Vol. 8, No. 4, August 2026

Journal of Information Systems and Informatics

Published By Asosiasi Doktor Sistem Informasi Indonesia

Application-Driven Parallel Differential Evolution: A Systematic Mapping Review of Scalable Optimization Applications

Said Iskandar Al Idrus[1], Rudi Setiawan[2]

[1]Faculty of Mathematics and Sciences, State University of Medan, Medan, Indonesia

[2]Information System Department, Trilogi University, Jakarta, Indonesia

Received:
February 24, 2026
Revised:
June 3, 2026
Accepted:
July 22, 2026
Published:
August 22, 2026

Corresponding Author:
Author Name*:
Said Iskandar Al Idrus
Email*:
saidiskandar@unimed.ac.id

DOI:
10.63158/journalisi.v8i4.1695

© 2026 Journal of Information Systems and Informatics. This open access article is distributed under a (CC-BY License)

Abstract. This systematic mapping review examines application-driven parallel Differential Evolution (DE) as a scalable optimization approach for engineering, energy, computing, and intelligent systems. The review is based on a Scopus-only corpus searched on 24 May 2026 using TITLE-ABS-KEY queries related to DE, parallel computing, distributed computing, GPU acceleration, and scalable optimization. From 220 records, 25 unique studies published between 2021 and 2026 were included after screening by year, document type, language, relevance, and methodological alignment. Because the corpus contains heterogeneous benchmark studies, application studies, and hybrid intelligent-system frameworks, the evidence was synthesized thematically rather than through meta-analysis. The synthesis distinguishes explicit parallel-DE implementations from broader scalable DE applications in which scalability is achieved through decomposition, model reformulation, or integration with computationally expensive systems. The findings indicate that GPU acceleration, CUDA, MPI migration, cooperative coevolution, Spark/Hadoop distribution, and resource-aware dispatch frequently report reduced computational cost while preserving or improving solution quality under the evaluated conditions. However, cross-study comparison remains limited by heterogeneous benchmarks, incomplete hardware reporting, inconsistent scalability metrics, and uneven baseline selection. This review contributes a cross-domain taxonomy linking application constraints, DE mechanisms, computational architectures, evaluation metrics, and reported outcomes.

Keywords: Differential Evolution, Parallel Computing, GPU Computing, Distributed Optimization, Scalability

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

1. INTRODUCTION

Differential Evolution (DE) is a population-based evolutionary algorithm for continuous, nonlinear, multimodal, and derivative-free optimization. Its basic search logic improves candidate solutions through mutation, crossover, and selection, enabling global exploration and local exploitation within a comparatively simple control-parameter structure [1]. Recent review literature confirms that DE remains influential in engineering design, computational intelligence, data analytics, power systems, and scientific computing because it is robust, flexible, and compatible with black-box objective functions [2], [3].

Although DE is robust, its population-based structure can become computationally expensive in modern application settings. Large decision spaces, high-dimensional benchmark suites, simulation-based objective functions, deep-learning architectures, distributed infrastructures, and near-real-time operational constraints may require many fitness evaluations. In such cases, sequential DE may be insufficient because evaluation cost, data movement, and hardware constraints become part of the optimization problem itself.

This review uses five related but distinct concepts. Parallel DE refers to implementations in which individuals, subpopulations, or operators are executed concurrently. Distributed DE refers to DE running across multiple nodes, workers, clusters, or distributed data platforms. GPU-accelerated DE refers to DE implementations that exploit massively parallel GPU kernels such as CUDA. Scalable DE is a broader term that includes parallel execution, distributed execution, decomposition, resource-aware scheduling, or model reformulation to handle computational growth. DE used inside a computationally expensive application refers to studies where DE is not always the only parallel component, but the optimization task is shaped by simulation, learning, data-processing, or operational constraints.

Four scalability mechanisms guide the conceptual framework of this review: hardware acceleration, distributed population execution, problem decomposition, and application-aware resource management. Hardware acceleration includes CUDA/GPU and many-threaded computation. Distributed population execution includes MPI, island migration,

Spark, Hadoop, cloud, and edge settings. Problem decomposition includes cooperative coevolution and differential grouping for large-scale optimization. Application-aware resource management includes cases where DE is embedded in intelligent systems, neural controllers, predictive models, or domain-specific computational pipelines.

Previous DE surveys have provided broad algorithmic histories, parameter-control analyses, and general reviews of DE variants [2], [3], [4]. Surveys on GPU-based swarm intelligence and cooperative coevolution have also explained hardware and decomposition perspectives [5], [6], [7]. However, fewer reviews organize recent DE studies from the perspective of application-driven scalability across engineering, energy, computing, big data, and intelligent-system domains. The originality of this paper lies in mapping application constraints to DE mechanisms, computational architectures, evaluation metrics, and reported outcomes rather than ranking DE variants universally. The review period 2021-2026 was selected because it captures recent developments in GPU acceleration, distributed data-processing platforms, edge/cloud computing, neural architecture search, and post-2020 reproducibility expectations in computational optimization. The corpus includes both benchmark-only studies and real-world application studies, but they are not compared as if they had identical evidential weight. Instead, benchmark studies are used to understand algorithmic scalability, whereas application studies are used to understand domain-level feasibility and value.

This review addresses the following research questions: RQ1: How have recent studies implemented or contextualized parallel, distributed, GPU-accelerated, and scalable DE across application domains? RQ2: Which application constraints motivate the use of scalable DE mechanisms? RQ3: Which computational architectures and DE mechanisms are associated with particular problem characteristics? RQ4: What reporting, reproducibility, validity, and sustainability gaps remain in application-driven scalable DE research?

2. METHODS

2.1 Review Design and Search Strategy

This article adopts a systematic mapping review design with PRISMA-informed reporting. The design was selected because the aim is to classify and synthesize heterogeneous evidence across applications, mechanisms, architectures, and metrics rather than to

estimate a pooled quantitative effect. Consequently, the synthesis is thematic and classificatory, not a meta-analysis of speedup or solution-quality effects [8], [9].

Scopus was the only database used in this review. It was selected because it provides broad interdisciplinary coverage of engineering, computer science, energy, applied optimization, and intelligent systems. The search was executed on 24 May 2026 using the following field-based query: TITLE-ABS-KEY ("differential evolution" AND ("parallel programming" OR GPU OR CUDA OR OpenMP OR MPI OR multicore OR "distributed computing" OR "island model" OR "cooperative coevolution" OR "many-threaded" OR "heterogeneous computing" OR Spark OR Hadoop OR cloud OR edge OR acceleration) AND (optimization OR "global optimization" OR "computational performance" OR scalability OR acceleration)).

