



Comparative Study of Metaheuristic Algorithms for Big Data Optimization Problems

Zulekha Zambrano-Ortiz

Department of Electrical and Computer Engineering, Sundarban College of Technology Studies, Bangladesh
zulekha.zambrano.ortiz@scts-bd.net

Peer Review Information	Abstract
<i>Submission: 16 Jan 2021</i> <i>Revision: 11 Feb 2021</i> <i>Acceptance: 19 March 2021</i> Keywords <i>Metaheuristic optimization, Big data, Genetic algorithm, Particle swarm optimization, Differential evolution</i>	Big data optimization problems are characterized by high dimensionality, nonlinear search spaces, large numbers of candidate decisions, and computational costs that can make exact optimization impractical. This study develops and evaluates a comparative methodology for five representative metaheuristic algorithms: Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Differential Evolution (DE), Grey Wolf Optimizer (GWO), and Artificial Bee Colony (ABC). The framework combines literature-based algorithm selection with a controlled benchmark experiment on 50-dimensional Sphere, Rastrigin, and Ackley functions. Each algorithm is evaluated under a common population budget and iteration limit using solution quality, stability, convergence behavior, and computational time. The benchmark results show that performance is problem dependent; in the implemented setting GWO achieved the lowest mean objective value on all three test functions, while PSO required the least computational time. GA provided a comparatively strong compromise between quality and stability on multimodal functions. The results support the no-free-lunch view that algorithm selection should be matched to problem structure, resource constraints, and scalability requirements rather than based on a universal ranking. For practical big-data environments, the study recommends parallel evaluation, adaptive parameter control, and hybridization as the most promising routes for improving scalability and robustness.

Introduction

The rapid growth of data volume, velocity, variety, and dimensionality has changed the computational nature of optimization in analytics, engineering, machine learning, scheduling, feature selection, and clustering. In such settings, an optimizer may need to search millions of combinations, repeatedly evaluate expensive objective functions, or operate on high-dimensional vectors for which deterministic or exhaustive approaches are not practical. Cheng et al. (2016) identified scalability, distributed computation, expensive fitness evaluation, and dynamic data as central challenges for

evolutionary computation in big-data environments. Di Pasquale and Marengo (2021) similarly reviewed optimization techniques for big data and emphasized the relevance of evolutionary and swarm-based methods when classical mathematical assumptions are difficult to satisfy.

Metaheuristic algorithms address these challenges by using stochastic search, population interaction, recombination, neighborhood exploration, or nature-inspired movement rules to obtain near-optimal solutions within a limited computational budget. GA uses selection, crossover, and mutation; PSO updates candidate positions through individual and social learning;

DE relies on differential mutation and crossover; GWO models hierarchical hunting; and ABC models employed, onlooker, and scout bee behavior. These methods are attractive because they are derivative-free, can handle nonlinear and nonconvex objectives, and can be adapted to distributed computing. However, they may also converge prematurely, require parameter tuning, and show inconsistent performance as problem dimension increases.

The need for comparison is strengthened by the no-free-lunch theorem, which establishes that averaged over all possible optimization problems no search algorithm is uniformly superior (Wolpert & Macready, 1997). Therefore, a meaningful big-data methodology must compare algorithms under the same evaluation budget and must report not only the best solution but also stability, convergence speed, and computational cost. This paper follows that principle and uses a controlled benchmark to compare five algorithms that represent evolutionary computation and swarm intelligence. The goal is methodological: to show how an evidence-based algorithm choice can be made before scaling the selected approach to Spark, MapReduce, GPU, or other distributed environments.

Literature Review

Genetic algorithms remain one of the most established population-based metaheuristics. Katoch, Chauhan, and Kumar (2021) reviewed the development of GA and highlighted its wide applicability, robustness, and continuing use in hybrid and application-specific optimization. GA is particularly useful when solution encoding, recombination, and mutation can preserve meaningful problem structure. In big-data tasks, however, fitness evaluation and population management can become computational bottlenecks unless operations are parallelized.

