International Journal of Advances in Applied Computational Intelligence

♦ ISSN: 2833-5600 Vol: 08, No. 02, 2026 ♦ PP. 01–10

Article DOI: https://doi.org/10.54216/IJAACI.080201

FULL LENGTH ARTICLE

Metaheuristic Lineage Mining for Electrical and Industrial Engineering Optimization: Detecting Rebranded Algorithms through Equation, Code, and Search-Dynamics Similarity

Amer Ramadan[1,*] **Ana Jurasović**[2]

[1] Faculty of Electrical Engineering, University of Belgrade, Serbia

[2] Faculty of Agriculture Engineering, University of Belgrade, Serbia

Emails: Amer.cce@hotmail.com · ana.jurasovic99@gmail.com

Received: March 24, 2026 Revised: May 20, 2026 Accepted: July 10, 2026 ⋆ Corresponding author

ABSTRACT

Metaheuristic optimization is extensively used in electrical and industrial engineering, where nonlinear, nonconvex, mixed-variable, and simulation-based models arise in power-system operation, energy management, production planning, scheduling, and manufacturing-system design. The rapid proliferation of newly named optimizers, however, makes it difficult to determine whether an apparent contribution introduces a distinct search mechanism or repackages an established algorithmic lineage. This article presents Metaheuristic Lineage Mining (MLM), a multimodal screening framework that integrates executable equation structure, normalized source-code structure, and empirical search dynamics. Identifier-invariant abstract-syntax motifs represent update equations, normalized token n-grams characterize implementation structure, and trajectory fingerprints summarize convergence, diversity, exploration, and improvement behavior. The evaluation includes 57 implementations grouped into 16 documented variant families and 1,710 controlled runs over six continuous benchmark functions and five random seeds. Fusion weights are selected under lineage-held-out validation, preventing algorithms from the evaluated family from influencing model selection. Code similarity is the strongest individual channel, with a mean ROC–AUC of 0.955 and top-1 lineage retrieval of 0.930; equation similarity reaches a ROC–AUC of 0.929. Search-dynamics similarity is weaker alone but supplies behaviorally independent corroboration. Constrained multimodal fusion yields the highest average precision of 0.769, a mean reciprocal rank of 0.935, and clustering normalized mutual information of 0.885. Interpretable cross-family associations, including WhaleFOA–WOA and JADE–SHADE, demonstrate the ability to expose hybrid or ancestral relations that categorical labels omit. The framework is not a detector of misconduct; it is an auditable engineering due-diligence tool for evaluating optimizer originality before costly application studies in power, energy, manufacturing, logistics, and production systems.

Keywords: Metaheuristic algorithms ▪ Electrical engineering optimization ▪ Industrial engineering ▪ Power systems ▪ Production scheduling ▪ Lineage mining ▪ Code similarity ▪ Search dynamics

1. INTRODUCTION

Electrical and industrial engineering contain many optimization problems for which exact or gradient-based methods are difficult to apply directly. Optimal power flow, unit commitment, microgrid energy management, production scheduling, facility configuration, maintenance planning, and simulation-based manufacturing design can involve nonconvex objec-

tives, discrete decisions, stochastic behavior, and expensive evaluations. Recent reviews document the extensive use of metaheuristics across electric-power-system paradigms [1] and production-system optimization [2]. Consequently, the scientific identity and actual operator-level novelty of an optimizer are not merely taxonomic concerns: they affect algorithm selection, benchmarking effort, engineering reproducibility, and the credibility of claimed improvements.

Metaheuristic optimization has developed into a large and diverse research area, but the rate at which newly named algorithms appear has created a methodological problem. New methods are frequently presented through biological, physical, social, or mathematical metaphors even when their computational cores reuse familiar population-update patterns. Recent inventories report hundreds of named nature-inspired optimizers and show that many contemporary proposals perform no better than established, carefully tuned variants [3]. The concern is not that reuse is inherently undesirable. Recombination, adaptation, and hybridization are legitimate sources of progress. The difficulty is that the field lacks a routine, scalable way to separate explicit inheritance from concealed equivalence, superficial re-description, and genuinely different search mechanisms before the method is embedded in an engineering application.

Calls to reduce metaphor-centered novelty claims have become increasingly direct [4, 5]. Critical analyses have shown that several widely cited algorithms can be rewritten as previously known mechanisms or simple mixtures of them [6, 7]. Other studies have responded constructively by proposing taxonomies, component descriptions, and formal notions of homologous or root algorithms [8, 9, 10, 11]. These contributions establish an important principle: algorithm names and inspiration stories are weak evidence of novelty; operators, information flow, state transitions, and behavior should be examined instead.

Editorial and peer-review practice still faces three obstacles. First, mathematical notation can be changed without changing the underlying update rule. Second, source code may share structure despite renamed variables, reordered statements, or rewritten literals. Third, two implementations can look different statically yet generate strongly similar population behavior. A defensible screen should therefore combine symbolic, lexical, and empirical evidence. It should also avoid a circular evaluation in which variants from the family being tested help tune the detector.

This study develops MLM, a reproducible multimodal framework for prioritizing suspiciously similar optimizer pairs. The term *lineage* is used in the software-evolution sense: a documented family relation expressed by an original algorithm and its named variants in a common implementation archive. It does not imply misconduct. The framework computes:

1. an *equation view* from abstract-syntax-tree (AST) motifs that preserve operators and structural roles while suppressing identifiers and numeric literals;
2. a *code view* from normalized token n-grams extracted from each optimizer's population-update method; and
3. a *dynamics view* from standardized trajectories of fitness improvement, population diversity, exploration, and improvement events.

The three similarities are fused under a non-zero evidence constraint and evaluated using lineage-held-out folds. This design asks a demanding question: can a weighting scheme learned from other families recognize a family it has never seen?

The study makes four contributions. First, it provides an operational representation of mathematical-update similarity that is invariant to ordinary renaming. Second, it adapts source-code similarity methods to optimizer evolution routines while excluding names, comments, strings, and file paths. Third, it introduces a behavior fingerprint that compares how algorithms search rather than only how well they finish. Fourth, it evaluates the integrated screen on an archived 2023 corpus of 57 implementations and releases all intermediate matrices, run-level measurements, source hashes, and figures. For electrical and industrial engineering researchers, these contributions provide a pre-application originality audit that can be performed before investing in problem-specific modeling, constraint handling, simulation, and large experimental comparisons.

