

Paper Type: Original Article

A Taxonomy and Evidence Framework for Hybrid Metaheuristics in Smart Manufacturing and Supply Chains

Maedeh Fasihi¹ , Azadeh Shemshad^{2,*} 

¹ Department of Industrial Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran; maedeh.fasihi@gmail.com.

² Department of Accounting and Control, Erasmus University, 3062 PA Rotterdam, Netherlands; azadeh.shemshad@gmail.com.

Citation:

Received: 06 April 2025

Revised: 15 June 2025

Accepted: 08 August 2025

Fasihi, M., & Shemshad, A. (2026). A taxonomy and evidence framework for hybrid metaheuristics in smart manufacturing and supply chains. *Computational Engineering and Technology Innovations*, 3(1), 41-51.

Abstract

Hybrid metaheuristics are often introduced by naming two algorithms, although an algorithm pair does not reveal what decisions are encoded, which component controls feasibility, when information is exchanged, or what evidence would justify deployment. This article develops a decision-contract perspective for hybrid optimization in smart manufacturing and supply chains. A hybrid is specified through five linked contracts: The decision model, component roles, information exchange, resource allocation, and evidential claim. This representation refines established high-/low-level and relay/teamwork taxonomies by making the operational interface auditable. The framework is applied to flexible job-shop scheduling, vehicle routing, supply-chain network design, maintenance, and energy-aware operations. A literature synthesis shows that recent work has expanded from handcrafted hybrids toward matheuristics, Adaptive Large-Neighborhood Search (ALNS), automated configuration, surrogate assistance, and reinforcement-learning-guided search, but comparison remains fragile when feasibility rules, tuning cost, latency, or certificate status differ. We therefore propose a matched-budget evaluation pipeline that separates feasibility, anytime quality, tail latency, robustness, and bound validity, and requires component ablation. The article is a methodological synthesis rather than a new computational benchmark: It does not rank hybrid families. Its contribution is a falsifiable language for moving from algorithm names to reproducible decision mechanisms and from isolated benchmark wins to evidence appropriate for industrial use.

Keywords: Hybrid metaheuristics, Matheuristics, Smart manufacturing, Scheduling, Vehicle routing, Supply-chain optimization, Algorithm benchmarking, Reproducibility.

1 | Introduction

Smart manufacturing does not create one optimization problem. It creates a web of decisions whose horizons, constraints, data rates, and consequences differ. A production sequence changes machine states and energy demand; maintenance blocks capacity; inventory commitments alter the volume and timing of transport; disruptions make yesterday's feasible plan obsolete. Reviews of flexible job-shop scheduling, energy-efficient

 Corresponding Author: azadeh.shemshad@gmail.com

 <https://doi.org/10.48314/ceti.v3i1.8>



Licensee System Analytics. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0>).

job shops, and Industry 4.0 supply-chain optimization consequently report heterogeneous models rather than a single canonical task [1–4]. The central design question is therefore not “which metaheuristic is best?” but “which computational responsibilities must be combined for this decision, and under what evidence contract?”

Hybrid metaheuristics combine search components, exact or approximate solvers, simulation, learning, or domain rules. The foundational taxonomies distinguish whether complete algorithms or internal operators are combined and whether they act sequentially or cooperatively [5–9]. Metaheuristics sharpen another boundary: Mathematical-programming structure may generate neighborhoods, relaxations, decompositions, or subproblems inside a heuristic design [10–12]. These classifications explain composition, but industrial evaluation also needs the semantics of the interface: The exchanged object, its feasibility status, the trigger for exchange, the budget charged to each component, and the claim that the resulting evidence permits.

This article proposes the decision contract as the unit of description. The contract has five layers: Decision model, component roles, exchange, resources, and evidence. The layers prevent three common category errors: 1) a familiar pairing such as Genetic Algorithm (GA)–local search is not a method until its representation, decoder, neighborhood, and acceptance logic are given, 2) adding an exact solver does not automatically provide a lower bound or convergence certificate, and 3) a faster search loop is not a real-time system unless sensing, inference, repair, communication, and actuation are included in latency. *Fig. 1* gives the architecture.