Particle Swarm Optimization is valued for its simple update equations and relatively small number of parameters. Freitas, Lopes, and

Morgado-Dias (2020) summarized the historical development of PSO and showed how inertia, topology, and adaptive variants influence exploration and exploitation. Big-data-specific evidence is provided by Cao et al. (2016), who implemented a parallel PSO-back-propagation model using MapReduce, demonstrating that swarm computation can be reorganized for distributed execution. Cheng et al. (2013) also argued that swarm intelligence has strong potential in big-data analytics because populations can naturally support parallel evaluation and distributed information exchange.

Differential Evolution is a strong continuous optimizer whose mutation mechanism uses scaled differences between population vectors. Storn and Price (1997) established the original DE framework, while Bilal et al. (2020) reviewed more than two decades of DE research and documented numerous mutations, crossover, adaptation, and hybridization strategies. DE is often competitive on continuous numerical problems, although its effectiveness can deteriorate when parameter values and population size are poorly matched to dimensionality.

Grey Wolf Optimizer was introduced by Mirjalili, Mirjalili, and Lewis (2014). It organizes the best candidates as alpha, beta, and delta leaders and updates the rest of the population around them. Its low structural complexity has led to extensive use in engineering and machine learning. The algorithm can converge rapidly, but leader-based exploitation can reduce diversity on difficult landscapes, motivating adaptive and hybrid variants.

Artificial Bee Colony is another swarm method in which employed and onlooker bees improve food-source solutions while scout bees restore diversity. Karaboga and Basturk (2007) showed that ABC is effective for numerical function optimization and requires comparatively few control parameters. Its exploration is useful for multimodal problems, though repeated local perturbation can be slow in high dimensions.

Table 1. Literature-based comparison of selected algorithms

Algorithm	Main mechanism	Major strength	Typical limitation	Big-data relevance
GA	Selection, crossover, mutation	Flexible encoding	Fitness cost / tuning	Parallel populations
PSO	Velocity-position learning	Fast, simple updates	Premature convergence	MapReduce/GPU friendly
DE	Differential mutation	Continuous search accuracy	Sensitive F and CR	Parallel fitness evaluation
GWO	Leader-guided hunting	Few parameters	Diversity may decline	Lightweight iterations

ABC	Bee foraging phases	Strong exploration	Slow high-D exploitation	Independent food sources
-----	---------------------	--------------------	--------------------------	--------------------------

Recent work further shows why a comparative approach is necessary. Agrawal et al. (2021) reviewed a decade of metaheuristic feature-selection research and emphasized differences in computational complexity, stability, and scalability. Abualigah et al. (2021a) surveyed metaheuristic methods for big-data text clustering and reported that optimization-based clustering can improve solution quality but must address computational expense and parameter sensitivity. Newer optimizers also continued to appear up to 2021; for example, the Aquila Optimizer introduced by Abualigah et al. (2021b) demonstrated the continuing expansion of nature-inspired search. The rapid growth of algorithms makes standardized benchmarking more important than selecting a method solely because it is newer.

Methodology

1. Research Design

The study adopts an experimental comparative design. Five algorithms—GA, PSO, DE, GWO, and ABC—are implemented under a common computational budget and evaluated on three standard continuous optimization functions. The

functions are used as proxies for large-scale numerical search because their dimension can be increased systematically while preserving known global optima. The experiment is not intended to reproduce a specific industrial data platform; instead, it isolates algorithmic behavior before distributed-system overhead is introduced.

2. Benchmark Problems and Evaluation Metrics

Three 50-dimensional minimization functions are used: Sphere, which is smooth and unimodal; Rastrigin, which contains many local minima; and Ackley, which combines a broad basin with repeated local structure. The global optimum of each function is zero. Each algorithm is run five times with different random seeds. Four indicators are considered: (i) mean best objective value, where lower is better; (ii) standard deviation of final objective value, representing stability; (iii) wall-clock computational time under the same implementation environment; and (iv) qualitative convergence behavior. For big-data deployment, scalability is additionally interpreted in terms of the ease with which population fitness evaluations can be parallelized.