The intended use is conservative. A high score initiates manual review; it does not establish plagiarism, redundant publication, or deceptive intent. This distinction matters because legitimate hybrids should score highly against their ancestors, and independent implementations can converge on similar code patterns. The practical objective is to give editors, reviewers, and engineering algorithm designers a transparent review queue supported by multiple forms of evidence.

2. RELATED WORK

2.1 Metaheuristics in electrical and industrial engineering

Metaheuristics are widely applied to electrical engineering problems involving power flow, renewable integration, microgrids, smart grids, power quality, and load-related decision models. A systematic review of electric-power-system optimization identified a large and heterogeneous body of work in which established evolutionary and swarm methods coexist with many recently introduced optimizers [1]. In such settings, algorithmic lineage matters because engineering comparisons often attribute gains to a newly named method without isolating whether the improvement comes from a genuinely different search rule, parameter tuning, constraint repair, or implementation details.

Industrial engineering presents the same methodological need. Production scheduling, facility and matrix-system configuration, logistics, maintenance, and simulation optimization commonly use metaheuristics because the decision spaces are combinatorial, stochastic, or computationally expensive. The matrix-production study of [2], for example, illustrates how the selected metaheuristic and its parameterization materially affect solution quality and computation time in a validated manufacturing simulation. Lineage-aware screening can therefore help an engineering study choose transparent baselines, identify inherited operators, and design ablations around the actual modification rather than the algorithm's metaphor or name.

2.2 Novelty, metaphors, and algorithm classification

The growth of named metaheuristics has motivated broad classifications of the field and detailed surveys of major algorithm families, including particle swarm optimization [12, 13]. It has also prompted several complementary lines of criticism. [4] translated recent metaphor-based algorithms into a smaller set of recognizable computational mechanisms. [5] argued that metaphor-driven naming can obscure incremental or absent methodological contributions. Large-scale empirical work by [3] strengthened this concern by cataloguing more than 500 algorithms and comparing recent proposals against stronger variants of established methods. Their results showed that popularity and novelty claims do not ensure competitive search performance.

Taxonomic work offers a more systematic response. [8] proposed a multi-level classification that separates high-level search organization from lower-level components. [9] organized global optimizers through features of their search process rather than inspiration. [10] introduced a component pool for structured comparison, and [11] formalized root and homologous relationships through mathematical operators. These studies support component-aware novelty assessment but generally require expert manual encoding of algorithm descriptions.

Recent empirical-similarity research is especially close to the present work. A behavioral-analysis framework [14] quantified metaheuristic search behavior through a suite of behavioral characteristics and used these signatures to assess algorithmic similarity. Its contribution demonstrates that empirical behavior can expose relationships that names do not. The present study differs by combining empirical fingerprints with equation and source-code evidence, and by evaluating fusion through lineage-held-out folds rather than relying on behavioral evidence alone. Table 1 summarizes the methodological distinction between MLM and the most closely related recent approaches.

2.3 Source-code similarity and reproducible frameworks

Source-code clone detection spans textual, token, tree, graph, and learned representations. A recent systematic review shows that token and AST approaches remain attractive when interpretability and language-specific structural fidelity are required [15]. Optimizer implementations are a distinctive case: much of the scientifically meaningful content is concentrated in one population-update method, while variable names and package boilerplate contribute noise. Normalizing identifiers and literals before computing n-gram similarity therefore offers a transparent baseline that can be inspected and replicated.

Open computational frameworks make such analysis feasible. MEALPY provides a large Python collection of metaheuristics under a shared interface [16]; other frameworks, including pymoo, emphasize standardized experimentation and extensibility [17]. Framework comparisons also warn that implementation choices affect measured behavior [18]. For this reason, the present analysis uses one archived library version and copies the exact source modules into the replication package.

2.4 Behavioral comparison and experimental validity

Benchmarking guidance stresses controlled budgets, repeated trials, transparent parameterization, and appropriate statistical analysis [19, 20, 21]. The use of a maximum generation count can be unfair when algorithms perform different numbers of objective evaluations per generation [22]. Here, a fixed-epoch design is not used to claim superior optimization performance; it is used to sample comparable temporal signatures from the archived implementations. Runtime and terminal errors are retained for auditing, while lineage classification uses normalized trajectories rather than raw final ranks.

Behavioral similarity must also be interpreted carefully. Structural bias can cause unrelated algorithms to prefer similar regions of a search space [23]. Conversely, variants can diverge substantially after a small modification. A dynamics channel is therefore useful as corroboration but should not dominate the static evidence. This principle is enforced in the fusion design: every modality receives at least 0.10 weight, but weights are selected from held-out training families.

3. MATERIALS AND METHODS

3.1 Archived optimizer corpus and reference lineages

The corpus is the MEALPY 3.0.1 software release archived on 5 November 2023 [24]. The release offers a suitable experimental setting because implementations share an interface and coding environment while covering evolutionary, swarm, physics-based, mathematical, human-based, and system-based methods. Sixteen modules containing at least two related implementations were selected. Every class within the predeclared selection was retained, yielding 57 algorithms. The module identity served only as the reference lineage label; module names, class names, file paths, comments, and documentation strings were excluded from predictive features.

Let $\mathscr{A} = \{a_1, \ldots, a_N\}$ denote the optimizer set and let $g_i \in \{1, \ldots, G\}$ be the documented family of algorithm a_i. For each unordered pair (i, j), the reference relation is

$$y_{ij} = \mathbb{I}(g_i = g_j), \qquad 1 \le i < j \le N, \tag{1}$$

where $\mathbb{I}(\cdot)$ is the indicator function. With $N = 57$ and $G = 16$, the corpus contains $\binom{57}{2} = 1{,}596$ pairs: 86 same-lineage pairs and 1,510 cross-family pairs. The resulting positive prevalence, $86/1{,}596 = 5.39\%$, reflects the imbalance expected in editorial screening, where most comparisons should be unrelated. The family composition is reported in table 2.