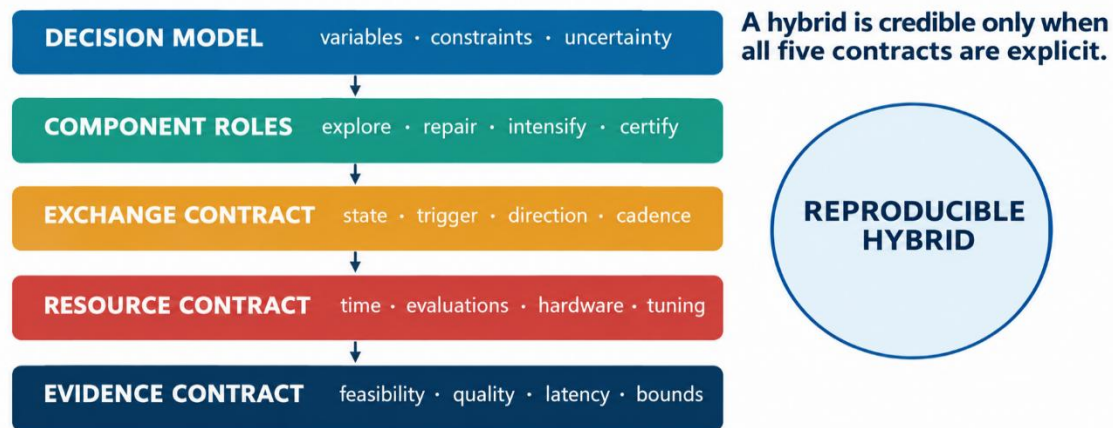


Fig. 1. The decision-contract architecture.

Unlike a catalogue of algorithm families, the figure follows the information that must remain coherent from the industrial decision model to the evidential claim. A missing layer does not merely reduce reporting detail; it prevents the hybrid from being reproduced or its result from being interpreted.

The contribution is fourfold: 1) the paper turns established hybrid taxonomies into a reproducible specification, 2) it synthesizes literature across manufacturing scheduling, routing, supply-chain design, adaptive search, and algorithm configuration without transferring performance claims between incompatible problems, 3) it supplies a domain-to-mechanism map and a gap analysis, and 4) it gives a matched-budget protocol in which feasibility precedes objective quality and every hybrid component is tested by ablation. The paper is a critical perspective, not a systematic review and not a report of new computational experiments; accordingly, it makes no universal superiority claim.

2 | Literature and the Interface Gap

2.1 | From Metaheuristic Families to Hybrid Architecture

Early syntheses established metaheuristics as reusable search frameworks while warning that labels conceal design choices [5]. Talbi classified hybrids by level and mode of cooperation [6], and Raidl's unified view made the distinction between high-level combinations of complete methods and low-level integration of functions or operators explicit [7]. Subsequent books and handbooks broadened hybridization to mathematical programming, constraint programming, simulation, data mining, and learning [8], [9], [13]. This trajectory matters because the object being exchanged changed: It may be a complete solution, a partial assignment, a bound, a neighborhood, a prediction, or a policy. A taxonomy based only on algorithm names cannot describe those semantics.

Two neighboring traditions further changed the design space. Matheuristics use mathematical models and solver mechanisms to construct heuristic frameworks; contemporary surveys distinguish model-based neighborhoods, relaxations, decomposition, restricted masters, and heuristic branch-and-bound variants [10–12]. Hyper-heuristics move one level upward by selecting or generating heuristics rather than directly manipulating problem solutions [14]. These traditions show why “hybrid” should not mean “two metaphors joined by a hyphen”. The scientific content lies in the division of computational labor and in the validity of the information passed between components.

2.2 | Domain-Specific Hybridization

Flexible job-shop scheduling combines operation sequencing with machine assignment and is extended in practice by setup, transport, maintenance, energy, and uncertainty constraints [3]. Taillard and Brandimarte supplied enduring benchmark families for basic and flexible shops [15], [16]. More recent reviews emphasize multiobjective, dynamic, and energy-aware settings [2], [7], [17]. In these models, a decoder or repair procedure may determine more of the effective algorithm than the population update itself. A hybrid claim is therefore incomplete unless it states whether feasibility is preserved, repaired, penalized, or delegated to a subsolver.

Vehicle routing reveals a different interface problem. Solomon's time-window instances made customer sequence, vehicle assignment, capacity, waiting, and service timing jointly testable [18]. Adaptive Large-Neighborhood Search (ALNS) introduced a transparent cooperation mechanism: Destroy and repair operators compete under an adaptive score [19], [20]. Hybrid genetic search integrates population diversity with aggressive neighborhood education [21]. These mechanisms are transferable, but their reported performance is not automatically transferable to live dispatch, electric fleets, inventory routing, or stochastic travel times; each variant changes both the feasible set and the evaluation budget.