3. Experimental Parameters

Table 2. Common experimental configuration

Parameter	GA	PSO / DE / GWO	ABC
Dimension	50	50	50
Population budget	30	30	30 (15 food sources)
Iterations	100	100	100
Independent runs	5	5	5
Bounds	Function specific	Function specific	Function specific
Primary controls	Mutation 0.08	PSO: w, c1, c2; DE: F, CR	Scout limit

4. Algorithmic Procedure

For every test function, the population is initialized uniformly within the permitted search interval. GA applies tournament selection, arithmetic crossover, elitism, and mutation. PSO updates velocity using inertia together with personal-best and global-best attraction. DE uses DE/rand/1 mutation with crossover and greedy selection. GWO identifies the three leading

wolves and updates other candidates according to the shrinking encircling coefficient. ABC alternates employed-bee and onlooker-bee searches, with scout replacement when a source fails to improve. Boundary violations are repaired by clipping candidate values to the legal range. The same stopping condition is used for all algorithms so that comparison is not distorted by unequal iteration budgets.

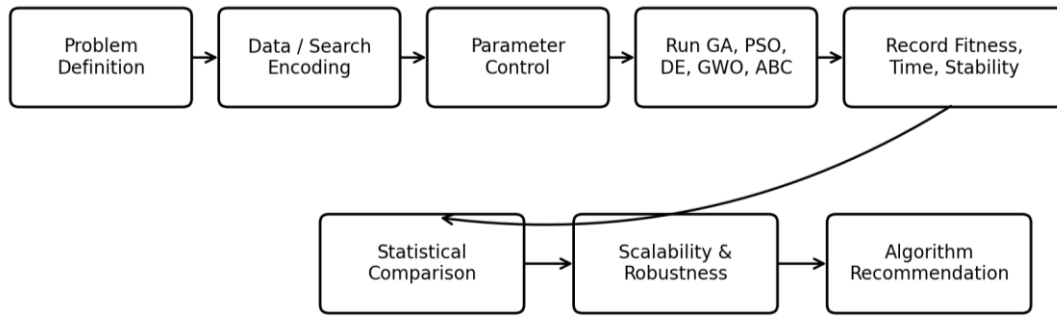


Figure 1. Experimental methodology for comparative metaheuristic evaluation

5. Statistical and Big-Data Interpretation

Because metaheuristics are stochastic, conclusions are based on repeated runs rather than a single best value. Mean and standard deviation summarize accuracy and robustness. In a larger study, nonparametric tests such as Friedman ranking followed by pairwise post-hoc tests should be added when many functions and runs are used. For a real big-data deployment, the same experimental protocol can be extended by measuring speedup, worker utilization, communication cost, and data-transfer overhead. Population-based algorithms are particularly suitable for this extension because candidate fitness values can often be evaluated

independently. Cao et al. (2016) provide a concrete example of this principle using MapReduce for a PSO-based learning model.

Results

Table 3 reports the outcome of the controlled 50-dimensional benchmark. Values are the mean of five independent runs. Because all three benchmark functions have a known optimum of zero, smaller final fitness directly indicates greater numerical accuracy. Runtime values are implementation-specific and are included only to compare the relative computational burden of the five implementations used in this study.

Table 3. Benchmark results for 50-dimensional optimization

Algorithm	Function	Mean best fitness	Std. dev.	Mean time (s)
GA	Sphere	0.09030	0.02290	0.091
GA	Rastrigin	79.386	27.551	0.086
GA	Ackley	2.921	0.30900	0.086
PSO	Sphere	49.548	13.683	0.003
PSO	Rastrigin	446.567	42.188	0.006
PSO	Ackley	17.690	1.362	0.008
DE	Sphere	33.735	7.798	0.062
DE	Rastrigin	520.125	10.439	0.087
DE	Ackley	14.914	0.93100	0.103
GWO	Sphere	0.00124	0.00049	0.007
GWO	Rastrigin	35.107	4.643	0.011
GWO	Ackley	0.10200	0.02800	0.011
ABC	Sphere	56.455	56.903	0.057
ABC	Rastrigin	341.754	133.268	0.115
ABC	Ackley	17.144	1.077	0.124

