applied sciences

MDPI

Article

Experimental Validation of Configurational AI Architectures

Roman Yavich [1,*] and Vladimir Rotkin [2]

[1] Department of Mathematics, Ariel University, Ariel 4070000, Israel
[2] New Israel Association of Sciences, Netanya 4280000, Israel; ed.el.fromru@gmail.com
* Correspondence: romany@ariel.ac.il

Featured Application

The reproducible evaluation protocol and the procedural hybrid pattern examined here (a neural network generates a candidate configuration; an independent verifier checks all hard constraints; and an exact deterministic solver is invoked only on verification failure) are intended to inform the design of configurator systems in education, engineering, and e-commerce, in which recommendations must satisfy mandatory budget, quality, safety, and compatibility rules, and in which an auditable log of the accepted route is required for compliance and quality assurance. The evidence reported below was obtained on a synthetic corpus and therefore supports the architectural pattern itself; deployment in any of these settings would require separate validation on domain data.

Abstract

Configurable intelligent design is formalized here as the selection of a consistent configuration from interdependent alternatives under constraints of budget, quality, risk, and compatibility. It is examined not as a universal solver but as a class of domain-specific systems, each defined for a task family with explicitly formalizable variables, hard constraints, and objectives. Locally correct predictions for individual components do not guarantee a globally feasible solution. This study compares direct neural-network inference, the exact HIM-D solver, and a procedural hybrid whose network candidate is checked by an independent verifier, with an exact search invoked whenever a constraint is violated. A reproducible synthetic corpus of 24,000 problems across educational, engineering, and commercial scenarios was assessed for feasibility, optimality, robustness to distribution shift, repeatability, and latency. Pure neural-network models were feasible in only 17.40% and 19.12% of cases and degraded sharply out of distribution, whereas the hybrid maintained 100.00% feasibility and 83.08% joint optimality. A staged ablation identifies pairwise incompatibility density as the dominant source of that degradation, and constraint-aware correction alone raises feasibility without reaching a guarantee. The results therefore support separating probabilistic candidate generation from independent formal verification within a bounded class of specialized configuration problems, not a universal solver.

Keywords: configurational AI architectures; hierarchical imitation models; constraint satisfaction problems; constraint programming; combinatorial optimization; formal verification; hybrid artificial intelligence; out-of-distribution generalization; ablation analysis; reproducible experiment

Check for updates

Academic Editor: Jose María Alvarez Rodríguez

Received: 19 August 2026
Revised: 12 September 2026
Accepted: 16 September 2026
Published: 18 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Artificial intelligence systems are increasingly used to select educational content, engineering components, production modes, and commercial offers. In all of these cases,

the final output is not a single prediction but a coordinated configuration of several interdependent elements. Such a configuration must simultaneously satisfy user preferences, numerical thresholds, logical dependencies, and an optimality criterion. A substantively convincing answer may remain practically unacceptable if even one mandatory condition is violated. The evaluation of such systems therefore cannot be limited to average accuracy or semantic similarity; it requires independent validation, quantitative assessment of deviations from the optimum, robustness analysis under new combinations of conditions, and a reproducible decision-making log.

In a previous conceptual study [1], an architecture for configurable intelligent design based on hierarchical imitation models (HIMs) was proposed as a deterministic alternative to neural-network machine learning for domains in which correctness, transparency, and personalization of results are prioritized. The configurable object was represented as a vector of variables, constraints were defined by a system of functionals, and a library of reference models was intended to be updated through controlled user interaction. Step-by-step validation, computational-complexity management, and an iterative cycle of model testing, tuning, and modification were also proposed.

The original architecture did not assume that a single universal deterministic model would cover arbitrary application domains. Instead, each local implementation was intended to define a bounded class of related tasks requiring particularly high accuracy, unambiguous interpretation, and verifiable compliance with domain-specific conditions. For every such class, a separate configurational module is constructed that represents the permissible variables, constraints, dependencies, and solution procedures. The present study operationalizes this principle through one controlled finite configuration family and does not claim to implement the full range of hierarchical and reference-learning mechanisms proposed in [1].

The scientific value of that original work lay in its comprehensive description of the architecture and methodology; however, its evidence base remained largely conceptual. A large, independent corpus of problems, predefined metrics, interval estimates, a separate generalization test outside the training distribution, and a comparison with direct neural-network solvers were not presented. This gap connects to a broader issue at the intersection of configuration, combinatorial optimization, and machine learning. Classical work treats configuration as a search for a consistent set of components in a finite space under a system of constraints [2,3]. Modern reviews show that machine learning can either strengthen a formal solver or attempt to replace it with a fast predictor [4,5]. The latter approach reduces latency but, without a dedicated verification mechanism, does not guarantee feasibility. A similar gap is observed in research on language models for optimization: a high-quality problem formulation or objective function does not eliminate constraint violations [6–9].