Supplementary query variants were used inside Scopus to improve recall for application terms including engineering, smart grid, microgrid, big data, feature selection, neural networks, generative adversarial networks, resource allocation, and intelligent control. These variants are summarized in Appendix A for transparency. The final selection remained restricted to records indexed in Scopus and falling within the 2021-2026 publication window. Early-access and 2026 records were retained only when bibliographic metadata, DOI, or publisher records were available at the time of search.

2.2 Eligibility Criteria and Study Types

Eligible studies had to use DE or a DE-based hybrid method as a meaningful optimization component and had to address at least one scalability mechanism: explicit parallel execution, distributed execution, GPU acceleration, cooperative decomposition, resource-aware dispatch, cloud/edge/fog execution, Spark/Hadoop implementation, or application-aware computational reformulation. The review therefore includes two evidence categories: core parallel-DE studies and contextual scalable-DE studies. Core studies explicitly implement or evaluate parallel DE. Contextual studies use DE inside computationally demanding applications where scalability is achieved indirectly through modelling, decomposition, or hybrid integration.

Studies were excluded if DE was mentioned only incidentally, if scalability was not substantively related to the method or application, if the work focused exclusively on

unrelated evolutionary algorithms, if essential bibliographic metadata were unavailable, or if no relevant performance indicator was reported. The expected outcomes of scalability were runtime reduction, speedup, convergence behavior, solution quality, accuracy, energy consumption, GPU-hour efficiency, real-time feasibility, or domain-specific operational performance.

2.3 Screening, PRISMA Counts, Extraction, and Quality Assessment

The initial Scopus search identified 220 records. No duplicates were removed before screening because the corpus was exported from a single database. During title-and-abstract screening, 166 records were excluded: 54 were outside the 2021-2026 period, 49 failed the document-type, language, or peer-reviewed-source criteria, and 63 were outside the topical focus. A total of 54 reports were retrieved and assessed for eligibility. After full-text assessment, 29 reports were excluded: 12 did not use DE as a core method, 7 did not include a scalable or computationally expensive mechanism, 4 lacked relevant performance indicators, and 6 did not align with the target application scope. The final corpus contained 25 unique included studies.

Screening, coding, and reporting-quality assessment were conducted collaboratively by both authors using predefined inclusion criteria and a shared extraction sheet rather than as an independent double-screening process. Coding decisions were resolved through discussion and rechecking against the eligibility criteria. The extraction fields were year, domain, optimization task, DE variant or mechanism, parallelization strategy, hardware or platform, benchmark or dataset, runtime metric, solution-quality metric, limitation, and whether the study represented core parallel DE or contextual scalable DE. Reporting quality was assessed using criteria adapted to computational optimization: clarity of algorithm description, parameter settings, benchmark or dataset description, hardware/platform specification, runtime or scalability reporting, baseline fairness, random-seed or repeated-run reporting, communication overhead, code availability, and energy-consumption reporting. This assessment evaluates reporting completeness and comparability; it is not a clinical or experimental risk-of-bias exclusion tool. These criteria are used to interpret the strength and comparability of the evidence [10].

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

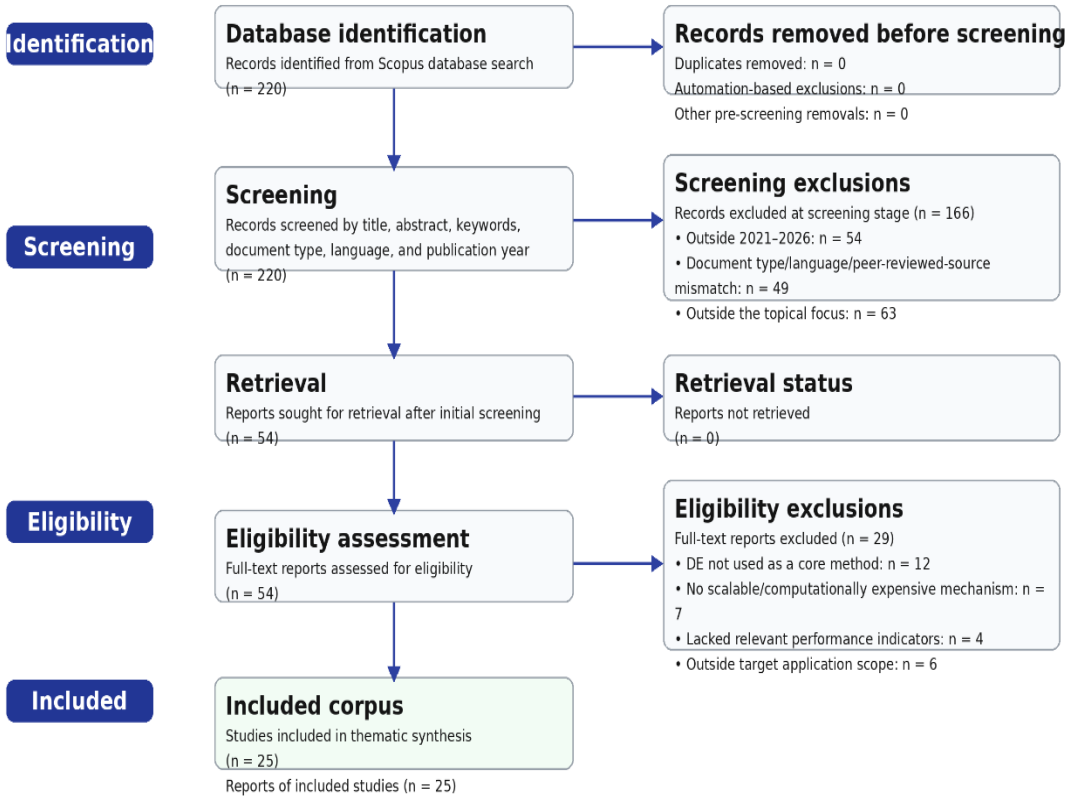

Figure 1. PRISMA-Informed Flow of Study Selection

2.4. Theoretical Background

DE exposes concurrency at multiple levels. Fitness evaluations can be executed independently, individuals can be evaluated in parallel, subpopulations can evolve in island models, and high-dimensional variables can be decomposed into smaller subcomponents. This structure explains why DE is frequently paired with GPU acceleration, MPI, Spark/Hadoop, cooperative coevolution, and resource-aware scheduling [5], [11], [12], [13].

Adaptive DE variants manage the balance between diversity and selection pressure by adjusting control parameters, mutation strategies, or crossover rates. Classical self-adaptive and strategy-adaptive DE studies remain important for foundational theory [14], [15], [16], while recent adaptive-DE surveys show that contemporary DE research increasingly treats parameter control, strategy selection, population resizing, and learning mechanisms as dynamic design choices [4].