Supply-chain network design connects strategic binary commitments to tactical flows, inventories, scenarios, and recourse. Reviews under uncertainty and sustainability document broad variation in objective functions, risk measures, time horizons, and policy constraints [1], [22–24]. This structure can justify decomposition or simulation–optimization, but only if the master–subproblem interface is stated. An exactly solved recourse problem supplies a well-defined conditional value; it does not necessarily supply a global lower bound when the master remains heuristic. The certificate status must therefore be reported independently from solution quality.

2.3 | Adaptation, Learning, and Configuration

Adaptive hybrids now include operator selection, surrogate models, reinforcement learning, and digital-twin feedback. A recent review of ensemble metaheuristics and reinforcement learning in manufacturing scheduling organized learning roles around parameter control, strategy selection, and local-search choice [25]. Simulation–optimization reviews likewise identify expensive evaluation and distribution change as central difficulties [26]. Learning can reduce wasted evaluations or respond to disturbances, but it creates a training

distribution, an update cost, and a failure mode under extrapolation. Those costs belong in the resource and evidence contracts rather than outside the comparison.

Algorithm configuration research makes the same point from another direction. Parameter Iterated Local Search (ParamILS), Sequential Model-Based Algorithm Configuration (SMAC), and IRACE treat parameter selection as an optimization problem [27–29]; parameter-control surveys distinguish values fixed before a run from those adapted during it [30]. Consequently, reporting a tuned hybrid against an untuned baseline confounds algorithm design with configuration effort. Training and test instances must be separated, tuning budgets disclosed, and the baseline granted a comparable opportunity to configure. Otherwise, the experiment measures unequal engineering investment.

2.4 | What the Literature Still Leaves Unresolved

The literature supplies rich taxonomies and strong domain solvers, but four interfaces are repeatedly under-specified. Representation is often separated from algorithm comparison; feasibility handling is hidden inside a decoder; tuning and surrogate training are excluded from cost; and evidence is compressed into a final average objective. Best-practice studies in optimization benchmarking have long argued for explicit problem sets, resource metrics, performance distributions, and fair stopping rules [31–33]. Statistical guidance similarly favors paired multi-instance analysis and multiplicity control over isolated mean comparisons [34–37]. The remaining gap is architectural: These requirements need to be bound to the hybrid’s internal exchanges, not appended as a generic checklist.

Table 1. Evolution of the literature and the unresolved interface carried into the present framework.

Literature Strand	Primary Contribution	What Remains Ambiguous	Decision-Contract Response
Hybrid taxonomy [6–9]	Level, sequence, cooperation	Meaning and validity of exchanged objects	Name state, trigger, direction, cadence, and feasibility status
Matheuristics [10–12]	Model-based neighborhoods and subsolvers	Whether bounds or convergence survive heuristic coupling	Separate incumbent quality from certificate status
Domain hybrids [3], [4], [17], [19], [20–26]	Problem-specific operators and adaptive mechanisms	Transfer across variants, horizons, and data regimes	Freeze the decision model and disturbance process
Algorithm configuration [27–30]	Systematic tuning and adaptive control	Unequal tuning budgets and test leakage	Declare training split and charge configuration cost
Benchmarking statistics [31–37]	Fair budgets, paired tests, multiplicity control	Weak connection to component-level causal claims	Cross the full pipeline with controlled ablations

3 | The Decision-Contract Taxonomy

Contract 1 (Decision Model). Let an instance be described by data ξ and a feasible set $X(\xi)$. A hybrid returns an incumbent $x \in X(\xi)$, an archive $A \subseteq X(\xi)$, or a policy π that maps observed states to decisions. The objective may be scalar $f(x; \xi)$, vector-valued $F(x; \xi)$, scenario-based $E[f(x; \xi, \omega)]$, or robust over an uncertainty set. This is not notation for its own sake: A permutation that is feasible for routing may require a decoder for machine assignment; a Pareto archive cannot be compared by a scalar error unless the preference model is declared. The first contract records variables, constraints, uncertainty, horizon, objective direction, and the status of each output.

Contract 2 (Component roles). Components should be named by responsibility before they are named by family. Useful roles include exploration, intensification, feasibility restoration, diversification, prediction, exact evaluation, bounding, and coordination. One algorithm can fill several roles, and the same family can fill different roles in different hybrids. This role-first description makes ablation interpretable: Removing a local search tests intensification; removing an exact recourse solver changes evaluation validity as well as speed. An ablation is fair only when the remaining system still satisfies the same decision contract.