Two boundaries have to be drawn explicitly, because the contribution of this paper is defined by both. The first separates the present study from the earlier conceptual paper [1]. That work contributed the architecture itself: the representation of a configurable object as a vector of variables constrained by a system of functionals, the hierarchy of reference models, and the proposal that the library be maintained through controlled user interaction. It contained no corpus, no predefined metrics, no interval estimates, and no comparison against a learned solver. The present study contributes none of the architecture and all of the measurement: a fixed generator, a single corpus with a separate out-of-distribution partition, an operational definition of every indicator, paired interval estimation, and an explicit decision log. The second boundary separates the procedural hybrid examined here from jointly trained neuro-symbolic models, and is developed in Section 2.4. Briefly, the two components of the present architecture are coupled only through the control flow: the verifier and the solver share no parameters with the network, receive no gradient from it,

and act after inference has finished. What has not been measured on a common corpus is how much reliability that purely procedural coupling buys relative to the direct neural candidate it corrects, and that is the quantity reported here.

The objective of this paper is to quantitatively test the separation-of-functions principle within a specialized configuration system: a probabilistic module generates a candidate, whereas an independently validated formal module is responsible for the properties that require guarantees. Importantly, this work does not pose the trivial task of proving the superiority of exhaustive search over an approximator: in the chosen problem space, HIM-D is an exact solver and a reference upper bound. Nor does the experiment test the universal superiority of a particular solver; it examines whether, within a predefined and fully formalized class of tasks, independent verification and conditional deterministic fallback prevent locally plausible neural predictions from being accepted as globally valid configurations. The substantive question concerns the proportion of infeasible direct neural-network outputs, their sensitivity to out-of-distribution (OOD) shift, and the extent to which independent verification with conditional backup search changes robustness, optimality, and latency. To this end, a fixed generator, a single corpus, predefined metrics, paired analysis, a separate OOD test, and an explicit solution-path log are used. The published results of ConstraintBench [6] are reported only as an external benchmark and are not combined with the internal experiment.

The study is organized around five research questions. RQ1: Does HIM-D provide a higher feasibility rate than pure neural-network models? RQ2: Does this advantage persist under controlled distribution shift? RQ3: Does the hybrid architecture combine neural-network flexibility with a formal feasibility guarantee? RQ4: What is the relationship between robustness gain, objective-function quality, and computational latency? RQ5: Which component of the distribution shift is responsible for the loss of feasibility, and does the architectural conclusion survive comparison with constraint-aware correction methods and with a general-purpose constraint-programming backend?

The scientific novelty of this study lies in translating the conceptual provisions of the original architecture into measurable variables within a single, reproducible experiment: a fully deterministic and testable implementation of HIM-D was formalized; a corpus of 24,000 tasks with separate OOD testing was constructed; a uniform comparison of the deterministic, neural-network, and hybrid approaches was conducted with paired statistical analysis; an external comparison with published results for language models on ConstraintBench was performed; and a reproducibility package with an operational description of the pipeline was prepared. The study further includes six analyses that bound the interpretation of these results: a staged ablation that attributes the out-of-distribution decline to individual generation factors, a comparison against constraint-aware correction applied to the same neural candidate, a comparison against general-purpose exact solvers, a scaling study beyond the 1024-configuration space, a measurement of what the guarantee costs when the verifier's rule set is incomplete, and five independent replications of the whole pipeline together with a sensitivity analysis of the objective-function weights. The resulting contribution is therefore architectural rather than universal: it demonstrates, under controlled conditions, how a specialized deterministic module can constrain the operational behavior of a probabilistic candidate generator. Generalization in this context refers to new instances and controlled distribution shifts within the defined task family, not to unrestricted transfer across unrelated domains or arbitrary configuration structures. Table 1 summarizes the transition from the conceptual provisions to the specific operationalizations and metrics used in Sections 4 and 5.

Table 1. Transition from the conceptual provisions of the original architecture to the measured variables of the present study.

Position in the Original Architecture	Operationalization in the Present Study	Primary Indicator
Determinism	Explicit constraints and complete, verifiable search	Proportion of feasible and optimal solutions
Hierarchy of models	Levels of domain, solution group, variant, and constraint functional	Correctness across three subject domains
Reference library	Fixed library structure with versioned parameters	Reproducibility at a fixed granularity
Computational-complexity management	Finite space of $4^5 = 1024$ configurations	Latency per task
Dialogue with the user	Machine-readable requirement vector and explainable log	Traceability of selected configurations
Step-by-step validation	Training, validation, ID-test, and OOD-test splits	Separation of results across distributions

Read row by row, Table 1 shows that each conceptual commitment of the original HIM architecture [1] is paired with a concrete operationalization and a primary indicator; these indicators reappear as the metrics defined formally in Section 4.5 and reported empirically in Section 5.

2. Theoretical Foundations and Targeted Analysis of the Literature

2.1. Configuration as a Constraint Satisfaction and Optimization Problem

Configuration design has historically been defined not as the free generation of a new object but as the selection and coordination of elements from predefined families. In the classical formulation, components must be selected, assigned to functions, and connected through feasible relationships so that the resulting system satisfies user requirements and domain rules [2]. A review of industrial configurators shows that the most stable concepts separate domain knowledge (components, resources, and compatibility rules) from the search procedure [3]. This line of research corresponds directly to the class of problems considered here: five decisions are made jointly, and correctness is determined not by the quality of individual choices but by the consistency of the complete set.