In scalable DE, computation is not merely an external implementation issue. Hardware architecture, data-transfer cost, communication overhead, decomposition quality, baseline fairness, and energy consumption can influence both runtime and search behavior. Therefore, this review treats scalable DE as an application-aware computational framework rather than simply a faster version of sequential DE.

3. RESULT AND DISCUSSION

3.1. Corpus Overview and Cross-Tagging Rule

The final corpus contained 25 unique included studies. Because several studies address more than one application domain or scalability mechanism, thematic categories are cross-tagged rather than mutually exclusive. For example, a resource-aware DE study for neural-network controllers can be tagged as engineering, intelligent control, and distributed optimization. Percentages in domain and mechanism tables therefore indicate representation within the corpus but may sum to more than 100% .

Table 1. Master evidence table of the 25 unique included studies.

Ref.	Year	Domain	Optimization task	DE /scalable mechanism	Platform	Benchmark /dataset	Runtime metric	Quality metric /outcome	Main limitation
[17] Liu et al. (2022)	2022	Engineering / control	NN controller training	Resource-aware distributed DE	Distributed heterogeneous resources	Power electronic circuit controller	Runtime, solution quality	Improved efficiency and quality	Platform reproducibility limited
[18] Fatigate et al. (2023)	2023	Engineering / medical simula tion	Hypertherm ia planning	CUDA-supported DE	CUDA/GPU PDE solver	3D bioheat simulation	Damage indicators, runtime feasibility	Tractable treatment planning	Clinical validation limited
[11] Kelkawi et al. (2023)	2023	Computing / LSGO	Large-scale global optimizatio n	GPU cooperative coevolution	GPU	CEC 2010 benchmarks	Speedup, quality	Speedups up to 13.01	Benchma rk-focused
[19] Wang et al. (2022)	2022	Energy	Wind-wave layout optimizatio n	GPU-accelerated DE	GPU	Hybrid renewable layout	Energy output, wave load, speedup	Higher output and 1136x speedup	Hardware-dependent
[20] Li et al. (2025)	2025	Energy / microgrid	Multi-period dispatch	Small-population parallel DE	MPI gather/scatter, migration	Microgrid dispatch	Runtime, objective value	Reduced time with maintained quality	Deploym ent evidence limited

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

Ref.	Year	Domain	Optimization task	DE /scalable mechanism	Platform	Benchmark /dataset	Runtime metric	Quality metric /outcome	Main limitation
[21] Thakur et al. (2022)	2022	Big data	Feature selection	RST-DE	Hadoop /Spark	Five datasets	Accuracy, reduction, scalability	Improved scalable feature selection	Dataset /classifier depen dence
[22] Ma et al. (2024)	2024	Edge/ cloud	Autoscaling/ offloading	DE-supported LSTM tuning	Cloud/edge framework	Resource-manage ment scenario	Execution time, energy, delay	Improved managemen t indicators	DE is auxiliary
[23] Ghahremani et al. (2025)	2025	Fog/IoT	Task allocation	Aquila-Differential Optimizer	Fog/IoT simulation	Mobile crowd sensing	Coverage, cost, reliability	Better QoS indicators	Simula tion-based
[24] Rafique et al. (2026)	2026	Genera tive models	GAN architecture search	DE architecture search	GPU training environ ment	Image datasets	FID, IS, GPU hours	Competitive scores with lower burden	Architect ure-specific
[25] Feng et al. (2026)	2026	Robotics / bench mark	Strategy diversity	Individual-level strategy diversity	GPU-oriented execution	Benchmarks and control	Performanc e, parallel suitability	Communica tion-free concurrenc y	Broader validation needed
[13] El-Abd et al. (2024)	2024	Distri buted compu ting	LSGO	Spark cooperative coevolution	Apache Spark cluster	Global optimiza tion benchmarks	Speedup, quality	Distributed speed gains	Cluster overhead underrep orted
[26] Pan et al. (2022)	2022	High-dimensional compu ting	Continuous optimizatio n	NEC-based parallel DE	MKL/CUDA	High-dimensional benchmarks	Runtime, accuracy	Improved computa tional performanc e	Applica tion transfer limited
[27] Dufek et al. (2026)	2026	Bilevel optimiza tion	Expensive bilevel search	Hierarchical many-threaded DE	GPU many-threading	Bilevel benchmarks	Runtime speedup, quality	Large speedups reported	Hardware -depen dent
[28] Chen et al. (2026)	2026	Meta-optimiza tion	DE self-configuratio n	MetaDE	GPU accelera tion	DE configuratio n tasks	Performanc e, configuratio n quality	Self-configures DE	Complexi ty limits reproduci bility
[29] Laguna-Sanchez et al. (2025)	2025	GPU compu ting	DE vs PSO comparison	CUDA DE	GPU	Benchmark functions	Runtime, conver gence	Runtime reduced	Conver gence behavior changes
[30] Klippel et al. (2026)	2026	Manufacturing	Parameter identificatio n	DE with GPU-accelerated SPH	GPU simulation	Ti6Al4V machining	Identificatio n quality	Strong identifi cation results	Model transfer unce rtainty

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

Ref.	Year	Domain	Optimization task	DE /scalable mechanism	Platform	Benchmark /dataset	Runtime metric	Quality metric /outcome	Main limitation
[31] Ha et al. (2022)	2022	Structu ral enginee ring	Truss sizing	Cooperative multi-search DE	CUDA + island migration	Truss structures	Runtime, design quality	At least twofold faster	Speed-quality trade-off
[32] Lin et al. (2024)	2024	Manufacturing intelli gence	Machining parameter search	Model-assisted DE	Predictive modelling framework	Machining data	Prediction/optimiza tion indicators	Interpretable optimizatio n	Parallel execution not central
[33] Zhou et al. (2022)	2022	Circuit design	FPRM power optimizatio n	Hybrid DE metaheurist ic	Computational optimization pipeline	Logic circuit benchmarks	Power reduction	Improved circuit power	Parallel details limited
[34] Parpinelli et al. (2021)	2021	Bioinformatics	Protein structure prediction	Speciation-based DE	Massively parallel GPU	3D-AB protein model	Speedup, diversity	Speedup up to 708.78	Represen tation-specific
[35] Li et al. (2024)	2024	Smart grid	Voltage measureme nt	Knowledge-assisted DE	Model-guided search	Multiconduc tor systems	Accuracy, robustness	Improved measureme nt quality	Scalabilit y knowledg e-guided
[36] Elbassoussi et al. (2026)	2026	Energy/desalin a tion	RO process optimizatio n	Dimensionle ss model-supported optimizatio n	Model reformula tion	Two-stage RO model	Cost/therm odynamic metrics	Operating envelopes identified	Explicit parallel DE not central
[37] Yang et al. (2023)	2023	Multitask optimization	Evaluation efficiency	Non-revisiting adaptive DE	CUDA-oriented scheme	Multitask benchmarks	Repeated evaluations, runtime	Improved efficiency	Applica tion specifi city limited
[38] Osuna-Enciso and Guevara-Martinez (2022)	2022	Decentralized metaheuristic s	DE without global best	Stigmergy-based DE	Decentralized design	Benchmarks	Conver gence	Maintains convergenc e	Paralle lism concep tual
[39] Shi et al. (2023)	2023	Model compression	Lithology identificatio n	DE pruning search	Learning-model pipeline	Lithology classes	Accuracy, model size	Compact deployment	Parallel DE not central

Table 2. Publication trend in the 2021-2026 corpus

Year	No. of studies	Percentage
2021	1	4%
2022	7	28%
2023	4	16%

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

2024	4	16%
2025	3	12%
2026	6	24%

Table 3. Domain distribution with cross-tagging.