Contract 3 (Exchange). The exchange contract specifies five elements: The object, its semantics, direction, trigger, and cadence. A relay may pass one feasible incumbent after a fixed budget. A cooperative team may share elites every k evaluations. A low-level learning component may update operator probabilities after each accepted move. The same pair of algorithm names can therefore implement fundamentally different systems. Fig. 2 locates common patterns by component autonomy and exchange cadence; it is an interface atlas, not a ranking.

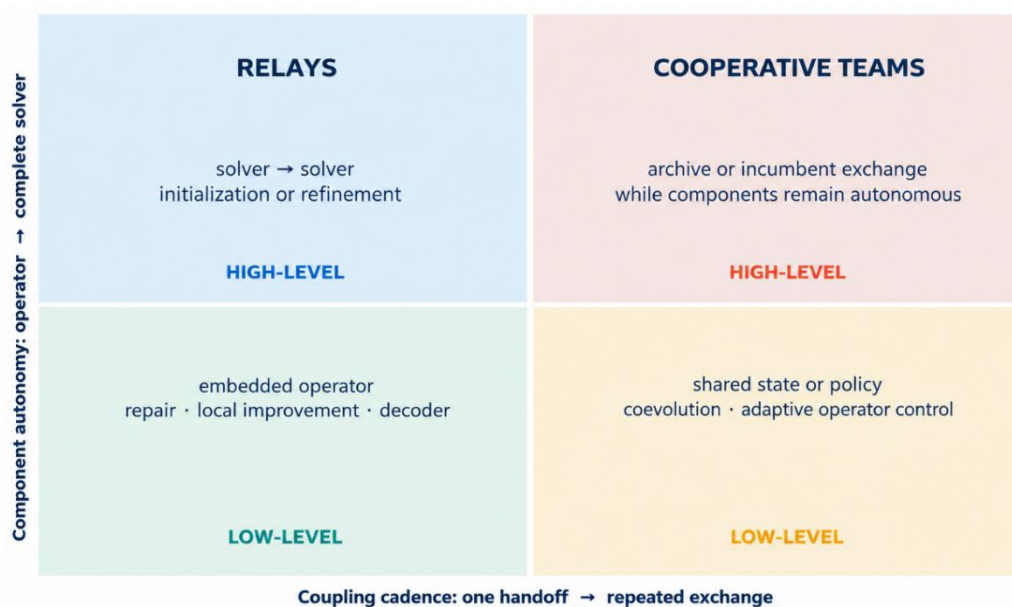


Fig. 2. Interface atlas for hybrid composition.

Vertical position records whether the components are internal operators or autonomous solvers; horizontal position records whether information is handed off once or exchanged repeatedly. The quadrant identifies the coordination burden that must be reported, not expected performance.

Contract 4 (Resources). Every computational activity that differs between methods belongs to the resource contract: Preprocessing, model training, parameter tuning, solver calls, simulation replications, communication, feasibility repair, and parallel hardware. Function evaluations are fair only when their costs are comparable. Wall-clock time is meaningful only when hardware, parallelism, warm starts, and measurement boundaries are fixed. For online use, report the latency distribution of the entire decision path, particularly p_{95} and p_{99} values, because average latency can hide deadline violations.

Contract 5 (Evidence). The evidence contract links outputs to claims. Feasible incumbents support claims about achieved objective values. Valid relaxations can support bounds. Repeated paired runs support probabilistic comparisons on the tested instance distribution. Out-of-sample event streams support limited robustness claims. None of these alone establishes universal dominance or production readiness. The No-Free-Lunch result reinforces the need to specify a problem distribution rather than expect one optimizer to dominate without qualification [38].

Table 2. Minimal reproducibility record for a hybrid metaheuristic.

Contract	Required Record	Diagnostic Question
Decision model	Variables; constraints; horizon; uncertainty; objective; output type	Would another group construct the same feasible set?
Component roles	Role, input, output, stopping rule, and failure behavior for each component	What computational responsibility disappears in an ablation?
Exchange	Object, semantics, direction, trigger, cadence, synchronization	Can an exchanged object be infeasible, stale, or incomparable?
Resources	Tuning, training, evaluations, solver calls, hardware, parallelism, end-to-end time	Which costs were excluded from the reported budget?
Evidence	Instances, seeds, metrics, bounds, statistical unit, artifacts, claim boundary	What exact claim would the experiment falsify?