Formally, such a problem is close to a finite constraint satisfaction problem with an objective function. Constraint-programming, SAT, mixed-integer-programming methods, and their hybrids can guarantee feasibility and, upon exact search termination, global optimality. The modern CP-SAT-LP solver combines lazy constraint generation, SAT search, linear programming, and a portfolio of strategies [10]. The strength of exact methods lies in the verifiability of the result and the availability of a certificate of search completion; their weakness is that solution time can grow rapidly as the number of variables, variants, and relationships increases.

The hierarchical imitation model HIM-D adheres to this formal tradition but organizes knowledge across the levels of domain, solution group, case, and constraint functional. In this study, the hierarchy is not used as a heuristic guarantee of scalability; rather, it serves as a schema for representing, logging, and reusing rules, while correctness is ensured by exhaustive search over a space of 1024 configurations. Consequently, the 100.00% feasibility

of HIM-D is an expected property of an exact algorithm operating in a small finite space, not empirical evidence of universal superiority.

2.2. Machine Learning for Combinatorial Optimization: Two Distinct Modes

The review by Bengio, Lodi, and Prouvost [4] draws a fundamental distinction. Machine learning can, first, improve individual decisions within a classical algorithm—such as branching, bound estimation, space reduction, or warm-start generation. Second, a model can attempt to map a problem instance directly onto a complete solution. A review of graph neural methods [5] develops this typology further and shows that relational representations are useful for problems in which the input structure is defined by a graph of variables and constraints.

The solver-boosting mode preserves the formal kernel: an error in a trained component typically degrades the speed or quality of the heuristic but does not necessarily render the final solution infeasible, provided full validation is performed before output. A direct neural-network solver operates differently—it produces a solution in one or more passes and can be very fast, but its output is a statistical prediction. Even when each component classification has high local accuracy, the joint solution may violate the global budget, the minimum quality, the risk limit, or at least one compatibility pair.

For tabular and structured data, strong baseline models include residual multilayer networks and attention architectures [11,12]. These are appropriate for testing how well a neural network can recover an optimal configuration from a fixed numerical representation. However, neither the MLP nor the attention architecture inherently encodes the condition "exactly one option from each group, with all pairs compatible." In the present experiment, pure neural-network models are deliberately evaluated without corrective post-processing. This is not an attempt to present them as the best possible neural optimization method, but a controlled test of the risk inherent in direct inference without a formal verification layer.

This motivates a methodological comparison criterion. For a direct solver, reporting average accuracy for individual groups is not sufficient; it is necessary to measure full configuration feasibility, the quality of the objective function among feasible solutions only, the proportion of exact optima, OOD degradation, and the types of simultaneous violations. These are the metrics used below.

2.3. Differentiable Optimization and Hard-Constraint Enforcement

Between the pure predictor and the external exact solver lies a broad class of integrated methods. OptNet incorporates quadratic optimization as a differentiable layer of a neural network [13]. Differentiating through a black-box combinatorial solver allows a model to be trained on final solution quality without replacing the discrete algorithm itself with a continuous approximation [14]. CombOptNet extends this idea to integer programs and allows the objective-function coefficients and constraints themselves to be learned [15]. The shared strength of these approaches is that the solution structure participates in learning rather than being considered only after prediction.

Another line of research builds feasibility into the design. Homeomorphic projection maps the neural-network output onto a feasible set for a wide class of compact sets [16]. LinSATNet introduces a differentiable, positive linear constraint-satisfaction layer and demonstrates it in routing, graph matching, and portfolio problems [17]. HardNet applies a differentiable projection for affine and convex constraints while preserving universal approximation power within the claimed class [18]. These results show that the claim "a neural network inevitably violates constraints" is too strong: constraints can be built into the architecture, but the resulting guarantee depends on their mathematical class and on the completeness of the projection layer.

For the configuration problem considered here, the constraints are heterogeneous: budget, quality, and risk are additive thresholds; the choice of one option from a group is a discrete condition; and compatibility is defined by arbitrary binary relations between groups. Positive linear layers can encode part of this structure, but an arbitrary incompatibility matrix together with the global-minimum requirement preserves the discrete combinatorial component. For this reason, the present study relies on an independent, complete verifier and an exact backup search rather than on a differentiable projection layer.

A further distinction concerns soft versus hard constraints. Adding a violation penalty to the loss function can increase average feasibility, but the resulting penalty does not guarantee feasibility of any single probabilistic output. In applications governed by mandatory rules, correctness must follow either from the design itself or from full verification prior to the issuance of the result. This distinction is central to the interpretation of the results below.

2.4. Hybrid and Neuro-Symbolic Systems with Independent Verification

Hybrid artificial intelligence combines learnable and symbolic components, but the mere presence of two module types does not by itself determine reliability. What matters is the computational contract: which module has the authority to produce the final answer, which module checks the constraints, and what happens when a violation is detected. In high-risk tasks, internally interpretable rules and verifiable decisions are preferable to post hoc explanations of black-box outputs [19]. System context also matters, since hidden dependencies on data, versions, and implicit transformations create technical debt [20].