Domain category	No. of studies	Percentage	Interpretive note
Engineering and simulation-based optimization	10	40%	Includes medical simulation, truss sizing, machining, circuit, and protein modelling.
Energy systems and smart infrastructure	8	32%	Includes wind-wave systems, microgrid dispatch, smart-grid measurement, RO desalination, and energy-aware edge/cloud.
Computing, big data, and distributed intelligent systems	12	48%	Includes LSGO, Spark/Hadoop, edge/cloud, fog/IoT, GPU benchmarks, and multitask optimization.
Neural networks, generative models, and intelligent control	10	40%	Includes NN controllers, GAN search, pruning, MetaDE, strategy diversity, and intelligent resource management.

Table 4. Core parallel-DE and contextual scalable-DE evidence categories.

Evidence category	No. of studies	Definition
Core explicit parallel-DE studies	16	Studies where parallel execution, GPU, CUDA, MPI, Spark/Hadoop, or many-threaded implementation is central to the method.

Evidence category	No. of studies	Definition
Contextual scalable-DE studies	9	Studies where DE is used in a computationally expensive, model-assisted, or application-aware setting, but explicit parallel execution is not the main contribution.

3.2. Mechanisms, Metrics, and Reporting Quality

The reviewed literature reveals substantial diversity in the mechanisms used to improve the scalability of Differential Evolution (DE). As summarized in Table 5, CUDA/GPU acceleration is the most frequently reported mechanism, appearing in 12 studies (48%). This finding indicates that fine-grained parallelization of population-based evolutionary operations remains a dominant strategy for reducing computational time, particularly when fitness evaluations or optimization procedures can be executed concurrently. Model-assisted or hybrid DE and contextual scalable DE approaches were each identified in six studies (24%), while cooperative coevolution or decomposition appeared in four studies (16%). MPI- or island-based migration and resource-aware or heterogeneous dispatch were each reported in three studies (12%), whereas Spark/Hadoop-based distribution appeared in two studies (8%). Only one study (4%) primarily relied on a small-population strategy as its scalability mechanism. Because several studies employed more than one mechanism, the percentages in Table 5 are not mutually exclusive and therefore do not sum to 100%.

Table 5. Frequency of DE scalability mechanisms.

Mechanism	No. of studies	Percentage
CUDA/GPU acceleration	12	48%
MPI or island migration	3	12%
Spark/Hadoop distribution	2	8%
Cooperative coevolution/decomposition	4	16%
Resource-aware or heterogeneous dispatch	3	12%
Small-population strategies	1	4%
Model-assisted or hybrid DE	6	24%
Contextual scalable DE, not explicit parallel DE	6	24%

The overlap among scalability mechanisms is an important characteristic of the corpus. For example, cooperative coevolution may decompose a large-scale optimization problem into subcomponents while simultaneously exploiting GPU acceleration for parallel fitness evaluation. Similarly, island-based DE can combine population migration with MPI-based communication, while application-oriented studies may integrate DE with Spark, Hadoop, machine-learning models, rough-set techniques, or heterogeneous computing resources. Consequently, scalability in the reviewed studies should not be interpreted solely as a property of a particular hardware platform. Instead, it emerges from the interaction among algorithmic structure, problem decomposition, computing architecture, and application-specific constraints. This multidimensional relationship forms the basis of the taxonomy presented later in Figure 2.

The evaluation practices identified in the corpus also demonstrate that scalability is assessed from multiple perspectives. As shown in Table 6, objective or solution quality is the most commonly reported evaluation criterion, appearing in 20 studies (80%), followed closely by runtime or execution time in 18 studies (72%). These results suggest that most researchers attempt to demonstrate computational improvement without sacrificing optimization effectiveness. Domain-specific operational outcomes are reported in 13 studies (52%), reflecting the application-driven nature of the reviewed literature, while explicit speedup ratios are presented in 10 studies (40%). Convergence behavior is examined in nine studies (36%), and accuracy, classification performance, or other model-quality measures are reported in eight studies (32%).

By contrast, relatively few studies examine the system-level costs associated with parallel or distributed computation. Table 6 shows that energy consumption is evaluated in only four studies (16%), communication overhead in three studies (12%), and GPU hours or training cost in two studies (8%). This imbalance is significant because a reduction in wall-clock execution time does not necessarily imply greater overall computational efficiency. A parallel DE implementation may achieve substantial speedup while simultaneously increasing communication traffic, energy consumption, hardware utilization, or infrastructure cost. Therefore, future evaluations of scalable DE would benefit from combining traditional optimization metrics with explicit measures of computational and resource efficiency.

Table 6. Frequency of evaluation metrics used in the corpus.

Evaluation metric	No. of studies	Percentage
Runtime or execution time	18	72%
Speedup ratio	10	40%
Objective or solution quality	20	80%
Convergence behavior	9	36%
Accuracy/classification/model quality	8	32%
Energy consumption	4	16%
Communication overhead	3	12%
GPU hours or training cost	2	8%
Domain-specific operational outcome	13	52%

A related issue concerns the completeness and consistency of experimental reporting. The reporting-quality assessment presented in Table 7 indicates that most studies provide adequate descriptions of the optimization problem and algorithmic approach. Twenty-four studies (96%) clearly describe their benchmark, dataset, or application context, while 23 studies (92%) provide a clear description of the DE algorithm or variant employed. Runtime or scalability measures are reported by 20 studies (80%), and parameter settings are documented in 17 studies (68%). These results indicate that the corpus is generally strong in describing what was optimized and which DE mechanism was used.