3.2 Equation-motif representation

Published equations are not always available in machine-readable form, and notation differs across papers. The implementation's `evolve()` method was therefore used as an executable proxy for the population-update equations. Python ASTs were parsed after indentation normalization. For every assignment, conditional, loop, and return statement, a structural motif was emitted. The representation preserves operator type and broad semantic roles while replacing local names and constants.

For example, a binary expression is represented recursively

Table 1. Positioning of the proposed framework relative to recent work.

Study	Primary evidence	Unit of comparison	Automated cross-family screen	Distinction from MLM
[4]	Mathematical reinterpretation	Named algorithms	No	Expert explanatory analysis without code or trajectory fusion
[10]	Component templates	Algorithm components	Partly	Structured manual comparison rather than source-derived signatures
[9]	Taxonomic features	Global optimizers	No	Classification framework, not pairwise screening
[11]	Mathematical paradigms	Root/homologous algorithms	Partly	Formal discriminant without implementation-level and behavioral evidence
[15]	Code representations	General software clones	Yes	Broad software setting; no optimizer equations or search behavior
[14]	Behavioral characteristics	Metaheuristic search behavior	Yes	Behavior-centered similarity without AST and code-token fusion
This study	Equation motifs, code n-grams, search trajectories	Optimizer implementations	Yes	Multimodal, lineage-held-out fusion with reproducible review queue

Table 2. Reference lineage composition.

Family	n	Family	n
AEO	5	ARO	3
BA	3	DE	4
EO	3	EP	2
ES	4	FOA	3
GWO	4	PSO	7
QSA	5	SCA	3
SHADE	2	TLO	3
TWO	4	WOA	2
		Total	**57**

as

$$M(x \circ y) = \mathrm{OP}_{\circ}\left[M(x), M(y)\right], \tag{2}$$

where identifiers map to `VAR`, numeric literals to `CONST`, population-index expressions to `VECTOR`, and attributes containing best/worst semantics to `BEST`/`WORST`. Random-number calls remain identifiable as `RNG`. Assignment values receive an `UPDATE=` prefix, conditionals receive `GUARD=`, and loops receive `LOOP=`.

The resulting motif sequence was vectorized with sublinear term-frequency inverse-document-frequency (TF–IDF) weighting over 1–3-grams. If n_{ik} is the count of motif n-gram k in algorithm i, and df_k is the number of algorithms containing it, the unnormalized feature is

$$\tilde{e}_{ik} = \left[1 + \log(n_{ik})\right] \log\left(\frac{N+1}{\mathrm{df}_k + 1}\right), \tag{3}$$

with $\tilde{e}_{ik} = 0$ when $n_{ik} = 0$. Let $\boldsymbol{e}_i = \tilde{\boldsymbol{e}}_i / \|\tilde{\boldsymbol{e}}_i\|_2$. Equation similarity between algorithms i and j is then

$$S_{\mathrm{eq}}(i,j) = \boldsymbol{e}_i^{\top} \boldsymbol{e}_j. \tag{4}$$

Values were clipped to $[0,1]$. This channel is invariant to ordinary renaming but intentionally sensitive to update composition, guard logic, and control structure.

3.3 Normalized code representation

The code view captures lexical and local structural resemblance that may be lost by the coarser equation motifs. Tokenization removed whitespace, indentation tokens, comments, strings, and documentation. Python keywords and operators were preserved. Numeric literals became `NUM`; ordinary identifiers became `ID`. A limited whitelist retained computationally meaningful calls such as random sampling, trigonometric functions, clipping, population evaluation, and greedy selection.

Normalized token streams were represented by TF–IDF 2–5-grams using the weighting in eq. (3). After ℓ_2 normalization of each code vector $\boldsymbol{c}_i$, the code similarity is

$$S_{\mathrm{code}}(i,j) = \boldsymbol{c}_i^{\top} \boldsymbol{c}_j. \tag{5}$$

Longer n-grams distinguish statement patterns while shorter n-grams tolerate limited rewriting. No source filename, family label, or class name enters $\boldsymbol{c}_i$.

3.4 Search-dynamics fingerprints

Static similarity can identify implementation inheritance but cannot show whether two algorithms behave similarly. Each optimizer was therefore executed on six standard minimization functions: Sphere, Rastrigin, Ackley, Griewank, Zakharov, and Levy. The functions cover separable and non-separable, unimodal and multimodal landscapes. Their definitions and bounds are summarized in table 3. These functions provide a domain-neutral pre-screen of search dynamics. They do not replace application-specific evaluation on constrained optimal power flow, energy scheduling, production planning, facility layout, or simulation-based industrial models; instead, they establish whether two optimizers exhibit similar generic behavior before those more expensive engineering experiments are undertaken.

For every algorithm–function combination, five seeded runs were executed with dimension $d = 10$, population size 30, and 60 epochs. Seeds were 11, 23, 37, 53, and 71. This produced $57 \times 6 \times 5 = 1{,}710$ runs. All runs used the archived default parameters except `HI_WOA` and `ImprovedTLO`, for which re-

Table 3. Continuous functions used to generate search-dynamics signatures. The global optimum value is zero.

Function	Definition	Lower	Upper	Property
Sphere	$f(\boldsymbol{x}) = \sum_{k=1}^{d} x_k^2$	-5.12	5.12	Smooth, separable, unimodal
Rastrigin	$f(\boldsymbol{x}) = 10d + \sum_{k=1}^{d}[x_k^2 - 10\cos(2\pi x_k)]$	-5.12	5.12	Highly multimodal, separable
Ackley	$f(\boldsymbol{x}) = -20e^{-0.2\sqrt{d^{-1}\sum x_k^2}} - e^{d^{-1}\sum\cos(2\pi x_k)} + 20 + e$	-32.768	32.768	Multimodal, nearly flat outer region
Griewank	$f(\boldsymbol{x}) = \sum x_k^2/4000 - \prod\cos(x_k/\sqrt{k}) + 1$	-600	600	Non-separable, many regular local minima
Zakharov	$f(\boldsymbol{x}) = \sum x_k^2 + (\sum 0.5kx_k)^2 + (\sum 0.5kx_k)^4$	-5	10	Non-separable, unimodal
Levy	Standard Levy function with $w_k = 1 + (x_k - 1)/4$	-10	10	Multimodal, non-separable

quired constructor arguments were fixed to documented valid values. No run failed.