In the hybrid architecture studied here, the neural network neither modifies the rules nor validates its own decision; its role is limited to proposing a candidate. The verifier recomputes all aggregates and checks all ten intergroup relations. If the candidate is acceptable, it may be adopted as a fast-route solution; if a violation is detected, control is transferred to the exact HIM-D solver. This scheme constitutes a procedural hybrid with conditional control transfer, rather than a jointly trained neuro-symbolic model.

It is worth stating precisely how this arrangement differs from a jointly trained neuro-symbolic model, since both are often described by the same word. In systems such as DeepProbLog [21] and SATNet [22], the symbolic component is differentiable and is trained together with the network, so that logical or combinatorial structure enters the gradient and shapes the representation; surveys of the field treat exactly this coupling as the defining property of the current neuro-symbolic wave [23]. Four consequences follow, and all four are reversed in the architecture studied here. First, in a jointly trained model the guarantee, where one exists, holds for the fragment of logic the layer can encode, and it is obtained at training time; here, the guarantee is obtained at inference time and covers whatever the verifier implements. Second, a jointly trained model must be retrained when the rules change, whereas the rule set here is a versioned artifact that can be edited without touching the network weights. Third, joint training makes the symbolic component part of the learned function, so a defect in it is difficult to localize; a procedural verifier can be tested against the rules in isolation, and Section 5.14 does exactly that by removing rules from it and measuring the effect. Fourth, joint training generally improves the network's own outputs, whereas procedural coupling leaves the network exactly as it was and changes only what is allowed to leave the system. The same asymmetry has recently been argued for language-model pipelines, where the model proposes and an external critic verifies, on the grounds that the proposing and the certifying roles have different failure modes and should not be merged [24]. The procedural hybrid is therefore the weaker form of integration, and the empirical question is how much of the reliability gap it nevertheless closes.

The guarantee here is conditional and must be stated precisely. Final feasibility reaches 100.00% only if (i) the verifier implements all mandatory rules, (ii) the input data contain no hidden errors, (iii) the backup solver finds a valid solution, and (iv) no unverified result bypasses the formal route. The log must therefore record the rule version, input parameters, candidate violations, whether the backup solver was invoked, and the final configuration. These documentation principles are consistent with established model-card practice [25].

The hybrid route creates a measurable trade-off. The higher the feasibility of the neural-network candidate, the less frequently the expensive solver is invoked. If a feasible candidate is accepted without an optimality check, speed may improve at the cost of a non-zero objective gap. It is therefore necessary to report, for any hybrid system, the proportion of backup-solver invocations together with final feasibility, joint optimality, and latency.

2.5. Large Language Models: Formulation, Solution, and Tool Use

The literature on large language models (LLMs) and optimization spans several tasks that should not be conflated. The first task is to transform a natural-language description into a mathematical model and solver code. OptiMUS uses a modular LLM agent to formulate mixed-integer problems, generate code, debug it, and verify the solution with an external solver [9]. OptiBench evaluates the full path from a text description to a numerical answer by invoking a software solver [8]. In these systems, the LLM is valuable as an interface and formalizer, while the guarantee of final optimality comes from the external tool.

The second task is direct solution without a formal tool. LR^{2}Bench demonstrates the difficulty of long-chain reflective reasoning on constraint satisfaction problems [7]. ConstraintBench separates formulation from direct search: the model receives a fully specified problem, returns a structured solution, and an independent deterministic verifier checks each condition and compares the objective with the optimum [6]. In this mode, the binding constraint proved to be feasibility itself, rather than objective quality among already-feasible answers.

These results support the functional separation adopted in the present paper. A language or neural-network model can interpret requirements, rank options, and generate a candidate; however, when mandatory discrete rules are present, the final decision must pass through a verifier- or solver-backed route. External LLM metrics are not treated further as direct experimental competitors to HIM-D, since the problem sets, models, search budgets, and computational environments differ.

2.6. Targeted Literature Synthesis and Research Gap

A targeted analysis of the sources reveals not a single linear scale of methods but eight distinct architectural modes. These differ in where the formal structure resides, what is being learned, and at what stage the guarantee arises. Table 2 links these modes to the design of the present experiment.

As the seventh row of Table 2 indicates, the procedural hybrid is the specific architectural mode this paper tests; the other seven rows situate it relative to the alternatives already covered by the literature. The reviewed body of work remains insufficiently focused on how reliability changes when moving from a direct neural-network candidate to independent verification with a conditional exact backup, evaluated on the same set of configuration problems. Work on differentiable optimization offers closer integration, while LLM benchmarks evaluate a different type of input. This study addresses a narrower but operationally clear gap: on a fixed 5×4 framework, it quantitatively compares di-

rect inference, an exact solution, and a procedural hybrid in terms of full feasibility, joint optimality, OOD stability, repeatability, and latency.