4 | Domain Design Maps

4.1 | Flexible Job-Shop and Production Scheduling

In a flexible job shop, the representation must encode both operation sequence and eligible-machine assignment. Feasibility covers precedence, non-overlap, eligibility, and any setup, transport, worker, or maintenance constraint. A population method can explore assignments while a critical-path neighborhood intensifies sequencing; a mixed-integer subproblem can optimize a restricted neighborhood; a repair operator can restore capacity after a disruption. These are different contracts. Comparisons should use identical instance versions, objective definitions, stopping budgets, and energy models [3], [4], [15–17]. Multiobjective studies should release nondominated sets and declare indicator reference points; Non-dominated Sorting Genetic Algorithm II (NSGA-II) is a search mechanism, not a substitute for decision preference [39].

4.2 | Routing and Intralogistics

Routing hybrids must expose the decoder for vehicle assignment and timing. Waiting, service duration, depot limits, split deliveries, charging, and route duration change feasibility. Adaptive Large Neighborhood Search (ALNS) is attractive when destroy–repair operators map naturally to disruptions, while hybrid genetic search makes diversity an explicit system state [19–21]. For live routing, the event stream must be replayed identically across algorithms. A method that obtains a better offline route after minutes cannot be ranked against a dispatcher required to return a feasible action in seconds without a deadline-aware metric.

4.3 | Supply-Chain Design and Inventory Decisions

Network design hybrids often benefit from a structural split: A master selects facilities, technologies, or contracts, and a subproblem evaluates flows, inventories, or scenario recourse [1], [22–24]. The contract must state whether infeasible master decisions are rejected, penalized, repaired, or evaluated through slack. Under uncertainty, scenarios used for design must be separated from those used for assessment. Robustness should be expressed through out-of-sample loss, service violations, regret, or recourse feasibility—not by relabeling a nominal solution as resilient.

4.4 | Maintenance, Energy, and Simulation-Coupled Operations

Maintenance and energy decisions create hidden evaluators. Energy may be computed from a machine-state model, measured physically, or estimated by simulation; these are not interchangeable. Maintenance may depend on degradation estimates that are themselves uncertain. When simulation or a surrogate evaluates candidates, report replication policy, random-number control, prediction error, update frequency, and the fraction of decisions checked by the true evaluator [4], [26], [40]. A digital twin adds value only if the data-to-decision loop is timed and its model discrepancy is examined.

Table 3. Domain-to-mechanism map.

Decision Setting	Structural Burden	Candidate Hybrid Mechanism	Evidence That Can Reject It
Flexible job shop	Sequence + machine assignment; precedence; dynamic events	Population exploration + critical-path improvement or exact neighborhood	Feasibility at deadline; paired anytime gap; ablation of neighborhood
Vehicle routing	Sequence + vehicle + timing; online requests	ALNS or diversity-controlled population search with repair	Event-replay service rate; p95 latency; route cost under equal time
Supply-chain design	Strategic binaries + flow/recourse; uncertainty	Heuristic master + exact or controlled-error subproblem	Out-of-sample recourse; incumbent quality; independently valid bound
Maintenance scheduling	Production precedence + downtime + degradation	Rolling horizon + scenario evaluation + feasibility restoration	Violation rate; rescheduling stability; update latency
Energy-aware operations	State-dependent energy + competing objectives	Multiobjective archive + metered/simulated evaluator or surrogate	Pareto quality; energy-model validation; surrogate error under shift

* Candidate mechanisms are hypotheses to test, not prescriptions.

5 | Evidence Architecture

5.1 | Freezing the Comparison Before Execution

A defensible experiment predeclares the instance roster, versions and hashes; objective direction; feasibility rules; event stream; tuning split; independent seeds; hardware and parallelism; and stopping rule. The experimental unit is normally the instance–seed pair, not an iteration. The simplest meaningful heuristic and a competitive domain baseline should be included. If a commercial solver is used as comparator or component, its version, settings, warm starts, and time accounting must be stated.

5.2 | Measure a Vector of Outcomes

Feasibility comes first. Conditional on feasibility, report objective quality, time-to-target, and anytime trajectories. For a minimization problem with positive reference value b_i , relative deviation $100 \frac{f_i - b_i}{b_i}$ is interpretable only when b_i is identified and comparable. It is undefined at $b_i = 0$ and should not be pooled across unrelated objectives. Performance profiles can summarize the fraction of instances solved within a factor of the best observed performance [32], but the underlying cost measure and failure convention must be stated. Online systems additionally require tail latency, deadline-feasible quality, and recovery after disturbances.