Four trajectories were extracted at each epoch: global-best fitness b_t, population diversity v_t, exploration percentage q_t, and positive log-improvement r_t. Fitness was transformed to $\ell_t = \log_{10}(\max(|b_t|, 10^{-12}))$. A direction-normalized convergence path was computed as

$$\gamma_t = \mathrm{clip}\left(\frac{\ell_t - \ell_T}{\ell_1 - \ell_T}, 0, 1\right), \tag{6}$$

with a constant vector when the denominator was negligible. Diversity was divided by its within-run maximum, exploration by 100, and improvement was

$$r_t = \frac{\max(0, -(\ell_t - \ell_{t-1}))}{\max_s \max(0, -(\ell_s - \ell_{s-1}))}. \tag{7}$$

Each trajectory was linearly resampled to $K = 20$ checkpoints. For function h and seed s, the behavioral vector is

$$\boldsymbol{z}_{ihs} = \left[\boldsymbol{\gamma}_{ihs}^\top, \boldsymbol{v}_{ihs}^\top, \boldsymbol{q}_{ihs}^\top, \boldsymbol{r}_{ihs}^\top\right]^\top \in \mathbb{R}^{4K}. \tag{8}$$

The componentwise median across the five seeds formed an 80-element function signature, and concatenation across the six functions produced $\boldsymbol{z}_i \in \mathbb{R}^{480}$. Each feature was standardized across algorithms as $d_{im} = (z_{im} - \mu_m)/\sigma_m$. Because standardized vectors may have negative cosine values, dynamics similarity was mapped to $[0, 1]$:

$$S_{\mathrm{dyn}}(i, j) = \frac{1}{2}\left(1 + \frac{\boldsymbol{d}_i^\top \boldsymbol{d}_j}{\|\boldsymbol{d}_i\|_2 \|\boldsymbol{d}_j\|_2}\right). \tag{9}$$

3.5 Constrained multimodal fusion

The fused score is a convex combination:

$$S_{\mathrm{fuse}}(i, j) = w_e S_{\mathrm{eq}}(i, j) + w_c S_{\mathrm{code}}(i, j) + w_d S_{\mathrm{dyn}}(i, j), \tag{10}$$

subject to $w_e + w_c + w_d = 1$ and $w_e, w_c, w_d \geq \varepsilon$, where $\varepsilon = 0.10$. Candidate weights were enumerated in increments of 0.05. The lower bound prevents an apparently strong static channel from eliminating behavioral corroboration.

For each held-out lineage g, training pairs excluded every pair containing an algorithm from g. Using eq. (10), the fold-specific weights were selected as

$$\boldsymbol{w}_g^* = \arg\max_{\boldsymbol{w} \in \Delta_\varepsilon} \frac{\mathrm{AUC}_g^{\mathrm{tr}}(\boldsymbol{w}) + \mathrm{AP}_g^{\mathrm{tr}}(\boldsymbol{w})}{2}, \tag{11}$$

where $\Delta_\varepsilon = \{\boldsymbol{w} \in \mathbb{R}^3 : \mathbf{1}^\top \boldsymbol{w} = 1,\ w_m \geq \varepsilon\}$ is the constrained simplex. Evaluation then included all pairs containing at least one member of g. This process was repeated for all 16 lineages. A final weight vector was fitted on all pairs only after cross-validation and was used for the released similarity matrix, retrieval analysis, clustering, and cross-family review queue.

3.6 Evaluation criteria

ROC–AUC measures ranking over all positive–negative pair comparisons, while AP is sensitive to the low positive prevalence and therefore more representative of screening quality. Retrieval was evaluated by querying every algorithm against the other 56 implementations. If ρ_i is the rank of the highest-scoring algorithm satisfying $g_j = g_i$ and $j \neq i$, then

$$\mathrm{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\rho_i}, \tag{12}$$

and top-1 accuracy is $N^{-1}\sum_i \mathbb{I}(\rho_i = 1)$. Agglomerative clustering with average linkage was applied to distance $D_{ij} = 1 - S_{\mathrm{fuse}}(i, j)$, specifying 16 clusters. Agreement with reference families was measured by adjusted Rand index (ARI) and normalized mutual information (NMI).

Cross-family pairs were sorted by S_{fuse}. These pairs were not treated as errors automatically: hybrids and algorithms with explicit ancestry are expected to cross module boundaries. They form a manual-review queue whose interpretation requires paper equations, chronology, citations, and provenance. The complete construction and evaluation sequence is summarized in fig. 1.

4. RESULTS

4.1 Held-out lineage discrimination

Table 4 reports the mean and standard deviation across the 16 held-out lineages. Code-token similarity was the strongest single view by ROC–AUC (0.955), followed by equation motifs (0.929). Their AP values were nearly equal because failures were concentrated in different small families. Dynamics alone was markedly weaker, with 0.723 ROC–AUC and 0.272 AP, confirming that behavioral resemblance is insufficient as a sole lineage criterion.

The fused model obtained the highest AP, 0.769, improving on equation and code views while retaining high ROC–AUC (0.949). The modest AUC reduction relative to code alone reflects the imposed minimum weight on the weaker dynamics channel. This is an intentional trade-off: the screen prioritizes

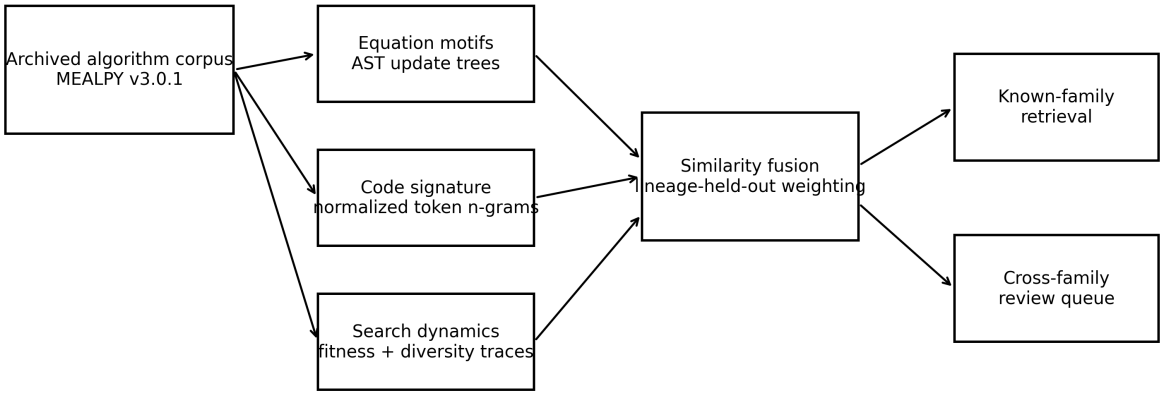

Figure 1. Workflow of the proposed lineage-mining framework. Family names are used only for evaluation, never as input features.