Table 7. Reporting-quality assessment summary

Reporting item	No. of studies	Percentage
Clear algorithm/variant description	23	92%
Parameter settings reported	17	68%
Benchmark/dataset/application context described	24	96%
Hardware/platform specified	16	64%
Runtime/scalability metric reported	20	80%
Baseline fairness clearly justified	15	60%
Random seeds/repeated-run protocol reported	9	36%
Communication/data-transfer overhead reported	5	20%
Code or implementation available	4	16%
Energy consumption reported	4	16%

However, Table 7 also reveals several weaknesses that affect reproducibility and cross-study comparison. Hardware or platform specifications are provided in only 16 studies (64%), while baseline fairness is clearly justified in 15 studies (60%). More importantly, only nine studies (36%) explicitly report random seeds or repeated-run protocols. Communication or data-transfer overhead is discussed in just five studies (20%), and implementation code is publicly available for only four studies (16%). Energy consumption is similarly reported by only four studies (16%). These limitations make it difficult to determine whether reported performance improvements originate from algorithmic advances, superior hardware, implementation-specific optimization, or differences in experimental configuration.

The hardware and software reporting patterns provide further evidence of this reproducibility challenge. As detailed in Table 8, GPU models are explicitly reported in 11 studies (44%), whereas CPU models are identified in only nine studies (36%). Software, compiler, or library versions are reported in eight studies (32%), memory specifications in six studies (24%), and detailed cluster configurations in only five studies (20%). Furthermore, only four studies (16%) provide access to implementation code. Such incomplete reporting is particularly problematic for parallel DE because performance can depend strongly on GPU architecture, CPU characteristics, memory hierarchy, interconnect bandwidth, compiler optimization, parallel libraries, and the number of available processing units. Consequently, nominal speedup values cannot always be directly compared across studies.

Table 8. Hardware and software reporting summary.

Hardware/software reporting item	No. of studies	Percentage
CPU model reported	9	36%
GPU model reported	11	44%
Cluster configuration reported	5	20%
Memory reported	6	24%
Software/compiler/library version reported	8	32%
Implementation code available	4	16%

Taken together, Tables 5–8 indicate that the current literature places greater emphasis on demonstrating optimization performance and execution-time improvements than on

providing standardized evidence of computational scalability. GPU-based acceleration represents the most visible implementation strategy, but scalable DE increasingly encompasses algorithmic decomposition, distributed processing, hybrid intelligent systems, and resource-aware execution. At the same time, heterogeneous evaluation metrics and incomplete hardware and software descriptions limit the comparability of reported results. These observations motivate the application-aware evaluation framework synthesized in Figure 2, which organizes scalable DE research according to the relationship among application constraints, DE mechanisms, computational architectures, evaluation metrics, and reported outcomes.

Figure 2 presents this taxonomy as an integrated representation of the evidence identified in the systematic mapping review. Rather than treating parallelism as an isolated implementation feature, the taxonomy conceptualizes scalable DE as a pipeline in which application requirements influence the choice of DE mechanism; the mechanism is mapped to a computational architecture; the resulting implementation is evaluated using computational, optimization, and domain-specific metrics; and the reported outcomes are interpreted within the quality and reproducibility of the experimental evidence. This structure provides a more systematic basis for comparing explicit parallel-DE implementations with broader application-driven DE studies in which scalability is achieved through decomposition, hybridization, distributed computing, or model reformulation.

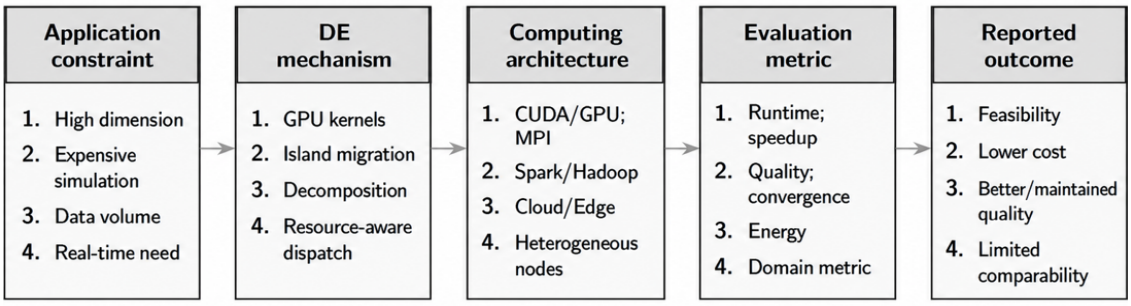

Figure 2. Taxonomy of application-aware scalable DE evaluation

3.3. Engineering and Simulation-Based Optimization

Engineering and simulation-based applications provide the clearest evidence for parallel DE when objective-function evaluation dominates runtime. In power electronic circuit design, resource-aware distributed DE dispatches candidates across heterogeneous

resources to train neural-network controllers [17]. Hyperthermia treatment planning combines DE with CUDA-supported three-dimensional bioheat simulation to make repeated treatment-planning evaluations more feasible [18]. Manufacturing parameter identification similarly benefits when GPU-accelerated simulation makes repeated Johnson-Cook parameter evaluation tractable [30].

Structural and scientific-computing cases further show the role of GPU and island-based strategies. Truss sizing uses CUDA-based cooperative multi-search and island migration to reduce runtime while maintaining acceptable design quality [31], while massively parallel speciation-based DE supports protein structure prediction with reported diversity and speedup advantages [34]. Contextual engineering studies such as machining parameter search and FPRM circuit power optimization demonstrate application value but are treated more cautiously because explicit parallel DE is not their central contribution [32], [33].

3.4. Energy Systems, Power Networks, and Smart Infrastructure

Energy-system studies illustrate why algorithm-level acceleration must be interpreted alongside application-level value. GPU-accelerated DE for hybrid wind-wave layout optimization reports computational speedup together with higher energy output and lower wave loads [19]. Small-population parallel DE for multi-period microgrid dispatch targets time-sensitive scheduling under renewable variability [20]. These studies suggest that speedup matters most when it improves operational feasibility, reliability, or sustainability. Other energy-related studies are more contextual. Knowledge-assisted DE improves non-contact voltage measurement by embedding smart-grid domain knowledge [35], while dimensionless modelling for reverse osmosis reduces the complexity of thermodynamic and economic search [36]. These cases show that scalability may arise not only from hardware acceleration but also from problem reformulation and domain-aware modelling.

3.5. Computing, Big Data, and Distributed Intelligent Systems

Computing and distributed intelligent-system studies treat scalability as an architectural problem. GPU-based cooperative coevolution decomposes large-scale optimization problems and evaluates subcomponents in parallel [11]. Spark-based cooperative

coevolution extends this logic to distributed clusters [13], whereas RST-DE embeds DE into Hadoop and Spark pipelines for feature selection over large datasets [21].