Table 2. Targeted synthesis of research directions and their relationship to the present work.

Direction	Typical Mechanism	Strong Side	Limitation for the Tasks Considered	Role in the Experiment
Configuration, CSP, CP/MILP [2,3,10]	Explicit variables, constraints, and exact search	Verifiable feasibility and optimality	Time grows with the expansion of the search space	HIM-D as the exact formal route
ML for solver enhancement [4,5]	Warm start, branch selection, search pruning	Preserves the formal core and can accelerate search	Requires integration with a specific solver	Justifies the role of the neural network as an auxiliary module
Direct neural-network solver [11,12]	Mapping the input directly to a configuration	Very low latency	No built-in guarantee of global compatibility	Residual MLP and group-attention network
Differentiable optimization [13–18]	Optimization layer, black-box solver, or projection	Constraints are involved in training or provided by design	Guarantee depends on the constraint class and the layer's implementation	Alternative route discussed as the next baseline
LLM with tools [8,9]	Text formalization, code generation, external solver	Flexible interface	Formulation errors and tool dependence	Context for separating interpretation from decision-making
Direct LLM solution [6,7]	Generating a ready-made solution without a solver	Universal natural-language interface	Feasibility remains narrow	External benchmark, not pooled with internal statistics
Procedural hybrid	Candidate generation, independent verification, conditional exact backup	Final feasibility across the full verification route	Savings depend on candidate quality; acceptability differs from optimality	Main verifiable architectural hypothesis of this study
Jointly trained neuro-symbolic models [21–23]	Differentiable logic or combinatorial layer trained end to end with the network	Symbolic structure enters the gradient and shapes the learned representation	The guarantee covers only the fragment the layer can encode, and the model must be retrained when the rules change	Contrast case: the architecture studied here shares neither parameters nor gradients

This formulation also bounds the scientific claims of the study. The work does not claim that brute-force search is a scalable solution, and it does not survey the whole space of constraint-aware architectures. Its contribution is to measure reproducibly the architectural effect of mandatory constraint checking, and to place that effect against three reference points on the same corpus: constraint-aware correction of the neural candidate without an exact solver (Section 5.11), general-purpose exact solvers in the form of CP-SAT and mixed-integer programming (Section 5.12), and the growth of exact-search cost as the configuration space expands (Section 5.13).

2.7. Implications of the Review for the Experimental Protocol

Five experimental requirements emerge from the literature review. First, the unit of analysis must be a complete configuration, not a single component. Second, feasibility

must be verified independently of the source data used to train the models. Third, objective quality must be assessed separately from validity. Fourth, the ID and OOD sets must be separated in advance. Fifth, the hybrid route and the proportion of backup-solver invocations must be reproducible. These requirements shape the formal model, the corpus description, and the metrics defined in the following sections.

The neural-network results reported below should be interpreted as a baseline for direct forward inference, not as an upper bound for modern neural optimization more broadly. This caveat matters for the correct positioning of the article and for preventing unwarranted generalization from two compact networks to the entire class of learning-based methods.

3. Formal Model and Research Hypotheses

3.1. Representation of the Configuration Task

Let u denote a requirement vector and $x = (x_1, \ldots, x_G)$ a configuration, in which each variable x_g selects exactly one of the O options available in group g. In the experiment, $G = 5$ and $O = 4$, so the configuration space contains $O^G = 1024$ candidates. Let $z_{g,o} \in \{0, 1\}$ be the binary indicator that option o is chosen in group g, so that $x_g = o$ if and only if $z_{g,o} = 1$. For a configuration x, the aggregate cost $C(x)$, quality $Q(x)$, risk $R(x)$, and preference-mismatch penalty $P(x)$ are the sums of the corresponding option attributes over the G groups. Compatibility is defined pairwise rather than globally: for each unordered pair of distinct groups (g, h), the instance supplies a binary matrix $K^{(g,h)} \in \{0, 1\}^{O \times O}$, with ${K^{(g,h)}}_{o,o'} = {K^{(h,g)}}_{o',o}$, whose entry equals one when option o of group g may be combined with option o' of group h. A configuration is feasible when it satisfies the following conditions:

$$\sum_{o=1}^{O} z_{g,o} = 1 \ \forall g; \quad C(x) \le B; \quad Q(x) \ge Q_{\min}; \quad R(x) \le R_{\max}; \quad K^{(g,h)}(x_g, x_h) = 1 \ \forall \ 1 \le g < h \le G \tag{1}$$

Three points of notation require comment. The summation index runs over the O options of a group, O being the quantity defined above; the risk term is the function $R(x)$ and not a subscripted symbol; and the compatibility condition is a conjunction of $G(G-1)/2 = 10$ scalar conditions, one for each unordered pair of groups, rather than a single equality imposed on a matrix. Writing it as $K(x) = 1$ would conflate the family of pairwise matrices with its evaluation at the chosen options.