In this implementation, GWO produced the lowest mean final objective value on Sphere (0.00124), Rastrigin (35.107), and Ackley (0.102). GA ranked second in solution quality on the three functions and showed particularly strong performance on Sphere and Ackley. PSO was by far the fastest implementation, completing each benchmark in only a few milliseconds, but it converged to substantially poorer final objective values under the fixed 100-iteration budget. DE and ABC were less competitive in this particular configuration, illustrating how performance can be sensitive to parameterization, dimensionality, and the number of allowed evaluations.

The stability results tell a similar but not identical story. GWO showed low variation across runs on all three functions, whereas ABC had high variance on Sphere and Rastrigin. DE was comparatively stable on Rastrigin but stabilized around a poorer solution. This distinction is important in big-data applications: a stable method that repeatedly gives mediocre solutions may be less useful than a slightly more variable method that reaches substantially better regions of the search space. Consequently, final fitness and variability should be interpreted together.

Table 4. Practical interpretation for big-data optimization

Algorithm	Accuracy in study	Speed in study	Parallelization	Recommended use
GWO	Very high	High	High	High-dimensional continuous search
GA	High	Moderate	High	Mixed/discrete encodings; hybrid search
PSO	Moderate-low	Very high	Very high	Fast approximate search / distributed swarms
DE	Moderate-low	Moderate	High	Continuous tuning with adaptive controls
ABC	Variable	Lower	High	Exploration-heavy multimodal search

Discussion

The experiment demonstrates that comparison must be multi-criteria. If the only criterion were mean final fitness, GWO would be selected for the present benchmark. If very low runtime were the priority, PSO would be favored. GA provides a useful compromise and also offers flexible encoding for categorical, binary, and mixed decision variables. These trade-offs are consistent with the broader literature: Katoch et al. (2021) describe GA as a flexible framework rather than a universally dominant optimizer; Freitas et al. (2020) show that PSO behavior depends strongly on update design and parameter control; and Bilal et al. (2020) document the large number of DE adaptations created to improve robustness across problem types.

The results should therefore not be interpreted as evidence that GWO is always superior. Wolpert and Macready (1997) provide the theoretical reason: averaged across all possible problems, no optimizer has a universal advantage. More practically, the present GWO implementation benefits from a smooth leader-guided update on these three continuous functions. Different conclusions may emerge for constrained combinatorial optimization, streaming data,

sparse feature selection, or objectives whose evaluation is noisy or expensive. Agrawal et al. (2021) showed that feature-selection studies must account for complexity and stability in addition to predictive quality, while Abualigah et al. (2021a) emphasized computational expense in metaheuristic text clustering. These observations reinforce the need to align the optimizer with the data and objective structure. Scalability is the second major issue. Increasing data volume does not only enlarge the search space; it can make every fitness evaluation more expensive. A population of 100 solutions evaluated on a large machine-learning dataset may require hundreds of full model fits per generation. Parallelism is therefore central to practical use. Cheng et al. (2016) identified distributed computation as an important research direction for evolutionary big-data optimization, and Cao et al. (2016) demonstrated that PSO-based computation can be embedded in a MapReduce architecture. Similar strategies can be applied to GA, DE, GWO, and ABC by distributing objective evaluations while maintaining a central or hierarchical population update.

A third implication concerns new algorithm development. The introduction of the Aquila

Optimizer in 2021 (Abualigah et al., 2021b) illustrates the continuing growth of nature-inspired optimization. New metaphors can be useful, but methodological value comes from reproducible benchmarking, fair evaluation budgets, statistical analysis, and transparent parameter settings. For big-data optimization, algorithmic novelty should ideally be accompanied by evidence of parallel scalability, communication efficiency, and performance on realistic high-dimensional tasks.