Edge, cloud, fog, and mobile crowdsensing studies show that DE can operate as a component within broader intelligent systems. DE-supported LSTM tuning contributes to auto-scaling and offloading decisions in edge/cloud environments [22], while hybrid Aquila-Differential optimization improves task allocation indicators in mobile crowdsensing [23]. In such hybrid cases, performance improvements should not be attributed to DE alone because fuzzy Q-learning, rough sets, predictive models, or other optimizers may also contribute.

3.6. Strategy Selection and Reporting Recommendations

Neural-network and intelligent-control studies reveal a growing role for DE in model design, controller optimization, pruning, and architecture search. GAN architecture search uses DE with dynamic similarity-aware weight sharing to reduce GPU-hour burden while maintaining competitive generative performance [24]. Individual-level strategy diversity makes the search more communication-free and GPU-oriented [25], while MetaDE uses DE to evolve DE configurations [28].

Model-compression and intelligent-control studies also show the limits of attribution. DE can search pruning parameters for lightweight lithology identification [39], train neural-network controllers in power electronics [17], or support model-assisted machining optimization [32]. However, when DE is embedded in larger learning systems, the observed outcomes result from interaction among DE, the learning architecture, datasets, and implementation choices.

3.7. Discussion

The synthesis supports the central interpretation that application-driven parallel DE should be selected according to problem structure, computational bottleneck, and hardware environment. GPU acceleration is most appropriate when repeated numerical evaluation dominates runtime and computation can be vectorized. MPI migration and island models are suitable when subpopulations can evolve semi-independently. Spark/Hadoop distribution is suitable for data-intensive feature selection or large-scale distributed processing. Cooperative coevolution is appropriate for high-dimensional

decomposition, while resource-aware dispatch is useful when heterogeneous resources differ in computational capacity.

Algorithm-level speedup and application-level value should be distinguished. A study may report a large speedup on a specific GPU, but the practical value depends on whether the faster execution improves treatment planning, energy dispatch, architecture search, feature selection, or real-time feasibility. Conversely, an application may report strong domain outcomes without proving that DE itself was the sole cause of improvement. This is particularly important for hybrid frameworks that include neural networks, fuzzy Q-learning, rough sets, predictive models, or other metaheuristics.

Reported speedups cannot be compared directly across studies unless hardware, software, benchmarks, implementation details, and measurement protocols are comparable. A speedup obtained on a CUDA implementation may not translate to another GPU, CPU, cluster, or cloud environment. Communication overhead also matters: distributed DE can reduce computation time but may introduce migration, synchronization, serialization, and data-transfer costs. Only a minority of studies report communication overhead, code availability, random seeds, or energy consumption, which limits cross-study quantitative comparison.

Sustainability is an underdeveloped dimension in the corpus. Faster execution does not automatically imply lower energy consumption because total energy use depends on device power draw, execution duration, data movement, and cluster utilization, especially when GPUs or distributed clusters are used intensively. Future scalable optimization studies should therefore report runtime and solution quality together with hardware power, energy consumption, GPU hours, and carbon-aware deployment considerations where feasible.

Table 9. Suitability matrix for selecting scalable DE strategies

Problem characteristic	Most suitable scalable DE strategy	Rationale	Reporting caution
Expensive numerical simulation	GPU/CUDA acceleration	High arithmetic intensity and repeated fitness evaluation	Transfer and synchronization costs must be reported.

Problem characteristic	Most suitable scalable DE strategy	Rationale	Reporting caution
High-dimensional continuous optimization	Cooperative coevolution or differential grouping	Reduces search dimension and enables subproblem parallelism	Decomposition quality strongly affects solution quality.
Distributed data or feature selection	Spark/Hadoop distribution	Suitable when data locality and parallel preprocessing matter	Classifier/dataset choices complicate attribution.
Heterogeneous devices/resources	Resource-aware dispatch	Matches individuals/tasks to available computational resources	Needs monitoring and fair baseline comparison.
Time-sensitive dispatch	Small-population or MPI-based DE	Reduces scheduling latency while preserving search diversity	Real-time deployment evidence remains limited.
Neural architecture or model tuning	DE with weight sharing/GPU training	Controls GPU-hour cost and supports exploration	Performance attribution can be mixed with learning components.

As summarized in Table 9, the selection of a scalable Differential Evolution (DE) strategy should be aligned with the computational characteristics of the target problem rather than based solely on the availability of parallel hardware. GPU/CUDA acceleration is particularly suitable for expensive numerical simulations with repeated fitness evaluations, whereas cooperative coevolution or differential grouping is more appropriate for high-dimensional continuous optimization because it enables problem decomposition and subproblem-level parallelism. Spark- or Hadoop-based distribution is better suited to distributed data and feature-selection problems, while resource-aware dispatch is advantageous in heterogeneous computing environments where tasks must be matched to available CPU, GPU, memory, or processing resources. For time-sensitive applications, small-population or MPI-based DE can reduce scheduling latency while maintaining search diversity, whereas neural architecture search and model tuning may

benefit from DE combined with GPU training and weight-sharing mechanisms. However, Table 9 also highlights that each strategy introduces specific reporting concerns, including synchronization and data-transfer costs, decomposition quality, attribution of improvements across hybrid components, workload imbalance, baseline fairness, and limited real-time deployment evidence. Therefore, future scalable DE studies should report not only runtime and solution quality but also hardware configuration, communication overhead, energy consumption, GPU hours, repeated-run protocols, and other resource-efficiency indicators to support reproducible and meaningful cross-study comparisons.

Based on the reporting-quality assessment, future scalable DE studies should report hardware specifications, software versions, parameter settings, random seeds, repeated-run protocols, baseline fairness, communication overhead, data-transfer costs, energy consumption, and code availability. These items are essential for reproducibility because optimization quality and runtime are jointly shaped by the algorithm and the computational environment.

This review has several threats to validity. First, the corpus is Scopus-centered and may miss relevant studies indexed only in IEEE Xplore, Web of Science, ACM Digital Library, ScienceDirect, or institutional repositories. Second, the search string may influence retrieval sensitivity because scalable DE appears under different terms such as island model, many-threaded optimization, cooperative coevolution, heterogeneous computing, or resource-aware dispatch. Third, thematic coding involves interpretive judgment, especially when a study belongs to multiple categories. Fourth, the corpus is heterogeneous across benchmarks, hardware, domains, and performance metrics, limiting quantitative comparison. Fifth, publication bias may favor studies reporting positive speedup or solution-quality improvements. These threats were mitigated by using explicit inclusion criteria, cross-tagging rules, reporting-quality assessment, and cautious interpretation of hybrid-study outcomes.