Table 4. Lineage-held-out pair discrimination (mean ± standard deviation across 16 families).

Method	ROC–AUC	AP
Equation	0.929 ± 0.130	0.744 ± 0.363
Code	$\mathbf{0.955} \pm 0.076$	0.743 ± 0.315
Dynamics	0.723 ± 0.242	0.272 ± 0.314
Fused	0.949 ± 0.087	$\mathbf{0.769} \pm 0.320$

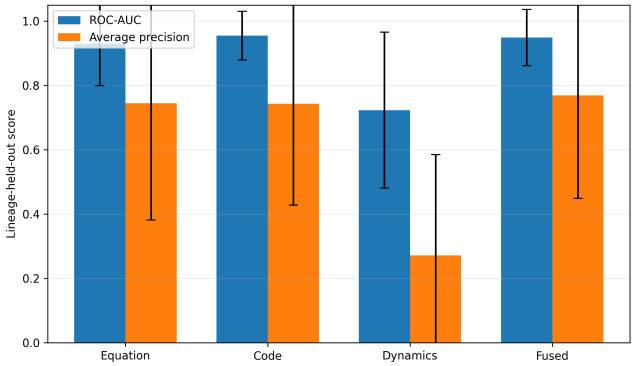

Figure 2. Held-out discrimination by evidence channel. Error bars show standard deviation across reference lineages.

multimodal support over maximizing a single aggregate statistic. The final all-pair weights were $w_e = 0.20$, $w_c = 0.70$, and $w_d = 0.10$. Figure 2 visualizes the corresponding between-lineage variation for both discrimination measures.

Variation across families was substantial. Families with closely related implementations, including AEO, ARO, EO, PSO, QSA, TWO, and WOA, were generally recovered reliably. DE was harder: its held-out fused ROC–AUC was 0.767 because adaptive DE mechanisms also occur in the separately implemented SHADE family. That apparent confusion is scientifically meaningful, not merely a model defect. It illustrates why a lineage screen should allow cross-family ancestry rather than force mutually exclusive labels.

Table 5. Nearest-lineage retrieval over 57 algorithm queries.

Method	Top-1	MRR
Equation	0.842	0.881
Code	**0.930**	**0.950**
Dynamics	0.474	0.556
Fused	0.912	0.935

4.2 Nearest-lineage retrieval

The retrieval results in table 5 parallel the pairwise evaluation. Code similarity placed a same-lineage algorithm first for 53 of 57 queries (0.930) and achieved an MRR of 0.950. Fusion retrieved the correct family first for 52 queries (0.912) with MRR 0.935. Equation motifs recovered 48 top-1 matches. Dynamics alone retrieved 27.

Family-level fused retrieval was perfect for 12 of the 16 lineages. The exceptions were ES (0.50), FOA (0.667), SCA (0.667), and DE (0.75). ES contains both classical and covariance-matrix variants, which differ structurally. FOA includes WhaleFOA, an explicit whale-assisted hybrid whose nearest representation can reasonably fall outside the FOA module. These cases demonstrate that module membership is a practical but imperfect ground truth.

4.3 Lineage-map structure

Agglomerative clustering of the fused matrix produced ARI 0.621 and NMI 0.885 against the 16 module families. The high NMI indicates that most lineage information is preserved even when a few variants merge with related external mechanisms or split within heterogeneous modules. The full matrix in fig. 3 shows strong block structure for most families. The dendrogram in fig. 4 provides a more useful editorial view because it exposes nested proximity rather than requiring a hard family decision.

4.4 Cross-family review candidates

Table 6 lists the ten highest cross-family scores. The strongest pair was WhaleFOA–OriginalWOA at 0.760, driven by code similarity of 0.865 and supported by equation and dynamics

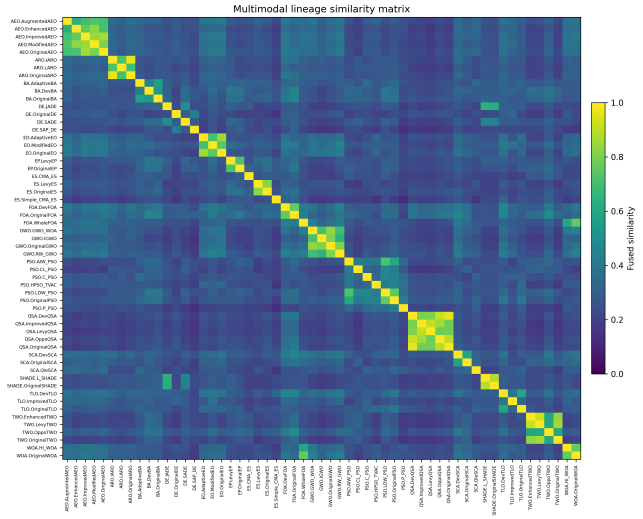

Figure 3. Fused similarity matrix sorted by documented family. Bright blocks indicate high within-family resemblance; off-diagonal bright regions identify cross-family candidates for review.

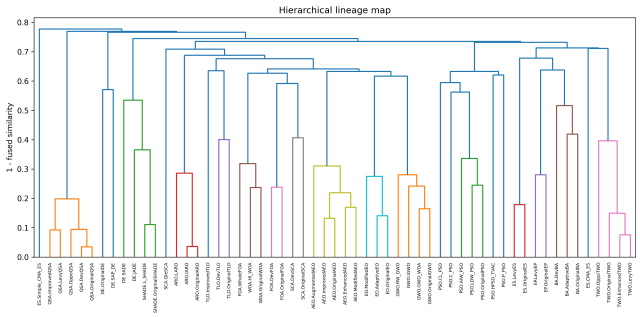

Figure 4. Hierarchical lineage map obtained from average-linkage clustering of $1 - S_{\text{fuse}}$.