The objective function combines cost, preference discrepancy, risk, and quality:

$$J(x) = C(x) + 18P(x) + 0.6R(x) - 0.15Q(x) \to min \tag{2}$$

The three weights were fixed before any test sample was examined. They were not elicited from practitioners and do not correspond to an industry standard; they were chosen so that, at the scale of the attribute ranges produced by the generator, the preference, risk, and quality terms, each makes a non-negligible contribution alongside cost, and so the resulting trade-off is genuine rather than dominated by a single criterion. The same applies to the budget, quality, and risk thresholds of Section 4.2.3, which are defined relative to a reference configuration drawn separately for each instance rather than against any external benchmark. Section 5.16 reports how far the identity of the optimum and the behavior of the hybrid move when each weight is halved and doubled.

The HIM-D implementation searches the finite space of 1024 configurations, discards every candidate that violates at least one hard constraint, and returns the feasible configuration with the smallest $J(x)$. As already noted in Section 2.1, its full feasibility and its agreement with the global minimum follow from the definition of an exhaustive algorithm and are therefore treated throughout this paper as a reference bound rather than as a

result; to avoid repeating the point at each occurrence, Section 5.13 states once how far that bound extends before exhaustive enumeration ceases to be practicable. The substantive experimental question concerns the direct neural candidates and the effect of independent verification with conditional fallback.

HIM-D should be interpreted as a task-family-specific implementation rather than as a universal configuration engine. Its variables, feasibility conditions, compatibility rules, and objective function are fixed for the configuration family examined in this experiment. A different class of application tasks would require a separately constructed and validated deterministic module, although the higher-level procedural architecture, that is, candidate generation, independent verification, and conditional fallback, could remain unchanged.

3.2. Hypotheses

The statements below are not all of the same epistemic kind, and the distinction governs how they should be read. H1, H2, and H4 concern quantities whose value follows from the design of the experiment rather than from an uncertain empirical mechanism. HIM-D is defined as an exhaustive exact search over all 1024 configurations; every generated instance is verified in advance to admit at least one feasible solution; and the hybrid returns the HIM-D solution whenever the neural candidate fails verification. Given a correct implementation, HIM-D and the hybrid must therefore be feasible on every instance and their decline from ID to OOD conditions must be zero. These three statements are retained because they are the operational checks that the solver, the generator, and the control-transfer rule behave as specified, and they are reported below as structural checks rather than as findings; a failure of any of them would indicate a defect in the implementation, not a property of the architecture. The empirical content of the study lies in the remaining statements: in how far the direct neural candidates fall short of feasibility and how that shortfall responds to distribution shift, in what fraction of the accepted fast-route candidates is also optimal, in whether repeatability survives once the inference rule is held constant across systems, and in whether constraint-aware correction of the same candidate can substitute for the formal route.

The decision thresholds were chosen as conservative, practically meaningful boundaries rather than post hoc choices, and they are settings of this experiment rather than established norms. The 5-percentage-point margin for H1 and the 1-percentage-point margin for H4 are both clearly larger than the width of the Wilson 95% confidence intervals for $n = 6000$ (Section 4.6), which allows a meaningful effect to be distinguished from random variation; the non-inferiority framing follows established equivalence-testing methodology [26]. The latency threshold of under 1 ms per task in H5 reflects the upper bound acceptable for interactive use in the computing environment of this experiment, and Section 5.13 shows how quickly that bound ceases to hold as the configuration space grows.

- H1 (structural check). The feasibility of HIM-D exceeds the feasibility of each pure neural-network model by at least 5 percentage points.
- H2 (structural check). The decline in feasibility when moving from ID to OOD conditions is smaller for HIM-D than for each pure neural-network model.
- H3 (empirical). Under a common deterministic inference rule, HIM-D, the hybrid, and both neural networks are exactly repeatable across five runs; exact repeatability is lost only when the network outputs are sampled from their softmax distributions instead of being decoded by argmax.
- H4 (structural check). The hybrid system is not inferior to HIM-D in feasibility, within a 1-percentage-point margin.

- H5 (empirical). The hybrid system improves joint feasibility and optimality relative to the pure attention network while maintaining sub-1 ms latency per task in the computing environment described in Section 4.4.
- H6 (empirical). Constraint-aware correction of the same neural candidate, applied without an exact solver, raises feasibility relative to uncorrected argmax decoding but does not attain the guarantee obtained by independent verification with exact fallback.

4. Materials and Methods

4.1. Research Design and Corpus Formation

The study was conducted as a reproducible computational experiment with a pre-fixed generator and an unchangeable evaluation protocol. The corpus contains no personally identifiable information and is not linked to real users, training platforms, production catalogs, or commercial transactions. A total of 24,000 independent numerical instances of a single configuration task were generated: 15,000 instances were used for training, 3000 for tuning and model-state selection, and 6000 for final evaluation. The test corpus was pre-partitioned into 3000 tasks within the training distribution (ID) and 3000 tasks under a controlled distribution shift (OOD). Numerical instances were distributed equally across the three domain parameterizations and the twelve semantic scenarios described below. Table 3 reports the resulting corpus composition, cross-tabulating each split (training, validation, ID test, and OOD test) against the three domains; it makes explicit how the 24,000 instances decompose into the 15,000/3000/3000/3000 split referenced throughout Section 4 and the 6000-instance test set analyzed in Section 5.