4. CONCLUSION

This systematic mapping review shows that application-driven parallel Differential Evolution is best understood as an application-aware scalable optimization framework

rather than a universally faster algorithm. The reviewed corpus indicates that scalable DE strategies should be selected according to problem structure, computational bottleneck, dimensionality, evaluation cost, and hardware environment: GPU acceleration is suitable for repeated numerical evaluation, cooperative coevolution supports high-dimensional decomposition, distributed frameworks support data-intensive and cluster-based optimization, and resource-aware dispatch is useful in heterogeneous environments. The review contributes a taxonomy that connects application constraints, DE mechanisms, computational architectures, evaluation metrics, and reported outcomes, but it does not provide a universal ranking of parallelization strategies. Cross-study quantitative comparison remains limited because the corpus is heterogeneous and reporting of hardware, baselines, communication overhead, code, random seeds, and energy consumption is uneven. Future research should therefore prioritize standardized reporting, reproducible implementations, benchmark protocols, and sustainability-aware evaluation for scalable DE.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to the Computer Science Department, Faculty of Mathematics and Natural Sciences, State University of Medan, and the Information System Department, Trilogi University, for their academic support during the preparation of this study. The authors also appreciate the scholarly contributions of researchers in the fields of Differential Evolution, parallel computing, GPU acceleration, distributed optimization, and intelligent systems, whose published works formed the foundation of this systematic mapping review. Special thanks are extended to colleagues and reviewers who provided constructive feedback to improve the clarity, structure, and scientific relevance of this manuscript. The authors hope that this review can contribute to future research on scalable, reproducible, and application-aware optimization frameworks.

APPENDIX A. Protocol Summary and Extraction Codes

Database used: Scopus only. Search date: 24 May 2026. Field syntax: TITLE-ABS-KEY. Main search terms combined "differential evolution" with parallel programming, GPU, CUDA, OpenMP, MPI, multicore, distributed computing, island model, cooperative coevolution,

many-threaded, heterogeneous computing, Spark, Hadoop, cloud, edge, acceleration, optimization, global optimization, computational performance, and scalability. Supplementary Scopus query variants included: (1) "differential evolution" AND engineering AND (parallel OR GPU OR CUDA OR scalability); (2) "differential evolution" AND (smart grid OR microgrid OR energy) AND (dispatch OR optimization OR scalability); (3) "differential evolution" AND (big data OR feature selection OR Spark OR Hadoop OR distributed computing); and (4) "differential evolution" AND (neural networks OR GAN OR intelligent control OR resource allocation) AND (GPU OR parallel OR scalable). Screening codes: period, document type/language, topical relevance, DE centrality, scalability mechanism, performance metric, and application alignment. Extraction codes: year, domain, task, DE mechanism, platform, benchmark/dataset, runtime metric, solution-quality metric, limitation, evidence category, and reporting-quality items.

REFERENCES

[1] R. Storn and K. Price, "Differential Evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Glob. Optim.*, vol. 11, pp. 341–359, 1997, doi: 10.1023/A:1008202821328.

[2] M. F. Ahmad, N. A. M. Isa, W. H. Lim, and K. M. Ang, "Differential evolution: A recent review based on state-of-the-art works," *Alex. Eng. J.*, vol. 61, no. 5, pp. 3831–3872, 2022, doi: 10.1016/j.aej.2021.09.013.

[3] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in Differential Evolution—An updated survey," *Swarm Evol. Comput.*, vol. 27, pp. 1–30, 2016, doi: 10.1016/j.swevo.2016.01.004.

[4] X. Ye, J. Li, P. Wang, and P. N. Suganthan, "A comprehensive survey of adaptive strategies in differential evolutionary algorithms," *Swarm Evol. Comput.*, vol. 99, Art. no. 102081, 2025, doi: 10.1016/j.swevo.2025.102081.

[5] Y. Tan and K. Ding, "A survey on GPU-based implementation of swarm intelligence algorithms," *IEEE Trans. Cybern.*, vol. 46, no. 9, pp. 2028–2041, 2016, doi: 10.1109/TCYB.2015.2479915.

[6] X. Ma et al., "A survey on cooperative co-evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 23, no. 3, pp. 421–441, 2018, doi: 10.1109/TEVC.2018.2868770.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[7] J. Liu, R. Sarker, S. M. Elsayed, D. Essam, and N. Siswanto, "Large-scale evolutionary optimization: A review and comparative study," *Swarm Evol. Comput.*, vol. 85, Art. no. 101466, 2024, doi: 10.1016/j.swevo.2023.101466.

[8] M. J. Page et al., "The PRISMA 2020 statement: An updated guideline for reporting systematic reviews," *BMJ*, vol. 372, Art. no. n71, 2021, doi: 10.1136/bmj.n71.

[9] M. D. J. Peters, C. Godfrey, P. McInerney, Z. Munn, A. C. Tricco, and H. Khalil, "Scoping reviews," in *JBI Manual for Evidence Synthesis*, E. Aromataris, C. Lockwood, K. Porritt, B. Pilla, and Z. Jordan, Eds. JBI, 2024, doi: 10.46658/JBIMES-24-09.

[10] J. P. T. Higgins et al., Eds., *Cochrane Handbook for Systematic Reviews of Interventions*, Version 6.5. Cochrane, 2024.

[11] A. Kelkawi, M. El-Abd, and I. Ahmad, "GPU-based cooperative coevolution for large-scale global optimization," *Neural Comput. Appl.*, vol. 35, no. 6, pp. 4621–4642, 2023, doi: 10.1007/s00521-022-07931-w.

[12] Y.-J. Gong et al., "Distributed evolutionary algorithms and their models: A survey of the state-of-the-art," *Appl. Soft Comput.*, vol. 34, pp. 286–300, 2015, doi: 10.1016/j.asoc.2015.04.061.

[13] M. El-Abd, A. Kelkawi, and I. Ahmad, "Spark-based cooperative coevolution for large scale global optimization," *Cluster Comput.*, vol. 27, no. 2, pp. 1911–1926, 2024, doi: 10.1007/s10586-023-04058-y.

[14] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in Differential Evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, 2006, doi: 10.1109/TEVC.2006.872133.

[15] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential Evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, 2009, doi: 10.1109/TEVC.2008.927706.

[16] J. Zhang and A. C. Sanderson, "JADE: Adaptive Differential Evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, 2009, doi: 10.1109/TEVC.2009.2014613.

[17] X.-F. Liu, Z.-H. Zhan, and J. Zhang, "Resource-aware distributed Differential Evolution for training expensive neural-network-based controller in power electronic circuit," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 11, pp. 6286–6296, 2022, doi: 10.1109/TNNLS.2021.3075205.