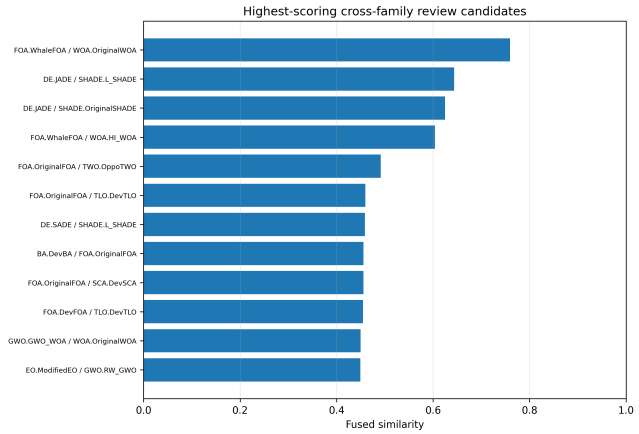

Figure 5. Highest cross-family review scores. The figure is intended for triage; component scores and provenance must be checked before reaching a conclusion.

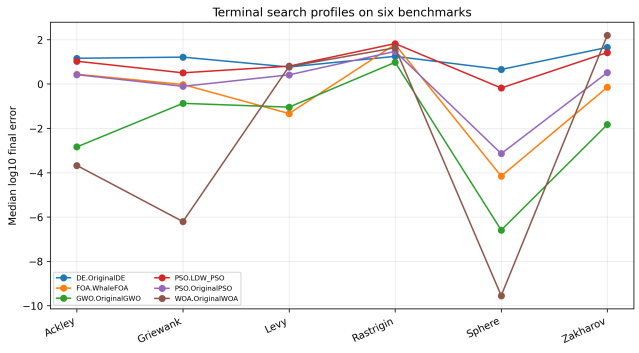

Figure 6. Median terminal log-error profiles for representative implementations. These values illustrate behavioral heterogeneity; lineage dynamics were computed from complete trajectories rather than terminal errors alone.

evidence. This is a face-valid result: the class name and implementation represent an explicit whale-enhanced FOA variant. The screen has therefore recovered a known hybrid component despite the pair receiving different module labels.

The next two candidates connected JADE with L-SHADE and OriginalSHADE. JADE and SHADE are adaptive differential-evolution methods, so their strong dynamic and code resemblance is consistent with shared DE ancestry. These examples show why cross-family candidates should not be counted as false positives without interpretation. The system is detecting operator lineage that a folder-based ground truth does not fully encode.

Several lower-ranked candidates were driven primarily by dynamics, such as OriginalFOA–OppoTWO. These require greater skepticism because common convergence patterns can arise independently. A practical review policy can therefore inspect the three component scores alongside the fusion score rather than applying one universal threshold. The ranking pattern is shown graphically in fig. 5, while table 6 retains the component-level evidence required for interpretation.

4.5 Behavioral evidence and computational audit

The representative terminal profiles in fig. 6 demonstrate why dynamics can corroborate but cannot independently establish lineage. Related PSO variants often show similar relative strengths across functions, yet unrelated methods can also end with comparable error patterns. The actual dynamic representation is richer than terminal performance: it contains 480 standardized trajectory features, including diversity and improvement timing. Even so, its held-out variance is high because stochastic behavior is affected by parameter defaults and implementation details.

All 1,710 planned runs completed. The archived optimizer code emitted one numerical overflow warning in a Levy-flight division routine; MEALPY's correction pathway contained the value and the run completed. No result was removed or replaced. Run-level runtime, initial and final fitness, area under the log-error curve, and final diversity are preserved in `data/run_summary.csv`. The warning is recorded in the execution log to avoid presenting numerical reproducibility as frictionless.

5. DISCUSSION

5.1 What the results establish

The results support three conclusions. First, source normalization is highly effective for recovering documented variant families. The code channel's 0.930 top-1 retrieval shows that meaningful implementation structure survives removal of identifiers, strings, comments, and literal values. This matters because ordinary renaming is insufficient to conceal operator-level resemblance from a token-structural representation.

Table 6. Highest-scoring cross-family pairs. Scores indicate review priority, not misconduct or redundant publication.

Rank	Algorithm A	Algorithm B	S_{eq}	S_{code}	S_{dyn}	S_{fuse}
1	FOA.WhaleFOA	WOA.OriginalWOA	0.537	0.865	0.469	0.760
2	DE.JADE	SHADE.L_SHADE	0.446	0.670	0.855	0.643
3	DE.JADE	SHADE.OriginalSHADE	0.394	0.668	0.787	0.625
4	FOA.WhaleFOA	WOA.HI_WOA	0.316	0.695	0.538	0.604
5	FOA.OriginalFOA	TWO.OppoTWO	0.068	0.553	0.912	0.492
6	FOA.OriginalFOA	TLO.DevTLO	0.147	0.580	0.242	0.460
7	DE.SADE	SHADE.L_SHADE	0.178	0.489	0.812	0.459
8	BA.DevBA	FOA.OriginalFOA	0.072	0.500	0.916	0.456
9	FOA.OriginalFOA	SCA.DevSCA	0.112	0.530	0.620	0.456
10	FOA.DevFOA	TLO.DevTLO	0.128	0.531	0.577	0.455

Second, equation motifs provide a complementary and more semantically compressed view. Their AP slightly exceeded code AP before fusion despite lower top-1 retrieval. Motifs emphasize assignments, guards, loops, random draws, and best/worst references, making them less sensitive to formatting and helper-code differences. A reviewer can also inspect motif sequences more readily than thousands of raw token n-grams.

Third, search dynamics should be treated as corroboration rather than proof. Dynamics recovered some relationships missed by static channels, particularly among adaptive DE methods, but it also produced high similarity between implementations with no documented family relation. Shared landscapes, population sizes, and convergence pressures can lead to behavioral convergence. The minimum dynamics weight of 0.10 is therefore sufficient to retain independent evidence without allowing it to dominate the screen.