Table 3. Composition of the corpus by split and domain (number of task instances).

Corpus Part	Education	Engineering	Commerce	Total	Purpose
Training	5000	5000	5000	15,000	Fitting neural-network weights
Validation	1000	1000	1000	3000	Quality control and model-state selection
ID test	1000	1000	1000	3000	In-distribution evaluation
OOD test	1000	1000	1000	3000	Evaluation under new ranges and dependencies
Total	8000	8000	8000	24,000	Full computational corpus

Note: rows sum to the totals reported in Section 4.1 (15,000 training + 3000 validation + 3000 ID test + 3000 OOD test = 24,000).

As Table 3 shows, the three domains contribute identical numbers of instances to every split; this balance is what allows the domain-level results in Section 5.4 to be averaged directly into the pooled figures of Section 5.2 without any weighting correction.

4.2. Task Generator

Each numerical instance comprised five component groups, each requiring the choice of one of four alternatives. Every alternative was described by continuous indicators of cost, quality, risk, and a penalty for non-compliance with preferences, and a symmetric compatibility matrix was formed for options belonging to different groups. To guarantee the feasibility of each instance, the generator first created a reference feasible configuration, preserved the compatibility of its pairs, and then set budget, quality, and risk thresholds relative to the reference sums. After generation, all 1024 possible configurations were checked with an exhaustive benchmark algorithm; instances for which the existence of a feasible solution could not be confirmed were automatically regenerated.

Three details of the generator matter for reproduction. The reference configuration is drawn uniformly over the 1024 candidates, and the three thresholds are then set as

fixed multiples of its own cost, quality, and risk sums, with the multipliers drawn from the intervals listed in Section 4.2.3; feasibility of at least one configuration is therefore guaranteed by construction rather than by rejection sampling, and the exhaustive post-check reported above never had to discard an instance in the corpus actually used. The compatibility matrices are drawn independently for each unordered pair of groups at the instance-specific incompatibility density, after which the pairs belonging to the reference configuration are forced to be compatible; the density reported in Section 4.2.3 is therefore the density before this correction. All attribute values are continuous and are normalized by constants fixed in advance and identical for the ID and OOD partitions, so that a distribution shift remains visible to the model instead of being absorbed by the scaling. The complete procedure is set out step by step in Appendix C, and the corresponding source file is part of the reproducibility package described in Section 4.7.

The task generator used to construct the experimental corpus must not be confused with the domain-specific configurational module represented by HIM-D. The former produces numerical problem instances for training and evaluation; the latter operates on a given instance, verifies its constraints, and identifies a feasible or optimal configuration. This terminological distinction is maintained throughout the remainder of the study.

The three domain parameterizations differed in attribute ranges and in the relative importance of criteria: in the educational parameterization, material quality received increased weight; in the engineering parameterization, cost and risk did; and in the commercial parameterization, price and preference compliance did. The OOD component was formed through three simultaneous modifications: increasing the incompatibility density from 5–13% to 16–28%, narrowing the permissible threshold intervals, and shifting the cost and risk ranges. The OOD problems therefore retained proven solvability but had a smaller feasible domain. It should be emphasized that the subject-area names do not represent real data; they define the interpretation of groups and criteria, while all statistical results refer to synthetic numerical instances.

4.2.1. Unit of Analysis and a Representative Test Task

The unit of analysis is a single, fully specified configuration instance. It is not a textual question and does not require the model to engage in open-ended dialogue; rather, it is a machine-readable task requiring the assembly of one solution from five interdependent choices. This formulation allows the ability to satisfy constraints to be separated from the ability to understand natural language.

1. *Input.* Twenty alternatives organized into five groups of four options. For each alternative, cost, quality, risk, and a preference-mismatch penalty are specified. In addition, three global thresholds, a problem domain, and binary compatibility criteria for all ten pairs of groups are specified.
2. *Solution.* A vector $x = (x_1, \ldots, x_5)$, where x_g indicates the option selected in group g; the complete answer therefore contains exactly five indices.
3. *Hard constraints.* Exactly one option is selected from each group; total cost does not exceed the budget; total quality is not below the minimum; total risk does not exceed the limit; and each of the ten intergroup pairs is compatible.
4. *Objective.* Among feasible configurations, $J(x)$ is minimized, combining cost, preference penalty, risk, and quality with pre-fixed coefficients.
5. *Benchmark.* All 1024 combinations are checked by exhaustive search. For each problem, the existence of a feasible solution, the minimum value J^*, and the optimal configuration are recorded.

6. *Success criteria.* Primary success equals one if and only if all hard constraints are satisfied. Secondary success additionally requires a relative objective gap no greater than 0.1%.

The experiment therefore tests not knowledge of a specific course, device, or product, but the method's ability to generate a globally consistent combination within a limited feasible domain. The subject-specific scenarios define the meaning of the variables and allow the architecture's robustness to be assessed across three numerical parameterizations.