5.3 | Separate Statistical Significance from Operational Relevance

For multiple algorithms evaluated on matched instance sets, rank-based omnibus and post-hoc procedures can be appropriate [34], [35], [37]. Holm [36] correction controls a declared family of comparisons. These procedures do not repair tuning leakage, dependence between reused runs, or heterogeneous objectives. Report paired differences and uncertainty, not only ranks and p-values. A practically meaningful margin should be set in the units of the decision—minutes of tardiness, energy, service failures, or computation time. Failure to reject a difference is not proof of equivalence.

5.4 | Make Hybrid Causality Testable

Ablation asks whether each component earns its cost. Remove one component while preserving the remaining representation, initialization, budget, and tuning protocol. If removal changes feasibility, report that fact rather than interpreting the comparison as a pure quality effect. Factorial ablations are useful when

two components interact. The key output is not “component A helps on average,” but the boundary: The instance features, budgets, or disturbance regimes under which its incremental value becomes positive.

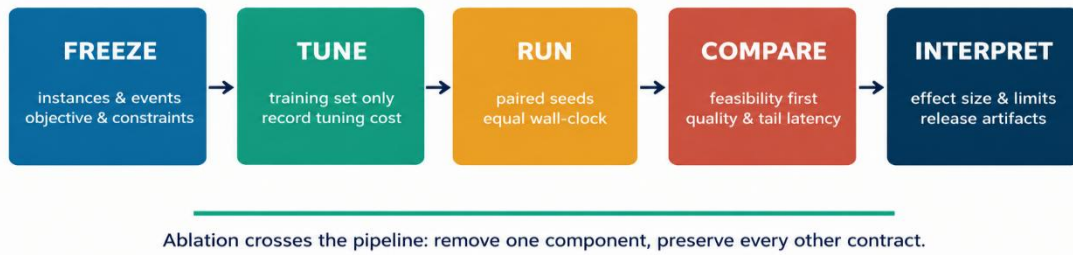


Fig. 3. Matched-budget evidence pipeline.

The sequence prevents test leakage and metric substitution. Component ablation is not an isolated final experiment; it must preserve the frozen problem, tuning split, execution budget, and interpretation rules across the pipeline.

5.5 | Reproducibility Package

The release should contain the complete instance manifest, preprocessing, configuration space, selected configurations, seeds, raw run-level data, feasibility checker, scripts for every table and figure, and software licences. For dynamic problems, include the timestamped event stream and the rule for deciding when a new plan becomes active. For surrogate or learning components, include training data provenance, model checkpoints, update logic, and error diagnostics. A paper can be reproducible without being universally generalizable; reproducibility fixes what was done, while generalization requires additional evidence.

6 | Research Gaps and Testable Agenda

The decision-contract view changes the research agenda from inventing additional pairings to testing interfaces. Five gaps are especially consequential: 1) representation-aware comparisons are rare: Two studies may share an algorithm label while solving different feasible sets, 2) certificate status is often blurred when exact and heuristic components are combined, 3) adaptation costs and distribution shift are incompletely charged, 4) end-to-end latency and post-disruption feasibility receive less attention than final objective values, and 5) transferable design rules require instance features that explain when a component helps, not only aggregate rankings.

Table 4. Research gaps and falsifiable studies.

Gap	Why Current Evidence is Insufficient	Falsifiable Next Study
Representation sensitivity	Algorithm labels hide decoder and repair effects	Cross decoders and search engines factorially under one feasible-set checker
Certificate integrity	Exact components are sometimes treated as if they certify the whole hybrid	Audit incumbent and bound provenance at every interface
Adaptation under shift	Learning is evaluated on events resembling training	Predeclare regime shifts; measure recovery, regret, and update cost
Deployment timing	Search-loop time omits sensing, transfer, repair, and actuation	Instrument the end-to-end path and report deadline-feasible quality
Conditional transfer	Average ranks do not reveal where components add value	Model component gain against instance and disturbance features on held-out suites

A useful multi-domain benchmark should therefore not pool makespan, route cost, and inventory loss into one score. It should preserve separate suites while sharing a common contract schema, resource ledger,

artifact format, and statistical protocol. Cross-domain transfer can then be studied at the level of mechanisms—repair, decomposition, operator adaptation, or surrogate control—rather than by assuming that a named algorithm carries a stable advantage across problems.

7 | Limitations

This article is a methodological synthesis. It does not report a Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA)-style systematic search, estimate the prevalence of hybrid patterns, or establish comparative performance. The cited studies were selected to anchor foundational taxonomies, domain formulations, representative mechanisms, and evaluation practice. The proposed contracts and maps should therefore be tested on openly released benchmark suites before they are treated as validated selection rules. The framework also does not eliminate judgment: Defining an operationally meaningful equivalence margin, risk measure, or deadline remains a domain decision.

