce expensive simulation noise, equality constraints, discrete manufacturing choices, or multiobjective trade-offs. Second, the sample of eight runs is sufficient for a compact methodology demonstration but smaller than a comprehensive statistical campaign. Third, the implementation uses straightforward canonical variants; advanced adaptive versions could alter the ranking. Finally, wall-clock runtime is not emphasized because vectorized implementation details may bias timing, whereas many engineering studies are dominated by the cost of evaluating the underlying simulation model.

Conclusion

This paper developed a concise methodology for comparative analysis of metaheuristic optimization algorithms for engineering applications. Five algorithms—GA, PSO, DE, GWO, and WOA—were evaluated using four benchmark landscapes under a standardized population size, dimensionality, iteration limit, and repeated-seed protocol. The results show that WOA and GWO provide the best final objective accuracy under the selected settings, while GWO demonstrates particularly consistent behavior on the difficult Rosenbrock problem. GA offers moderate but comparatively stable performance and remains attractive when flexible solution encoding is required. DE and PSO should not be dismissed on the basis of the present results; instead, their sensitivity highlights the importance of parameter tuning, restart strategies, and application-specific validation. For engineering practice, the recommended workflow is to use benchmark screening only as the first stage, then test the strongest candidates on the actual constrained model using equal function-evaluation budgets. Future studies can extend the framework to welded-beam, pressure-vessel, truss, economic-dispatch, controller-tuning, and scheduling problems; include nonparametric significance tests; and compare adaptive or hybrid variants. The key methodological requirement is reproducibility: algorithm settings, seeds, evaluation counts, constraint rules, and stopping criteria should be reported explicitly so that

performance claims remain technically meaningful.

Recommended Selection Rule

For a new continuous engineering design problem, GWO or WOA can be used as an initial low-parameter baseline, while DE should be retained as a strong conventional comparator and GA should be included when mixed or discrete variables are important. PSO is appropriate when fast population communication is desirable, but robustness should be checked through repeated runs. No method should be selected solely because it performed best on one benchmark; selection should be based on accuracy, variance, feasibility handling, computational budget, and consistency on the actual engineering model.

References

- Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, 4, 1942–1948.
<https://doi.org/10.1109/ICNN.1995.488968>
- Storn, R., & Price, K. (1997). Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11, 341–359.
<https://doi.org/10.1023/A:1008202821328>
- Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 39, 459–471. <https://doi.org/10.1007/s10898-007-9149-x>
- Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey Wolf Optimizer. *Advances in Engineering Software*, 69, 46–61.
<https://doi.org/10.1016/j.advengsoft.2013.12.007>
- Mirjalili, S., & Lewis, A. (2016). The Whale Optimization Algorithm. *Advances in Engineering Software*, 95, 51–67.
<https://doi.org/10.1016/j.advengsoft.2016.01.008>
- Mirjalili, S. (2015). Moth-Flame Optimization Algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems*, 89, 228–249.
<https://doi.org/10.1016/j.knosys.2015.07.006>
- Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for

engineering design problems. *Advances in Engineering Software*, 114, 163–191. <https://doi.org/10.1016/j.advengsoft.2017.07.002>

Freitas, D., Lopes, L. G., & Morgado-Dias, F. (2020). Particle Swarm Optimisation: A historical review up to the current developments. *Entropy*, 22(3), 362. <https://doi.org/10.3390/e22030362>

Yang, X.-S., & Deb, S. (2009). Cuckoo Search via Lévy flights. *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, 210–214. <https://doi.org/10.1109/NABIC.2009.5393690>

Ezugwu, A. E., Adeleke, O. J., Akinyelu, A. A., & Viriri, S. (2020). A conceptual comparison of several metaheuristic algorithms on continuous optimisation problems. *Neural Computing and Applications*, 32, 6207–6251. <https://doi.org/10.1007/s00521-019-04132-w>