The fusion's highest AP is practically important. In a low-prevalence review queue, average precision better reflects the burden placed on an editor than ROC–AUC alone. A score that ranks a small number of plausible relationships near the top is more useful than one that separates most easy negatives but floods the top ranks with unrelated pairs.

5.2 Implications for electrical and industrial engineering

The proposed screen is particularly relevant when a newly named optimizer is introduced for an engineering application. In electrical engineering, the method can be compared against archived lineages before it is used for optimal power flow, reactive-power dispatch, microgrid scheduling, controller tuning, network reconfiguration, or renewable-energy planning. A high static similarity to an established family would indicate that the engineering study should compare against that ancestor directly and explain the exact constraint-handling or search modification responsible for any improvement.

In industrial engineering, the same procedure can precede experiments on production scheduling, maintenance planning, facility layout, supply-chain coordination, or simulation-based system configuration. These studies often require substantial model development and many stochastic evaluations. Detecting lineage early reduces redundant benchmarking and supports more informative ablation studies. It also separates three sources of performance that are frequently conflated: the inherited optimizer core, the engineering encoding and repair strategy, and the genuinely new operator.

The present results should not be interpreted as evidence that one lineage is preferable for a particular electrical or industrial problem. Engineering suitability still depends on feasibility preservation, mixed-variable handling, computational budget, robustness to uncertainty, scalability, and sensitivity to operating constraints. MLM addresses the prior question of algorithmic identity and relation; domain-specific optimization performance remains a separate empirical question.

5.3 Editorial and research workflow

A responsible workflow would use MLM in four stages. First, a submitted algorithm is converted into the same normalized equation and code signatures. Second, its implementation is run under a small standardized trajectory suite. Third, the fused nearest neighbors and component scores are produced. Fourth, a human reviewer examines the original papers, equations, chronology, citations, and declared modifications. For an engineering paper, this audit should occur before the optimizer is coupled to the electrical network model, production simulator, scheduling formulation, or other domain-specific evaluation environment.

The outcome should be one of several nuanced categories: independently novel mechanism; explicit variant with adequate attribution; legitimate hybrid; implementation reuse with clear licensing; substantial equivalence requiring stronger differentiation; or unresolved similarity requiring author clarification. Binary labels such as "novel" and "copied" are usually too crude. The framework is designed to make the evidence inspectable, not to automate an accusation.

Algorithm designers can use the same process before submission. A proposed optimizer that scores highly against an existing method can be rewritten transparently as a variant, with ablation studies identifying the actual added component. This may improve the quality of contributions by shifting emphasis from a new metaphor to measurable operator changes.

5.4 Interpretation of cross-family relations

The strongest cross-family results demonstrate both the utility and the limits of folder-derived ground truth. WhaleFOA belongs to the FOA module but intentionally imports a WOA mechanism. JADE and SHADE reside in separate files yet share differential-evolution ancestry. A strict classifier would penalize these links; a lineage miner should expose them.

This distinction suggests that future benchmarks should use a

graph rather than one categorical label per algorithm. Edges could encode direct variants, hybrids, borrowed operators, common root mechanisms, and independent implementations. The present corpus provides categorical families because those labels are reproducible from the archive, but the cross-family queue already approximates missing graph edges.

5.5 Threats to validity

Construct validity. The equation channel extracts executable AST motifs, not equations directly from article manuscripts. It therefore represents implemented mathematics and may omit conceptual distinctions described outside `evolve()`. Conversely, implementation shortcuts can make two theoretically different methods look similar. Manual paper review remains necessary.

Reference-lineage validity. Module families are convenient and documented, but they are not a complete genealogy. Hybrids cross module boundaries, and heterogeneous modules can contain variants with major structural differences. The DE–SHADE and WhaleFOA–WOA findings illustrate this limitation.

External validity. All implementations come from one Python library. Shared coding conventions may increase within-family and cross-family code similarity. Evaluation on additional archives and independent reimplementations is needed before setting universal thresholds. The present corpus also does not contain full electrical-network or production-system models; transfer to engineering practice should therefore be assessed with constrained, mixed-variable, simulation-based, and uncertainty-aware test cases.

Experimental validity. The trajectory protocol uses six classical functions, dimension 10, a population of 30, and 60 epochs. It was selected to characterize behavior within a feasible, fully repeated design, not to rank optimizers. Fixed epochs do not equalize objective evaluations across algorithms [22]. Future work should include evaluation-budget signatures and more varied dimensions.

Parameter sensitivity. Default parameters were retained to preserve archived implementations, but defaults can alter dynamics. Static channels are unaffected, whereas the dynamic channel may change under tuning. Multi-configuration fingerprints would be more robust but substantially more expensive.

Statistical uncertainty. Five seeds stabilize median trajectories but are not intended for fine-grained performance claims. Family-level folds contain different numbers of positive pairs, causing high AP variance for two- and three-member families. The released predictions allow alternative resampling and confidence-interval analyses.

Ethical validity. Similarity is not evidence of author intent. Common operators, lawful reuse, parallel discovery, and explicit hybridization can all produce high scores. Any public claim about duplication should be based on source chronology, licenses, citations, and expert mathematical comparison, not the fused score alone.

6. CONCLUSION

This article presented a reproducible approach to metaheuristic lineage screening for optimization research in electrical and industrial engineering. The framework triangulates executable equation structure, normalized code, and search behavior before a candidate optimizer is embedded in a domain-specific application. On 57 archived implementations across 16 families, static representations recovered documented lineages with high held-out discrimination. Code similarity achieved 0.955 ROC–AUC and 0.930 top-1 retrieval, while equation motifs achieved 0.929 ROC–AUC. A constrained multimodal fusion delivered the strongest average precision and exposed interpretable cross-family relations that categorical module labels miss.