8 | Conclusion

A hybrid metaheuristic becomes scientifically intelligible when its interfaces are more explicit than its name. The decision-contract framework links five objects: The industrial decision model, the responsibilities assigned to components, the information exchanged between them, the resources consumed, and the evidence used to justify a claim. This architecture accommodates relay and cooperative hybrids, matheuristics, adaptive operator selection, surrogate assistance, and learning-guided search without treating them as interchangeable. It also enforces a disciplined empirical order: Feasibility, matched budgets, anytime quality, tail latency, robustness, certificate validity, and ablation. The resulting contribution is not a universal recipe for choosing algorithms. It is a way to formulate hybrid design as a set of claims that another researcher can reproduce, challenge, and—when the evidence warrants—carry into industrial decision making.

Acknowledgments

The authors have no additional acknowledgments to declare.

Author Contributions

M. F.; Conceptualization, literature review, investigation, and writing—original draft. A. Sh.; Conceptualization, methodology, critical analysis, and writing—review and editing. Both authors reviewed and approved the final manuscript.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Data Availability

No new datasets were generated or analyzed in this methodological synthesis. The sources supporting the findings are cited in the manuscript.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this article.

References

- [1] Xu, Z., Elomri, A., Baldacci, R., Kerbache, L., & Wu, Z. (2024). Frontiers and trends of supply chain optimization in the age of industry 4.0: An operations research perspective. *Annals of Operations Research*, 338(2), 1359–1401. <https://doi.org/10.1007/s10479-024-05879-9>

- [2] Zhang, W., Bao, X., Hao, X., & Gen, M. (2025). Metaheuristics for multi-objective scheduling problems in industry 4.0 and 5.0: A state-of-the-arts survey. *Frontiers in Industrial Engineering*, 3, 1540022. <https://doi.org/10.3389/fieng.2025.1540022>
- [3] Dauzère-Pérès, S., Ding, J., Shen, L., & Tamssaouet, K. (2024). The flexible job shop scheduling problem: A review. *European Journal of Operational Research*, 314(2), 409–432.
- [4] Gahm, C., Denz, F., Dirr, M., & Tuma, A. (2016). Energy-efficient scheduling in manufacturing companies: A review and research framework. *European Journal of Operational Research*, 248(3), 744–757. <https://doi.org/10.1016/j.ejor.2015.07.017>
- [5] Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3), 268–308. <https://doi.org/10.1145/937503.937505>
- [6] Talbi, E. G. (2002). A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8(5), 541–564. <https://doi.org/10.1023/A:1016540724870>
- [7] Raidl, G. R. (2006). A unified view on hybrid metaheuristics. *International workshop on hybrid metaheuristics* (pp. 1–12). Springer. https://doi.org/10.1007/11890584_1
- [8] Blum, C., Puchinger, J., Raidl, G. R., & Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6), 4135–4151. <https://doi.org/10.1016/j.asoc.2011.02.032>
- [9] Blum, C., & Raidl, G. R. (2016). *Hybrid metaheuristics: Powerful tools for optimization*. Springer. <https://doi.org/10.1007/978-3-319-30883-8>
- [10] Maniezzo, V., Stützle, T., & Voß, S. (2009). *Matheuristics: Hybridizing metaheuristics and mathematical programming*. Springer. <https://doi.org/10.1007/978-1-4419-1306-7>
- [11] Boschetti, M. A., & Maniezzo, V. (2024). Contemporary approaches in matheuristics an updated survey. *Annals of Operations Research*, 343(2), 663–700. <https://doi.org/10.1007/s10479-024-06302-z>
- [12] Puchinger, J., & Raidl, G. R. (2005). Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. *International work-conference on the interplay between natural and artificial computation* (pp. 41–53). Springer, Berlin, Heidelberg. https://doi.org/10.1007/11499305_5
- [13] Gendreau, M., & Potvin, J. Y. (2019). *Handbook of metaheuristics* (3rd ed.). Springer. <https://doi.org/10.1007/978-3-319-91086-4>
- [14] Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Qu, R. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12), 1695–1724. <https://doi.org/10.1057/jors.2013.71>
- [15] Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2), 278–285. [https://doi.org/10.1016/0377-2217\(93\)90182-M](https://doi.org/10.1016/0377-2217(93)90182-M)
- [16] Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 41(3), 157–183. <https://doi.org/10.1007/BF02023073>
- [17] Neufeld, J. S., Schulz, S., & Buscher, U. (2023). A systematic review of multi-objective hybrid flow shop scheduling. *European Journal of Operational Research*, 309(1), 1–23. <https://doi.org/10.1016/j.ejor.2022.08.009>
- [18] Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2), 254–265. <https://doi.org/10.1287/opre.35.2.254>
- [19] Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4), 455–472. <https://doi.org/10.1287/trsc.1050.0135>
- [20] Pisinger, D., & Ropke, S. (2018). Large neighborhood search. In *Handbook of metaheuristics* (pp. 99–127). Springer. https://doi.org/10.1007/978-3-319-91086-4_4
- [21] Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., & Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3), 611–624. <https://doi.org/10.1287/opre.1120.1048>
- [22] Farahani, R. Z., Rezapour, S., Drezner, T., & Fallah, S. (2014). Competitive supply chain network design: An overview of classifications, models, solution techniques and applications. *Omega*, 45, 92–118. <https://doi.org/10.1016/j.omega.2013.08.006>