[18] G. R. Fatigate, M. Lobosco, and R. F. Reis, "A 3D approach using a control algorithm to minimize the effects on the healthy tissue in the hyperthermia for cancer treatment," *Entropy*, vol. 25, no. 4, Art. no. 684, 2023, doi: 10.3390/e25040684.

[19] Y. Wang, Z. Liu, and H. Wang, "Proposal and layout optimization of a wind-wave hybrid energy system using GPU-accelerated Differential Evolution algorithm," *Energy*, vol. 239, Art. no. 121850, 2022, doi: 10.1016/j.energy.2021.121850.

[20] T. Li, Y. Li, F. Wang, C. Gong, J. Zhang, and H. Ma, "Improved parallel Differential Evolution algorithm with small population for multi-period optimal dispatch problem of microgrids," *Energies*, vol. 18, no. 14, Art. no. 3852, 2025, doi: 10.3390/en18143852.

[21] S. Thakur, R. Dharavath, A. Shankar, P. Singh, M. Diwakar, and M. R. Khosravi, "RST-DE: Rough sets-based new Differential Evolution algorithm for scalable big data feature selection in distributed computing platforms," *Big Data*, vol. 10, no. 4, pp. 356–367, 2022, doi: 10.1089/big.2021.0267.

[22] X. Ma, K. Zong, and A. Rezaeipanah, "Auto-scaling and computation offloading in edge/cloud computing: A fuzzy Q-learning-based approach," *Wirel. Netw.*, vol. 30, no. 2, pp. 637–648, 2024, doi: 10.1007/s11276-023-03486-3.

[23] H. Ghahremani, M. Damrudi, A. Ghaffari, and K. J. Aval, "ADOPT—The Aquila-Differential Optimizer for efficient task allocation and QoS enhancement in mobile crowdsensing," *Cluster Comput.*, vol. 28, Art. no. 798, 2025, doi: 10.1007/s10586-025-05413-x.

[24] A. Rafique, Y. Xue, M. Alhussein, K. Iqbal, M. K. Hasan, and K. Aurangzeb, "Differential evolutionary architecture search with dynamic similarity-aware weight sharing for optimization of GANs," *Neurocomputing*, vol. 671, Art. no. 132619, 2026, doi: 10.1016/j.neucom.2026.132619.

[25] C. Feng, M. Chen, Z. Li, and R. Cheng, "Unleashing the potential of Differential Evolution through individual-level strategy diversity," *IEEE Trans. Evol. Comput.*, early access, 2026, doi: 10.1109/TEVC.2026.3659799.

[26] Z.-B. Pan, L. Yang, Z.-X. Xu, and D.-Y. Wang, "A NEC-based parallel differential evolution algorithm with MKL/CUDA," *J. Netw. Intell.*, vol. 7, no. S1, pp. 114–128, 2022.

[27] A. S. Dufek, J. S. Angelo, D. A. Augusto, and H. J. C. Barbosa, "Accelerating bilevel optimization with hierarchical many-threaded parallel Differential Evolution," *IEEE Trans. Evol. Comput.*, vol. 30, no. 2, pp. 491–503, 2026, doi: 10.1109/TEVC.2025.3560217.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[28] M. Chen, C. Feng, and R. Cheng, "MetaDE: Evolving Differential Evolution by Differential Evolution," *IEEE Trans. Evol. Comput.*, vol. 30, no. 1, pp. 141–155, 2026, doi: 10.1109/TEVC.2025.3541587.

[29] G. Laguna-Sanchez, M. Olguin-Carbajal, J. C. Herrera-Lozada, and O. Cervantes Martinez, "Comparative study of Particle Swarm Optimization and Differential Evolution algorithms on a Graphics Processing Unit," *Comput. Sist.*, vol. 29, no. 1, 2025.

[30] H. Klippel, M. Rothlin, M. Afrasiabi, M. Kuffa, and K. Wegener, "Inverse identification of Johnson-Cook flow stress parameters for Ti6Al4V," *CIRP J. Manuf. Sci. Technol.*, vol. 64, pp. 15–31, 2026, doi: 10.1016/j.cirpj.2025.11.012.

[31] T.-V. Ha, Q.-H. Nguyen, and T.-T. Nguyen, "A parallel differential evolution with cooperative multi-search strategy for sizing truss optimization," *Appl. Soft Comput.*, vol. 131, Art. no. 109762, 2022, doi: 10.1016/j.asoc.2022.109762.

[32] C.-J. Lin, J.-Y. Jhang, and X.-Q. Lin, "Designing an ultrasonic-assisted machining system by using a network mapping fusion-convolutional neuro-fuzzy network model," *IEEE Trans. Ind. Inform.*, vol. 20, no. 7, pp. 9286–9297, 2024, doi: 10.1109/TII.2024.3383866.

[33] Y. Zhou, Z. He, C. Chen, T. Wang, L. Xiao, and X. Wang, "An efficient power optimization approach for fixed polarity Reed-Muller logic circuits based on metaheuristic optimization algorithm," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 41, no. 12, pp. 5380–5393, 2022, doi: 10.1109/TCAD.2022.3149720.

[34] R. S. Parpinelli, M. Boiani, and A. E. P. Dias, "A massively parallel speciation-based Differential Evolution algorithm applied to the 3D-AB protein structure prediction," *Concurr. Comput. Pract. Exp.*, 2021, doi: 10.1002/cpe.6745.

[35] H. Li, C. Ma, Q. Chen, Y. Jiao, and C. He, "Knowledge-assisted Differential Evolution based non-contact voltage measurement for multiconductor systems in smart grid," *Adv. Eng. Inform.*, vol. 62, Art. no. 102740, 2024, doi: 10.1016/j.aei.2024.102740.

[36] M. H. Elbassoussi, O. G. Kaoud, and S. M. Zubair, "On dimensionless modeling of two-stage reverse osmosis: Thermodynamic and economic insights," *Desalination*, vol. 618, Art. no. 119472, 2026, doi: 10.1016/j.desal.2025.119472.

[37] Y. Yang, C. Zhang, and B. Zhang, "A non-revisiting framework for evolutionary multi-task optimization," *Appl. Intell.*, vol. 53, pp. 25931–25953, 2023, doi: 10.1007/s10489-023-04918-5.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[38] V. Osuna-Enciso and E. Guevara-Martinez, "A stigmergy-based Differential Evolution," *Appl. Sci.*, vol. 12, no. 12, Art. no. 6093, 2022, doi: 10.3390/app12126093.

[39] H. Shi, W. Ma, Z.-H. Xu, and P. Lin, "A novel integrated strategy of easy pruning, parameter searching, and re-parameterization for lightweight intelligent lithology identification," *Expert Syst. Appl.*, vol. 231, Art. no. 120657, 2023, doi: 10.1016/j.eswa.2023.120657.