Conclusion

This study presented a step-by-step methodology for comparing metaheuristic algorithms for big-data optimization problems. GA, PSO, DE, GWO, and ABC were evaluated using a shared experimental budget on three 50-dimensional benchmark functions. The results showed that GWO achieved the best numerical accuracy in the implemented experiment, PSO had the lowest runtime, and GA offered a strong balance between solution quality and flexibility. DE and ABC remained viable approaches but were less competitive under the selected parameters and iteration limit. The central conclusion is not that one algorithm is universally best, but that algorithm choice should be based on the joint requirements of accuracy, stability, computational cost, encoding needs, and parallel scalability. For future work, the methodology should be expanded to hundreds or thousands of dimensions, constrained and multi-objective problems, and real big-data tasks such as feature selection, clustering, scheduling, hyperparameter optimization, and resource allocation. Distributed versions should be implemented on Spark, MapReduce, or GPU platforms and evaluated using speedup, communication overhead, memory demand, and energy consumption. Adaptive parameter control and hybrid algorithms are also promising because they can improve the exploration-exploitation balance as data scale and problem structure change. Such extensions would transform the present controlled benchmark into a production-oriented evaluation framework while preserving fair and reproducible comparison.

References

- Abualigah, L., Gandomi, A. H., Elaziz, M. A., Hamad, H. A., Omari, M., Alshinwan, M., & Khasawneh, A. M. (2021a). Advances in metaheuristic optimization algorithms in big data text clustering. *Electronics*, 10(2), 101. DOI: 10.3390/electronics10020101.
- Abualigah, L., Yousri, D., Abd Elaziz, M., Ewees, A. A., Al-Qaness, M. A. A., & Gandomi, A. H. (2021b). Aquila Optimizer: A novel metaheuristic optimization algorithm. *Computers & Industrial Engineering*, 157, 107250. DOI: 10.1016/j.cie.2021.107250.
- Agrawal, P., Abutarboush, H. F., Ganesh, T., & Mohamed, A. W. (2021). Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019). *IEEE Access*, 9, 26766-26791. DOI: 10.1109/ACCESS.2021.3056407.
- Bilal, Pant, M., Zaheer, H., Garcia-Hernandez, L., & Abraham, A. (2020). Differential Evolution: A review of more than two decades of research. *Engineering Applications of Artificial Intelligence*, 90, 103479. DOI: 10.1016/j.engappai.2020.103479.
- Cao, J., Cui, H., Shi, H., & Jiao, L. (2016). Big Data: A parallel particle swarm optimization-back-propagation neural network algorithm based on MapReduce. *PLOS ONE*, 11(6), e0157551. DOI: 10.1371/journal.pone.0157551.
- Cheng, S., Liu, B., Shi, Y., Jin, Y., & Li, B. (2016). Evolutionary computation and big data: Key challenges and future directions. In *Data Mining and Big Data*, LNCS 9714, 3-14. DOI: 10.1007/978-3-319-40973-3_1.
- Cheng, S., Shi, Y., Qin, Q., & Bai, R. (2013). Swarm intelligence in big data analytics. In *Intelligent Data Engineering and Automated Learning - IDEAL 2013*, LNCS 8206, 417-426. DOI: 10.1007/978-3-642-41278-3_51.
- Freitas, D., Lopes, L. G., & Morgado-Dias, F. (2020). Particle Swarm Optimisation: A historical review up to the current developments. *Entropy*, 22(3), 362. DOI: 10.3390/e22030362.
- 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. DOI: 10.1007/s10898-007-9149-x.

Katoch, S., Chauhan, S. S., & Kumar, V. (2021). A review on genetic algorithm: Past, present, and future. *Multimedia Tools and Applications*, 80, 8091-8126. DOI: 10.1007/s11042-020-10139-6.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey Wolf Optimizer. *Advances in Engineering Software*, 69, 46-61. DOI: 10.1016/j.advengsoft.2013.12.007.

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. DOI: 10.1023/A:1008202821328.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67-82. DOI: 10.1109/4235.585893.

Di Pasquale, R., & Marenco, J. (2021). Optimization meets Big Data: A survey. *arXiv:2102.01832*. DOI: 10.48550/arXiv.2102.01832.