The main value of the framework is procedural rather than punitive. It transforms a vague novelty concern into an auditable sequence of pairwise evidence, held-out validation, and manual review. For power-system, energy-management, manufacturing, scheduling, and simulation-optimization studies, this process can clarify whether reported gains originate from a new search mechanism, an inherited optimizer, or the engineering formulation and constraint-handling strategy. The released package makes every score traceable to an archived source file, a normalized signature, or one of 1,710 recorded optimization runs. Future work should extend the corpus across languages and libraries, extract equations directly from manuscripts, model ancestry as a graph, and validate transfer on constrained electrical-engineering and industrial-engineering benchmark suites under multiple parameter configurations and evaluation budgets.

REFERENCES

[1] G. H. Valencia-Rivera, M. T. Benavides-Robles, A. Vela Morales, I. Amaya, J. M. Cruz-Duarte, J. C. Ortiz-Bayliss, and J. G. Avina-Cervantes, "A systematic review of metaheuristic algorithms in electric power systems optimization," *Applied Soft Computing*, vol. 150, p. 111047, 2024.

[2] M. Benfer, V. Heyer, O. Brützel, C. Liebrecht, S. Peukert, and G. Lanza, "Analysis of metaheuristic optimisation techniques for simulated matrix production systems," *Production Engineering*, vol. 18, pp. 159–168, 2024.

[3] Z. Ma, G. Wu, P. N. Suganthan, A. Song, and Q. Luo, "Performance assessment and exhaustive listing of 500+ nature-inspired metaheuristic algorithms," *Swarm and Evolutionary Computation*, vol. 77, p. 101248, 2023.

[4] M. A. Lones, "Mitigating metaphors: A comprehensible guide to recent nature-inspired algorithms," *SN Computer Science*, vol. 1, p. 49, 2020.

[5] C. Aranha, C. L. Camacho Villalón, F. Campelo, M. Dorigo, R. Ruiz, M. Sevaux, K. Sörensen, and T. Stützle, "Metaphor-based metaheuristics, a call for action: The elephant in the room," *Swarm Intelligence*, vol. 16, no. 1, pp. 1–6, 2022.

[6] C. L. Camacho-Villalón, M. Dorigo, and T. Stützle, "Exposing the grey wolf, moth-flame, whale, firefly, bat, and antlion algorithms: Six misleading optimization techniques inspired by bestial metaphors," *International Transactions in Operational Research*, vol. 30, no. 6, pp. 2945–2971, 2023.

[7] L. Velasco, H. Guerrero, and A. Hospitaler, "A literature review and critical analysis of metaheuristics recently developed," *Archives of Computational Methods in Engineering*, vol. 31, pp. 125–146, 2024.

[8] H. Stegherr, M. Heider, and J. Hähner, "Classifying metaheuristics: Towards a unified multi-level classification system," *Natural Computing*, vol. 21, pp. 155–171, 2022.

[9] J. Stork, A. E. Eiben, and T. Bartz-Beielstein, "A new taxonomy of global optimization algorithms," *Natural Computing*, vol. 21, no. 2, pp. 219–242, 2022.

[10] J. de Armas, E. Lalla-Ruiz, S. L. Tilahun, and S. Voß, "Similarity in metaheuristics: A gentle step towards a comparison methodology," *Natural Computing*, vol. 21, pp. 265–287, 2022.

[11] Z. Hu, Q. Zhang, Y. Wang, Q. Su, and Z. Xiong, "Research orientation and novelty discriminant for new metaheuristic algorithms," *Applied Soft Computing*, vol. 157, p. 111521, 2024.

[12] A. E. Ezugwu, A. K. Shukla, R. Nath, A. A. Akinyelu, J. O. Agushaka, H. Chiroma, and P. K. Muhuri, "Metaheuristics: A comprehensive overview and classification along with bibliometric analysis," *Artificial Intelligence Review*, vol. 54, pp. 4237–4316, 2021.

[13] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, and S. Mirjalili, "Particle swarm optimization: A comprehensive survey," *IEEE Access*, vol. 10, pp. 10 031–10 061, 2022.

[14] L. Hayward and A. P. Engelbrecht, "Determining metaheuristic similarity using behavioral analysis," *IEEE Transactions on Evolutionary Computation*, vol. 29, no. 1, pp. 262–274, 2025.

[15] M. Zakeri-Nasrabadi, S. Parsa, M. Ramezani, C. K. Roy, and M. Ekhtiarzadeh, "A systematic literature review on source code similarity measurement and clone detection: Techniques, applications, and challenges," *Journal of Systems and Software*, vol. 204, p. 111796, 2023.

[16] N. Van Thieu and S. Mirjalili, "MEALPY: An open-source library for latest meta-heuristic algorithms in python," *Journal of Systems Architecture*, vol. 139, p. 102871, 2023.

[17] J. Blank and K. Deb, "pymoo: Multi-objective optimization in python," *IEEE Access*, vol. 8, pp. 89 497–89 509, 2020.

[18] A. Ramírez, R. Barbudo, and J. R. Romero, "An experimental comparison of metaheuristic frameworks for multi-objective optimization," *Expert Systems*, vol. 40, no. 4, p. e12672, 2023.

[19] E. Osaba, E. Villar-Rodriguez, J. Del Ser, A. J. Nebro, D. Molina, A. LaTorre, P. N. Suganthan, C. A. Coello Coello, and F. Herrera, "A tutorial on the design, experimentation and application of metaheuristic algorithms to real-world optimization problems," *Swarm and Evolutionary Computation*, vol. 64, p. 100888, 2021.

[20] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff, "COCO: A platform for comparing continuous optimizers in a black-box setting," *Optimization Methods and Software*, vol. 36, no. 1, pp. 114–144, 2021.

[21] F. Campelo and E. F. Wanner, "Sample size calculations for the experimental comparison of multiple algorithms on multiple problem instances," *Journal of Heuristics*, vol. 26, pp. 851–883, 2020.

[22] M. Ravber, S.-H. Liu, M. Mernik, and M. Črepinšek, "Maximum number of generations as a stopping criterion considered harmful," *Applied Soft Computing*, vol. 128, p. 109478, 2022.

[23] K. Rajwar and K. Deep, "Structural bias in metaheuristic algorithms: Insights, open problems, and future prospects," *Swarm and Evolutionary Computation*, vol. 92, p. 101812, 2025.

[24] N. Van Thieu, "MEALPY version 3.0.1," Zenodo software archive, 2023, version 3.0.1.