- [23] Govindan, K., Fattahi, M., & Keyvanshokoo, E. (2017). Supply chain network design under uncertainty: A comprehensive review and future research directions. *European Journal of Operational Research*, 263(1), 108–141. <https://doi.org/10.1016/j.ejor.2017.04.009>
- [24] Eskandarpour, M., Dejax, P., Miemczyk, J., & Péton, O. (2015). Sustainable supply chain network design: An optimization-oriented review. *Omega*, 54, 11–32. <https://doi.org/10.1016/j.omega.2015.01.006>
- [25] Fu, Y., Wang, Y., Gao, K., & Huang, M. (2024). Review on ensemble meta-heuristics and reinforcement learning for manufacturing scheduling problems. *Computers and Electrical Engineering*, 120, 109780. <https://doi.org/10.1016/j.compeleceng.2024.109780>
- [26] Ghasemi, A., Farajzadeh, F., Heavey, C., Fowler, J., & Papadopoulos, C. T. (2024). Simulation optimization applied to production scheduling in the era of industry 4.0: A review and future roadmap. *Journal of Industrial Information Integration*, 39, 100599. <https://doi.org/10.1016/j.jii.2024.100599>
- [27] Hutter, F., Hoos, H. H., Leyton-Brown, K., & Stützle, T. (2009). ParamILS: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36, 267–306. <https://doi.org/10.1613/jair.2861>
- [28] Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. *International conference on learning and intelligent optimization* (pp. 507–523). Springer. https://doi.org/10.1007/978-3-642-25566-3_40
- [29] López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., & Stützle, T. (2016). The IRACE package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3, 43–58. <https://doi.org/10.1016/j.orp.2016.09.002>
- [30] Eiben, A. E., & Smit, S. K. (2011). Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1), 19–31. <https://doi.org/10.1016/j.swevo.2011.02.001>
- [31] Beiranvand, V., Hare, W., & Lucet, Y. (2017). Best practices for comparing optimization algorithms. *Optimization and Engineering*, 18(4), 815–848. <https://doi.org/10.1007/s11081-017-9366-1%0A%0A>
- [32] Dolan, E. D., & Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2), 201–213. <https://doi.org/10.1007/s101070100263>
- [33] Bartz-Beielstein, T., Doerr, C., Berg, D. V. D., Bossek, J., Chandrasekaran, S., Eftimov, T., ..., & Weise, T. (2020). *Benchmarking in optimization: Best practice and open issues*. <https://doi.org/10.48550/arXiv.2007.03488>
- [34] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1), 1–30. <https://jmlr.org/papers/v7/demsar06a.html>
- [35] Garcia, S., & Herrera, F. (2008). An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9(12), 2677–2694. <https://jmlr.org/papers/v9/garcia08a.html>
- [36] Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 7(2), 65–70. <https://www.jstor.org/stable/4615733>
- [37] Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1), 3–18. <https://doi.org/10.1016/j.swevo.2011.02.002>
- [38] Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67–82. <https://doi.org/10.1109/4235.585893>
- [39] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2), 182–197. <https://doi.org/10.1109/4235.996017>
- [40] Sitahong, A., Wang, G., Yuan, Y., Wubuli, A., Mo, P., & Chen, Y. (2025). Energy-efficient scheduling in dynamic flexible job shops: A review. *Sustainability*, 17(24), 11009. <https://doi.org/10.3390/su172411009>