4.2.2. Compact Description of the Twelve Subject Scenarios

Each of the three parameterizations is associated with four scenarios. Each scenario contains 2000 instances: 1250 training, 250 validation, 250 ID test, and 250 OOD test. The mathematical structure is identical across scenarios, but the substantive interpretation of the five groups and the meaning of the constraints differ. Table 4 allows the reader to see at a glance what exactly is being configured in each case.

Table 4. Compact description of the twelve tested configuration scenarios.

Code	Task Under Test	Five Joint Decisions	Domain-Specific Meaning of the Hard Constraints
E1	Personalized calculus lesson	Theory; explanation method; worked example; practice; assessment	Time/cost constraints, minimum pedagogical value, cognitive risk, consistency of lesson elements
E2	Mathematics test set	Topic; difficulty; presentation; response type; feedback	Volume, coverage, acceptable complexity, format compatibility and verifiability
E3	STEM laboratory exercise	Process; equipment; measurement; processing; report	Resource limit, data quality, procedural risk, equipment-and-method compatibility
E4	Programming training module	Concept; language/environment; demonstration; practice; test	Time, minimum learning outcome, complexity/risk, compatibility of environment, tasks, and assessments
I1	Sensor-node measurement unit	Sensor; converter; controller; communication; power supply	Price, accuracy, technical risk, electrical and interface compatibility
I2	Industrial monitoring system	Measurement; data collection; computer; network; protective design	Budget, reliability, failure risk, protocol and operational compatibility
I3	Thermal management system	Heat exchanger; pump/fan; sensor; controller; housing	Cost, thermal efficiency, risk, range and component-interface compatibility
I4	Automated production cell	Actuator; controller; sensor; safety system; operator interface	Resources, performance, safety, control-and-protection compatibility
C1	Computer equipment bundle	Processor; memory; storage; display; service	Price, performance, incompatibility risk, hardware-and-service consistency
C2	Telecommunications plan	Data; voice; roaming; device; support	Monthly budget, utility, overpayment risk, tariff-and-technical compatibility
C3	SaaS package	Tariff; users; storage; security; support	Cost, functional value, risk, licensing, security, and service alignment
C4	Office equipment set	Printing; scanning; consumables; warranty; installation	Overall price, quality, operational risk, equipment-and-service compatibility

Note: The twelve scenarios are semantic interpretations of one common numerical structure and contain no real business, catalog, or curriculum data; no scenario is drawn from a commercial product line, a course syllabus, or an engineering bill of materials. The input presented to every system is a machine-readable numerical vector of attributes, thresholds, and binary compatibility relations, not a natural-language description, so the task is not comparable to the natural-language optimization benchmarks discussed in Section 2.5.

Three examples illustrate how to read the table. In E1, a combination in which the explanation method, worked example, and assessment are pitched at different expertise

levels, or one that exceeds the available time, is unacceptable. In I1, a candidate may satisfy the price and quality criteria yet be rejected because the controller, communication channel, and power supply are incompatible. In C3, a formally attractive plan is rejected if the number of users, security mode, and support tier do not form a valid package.

The scenarios are a semantic shell over a common numerical structure. The generator modifies distributions at the level of the three domains but does not draw on real catalogs or simulate a separate empirical distribution for each scenario. The results therefore pertain to the architectural validation of configuration solutions rather than to demonstrating effectiveness for specific lessons, devices, or products.

4.2.3. Controlled Complexity and Formation of the OOD Shift

The OOD partition was formed by simultaneously changing three factors: increasing the density of incompatibility, narrowing the permissible threshold intervals, and shifting the cost and risk ranges. Each OOD problem retained proven solvability, but the proportion of feasible configurations decreased. This design tests robustness to a more stringent combination of already-known conditions, rather than the emergence of entirely new variable types.

Table 5 lists the seven generation factors that jointly define task difficulty and shows, factor by factor, how each one moves between the ID and OOD levels; the last column states why each factor was included in the experimental design.

Table 5. Generation factors used to control task complexity and to form the OOD shift.

Factor	ID Level	OOD Level	Role in the Experiment
Number of groups	5	5	Size of the hierarchy
Options per group	4	4	1024 configurations per task
Incompatibility density	5–13%	16–28%	Complexity of interaction constraints
Budget threshold	102–114% of the reference cost	100–103.5% of the reference cost	Resource stringency
Minimum quality	87–99% of the reference quality	96.5–100% of the reference quality	Quality stringency
Maximum risk	102–114% of the reference risk	100–103.5% of the reference risk	Safety stringency
Attribute ranges	Training ranges	Shifted cost and risk ranges	Out-of-distribution generalization

Together, the three tightened thresholds and the higher incompatibility density in Table 5 explain why OOD feasibility is lower than ID feasibility for the two pure neural-network models evaluated in Section 5.3, the formal route being unaffected, even though the underlying task family and the number of groups and options per group are unchanged.

4.3. Compared Systems

All systems received an identical machine-readable description of the problem. Pure neural-network models generated five probability distributions and independently selected one option per group by argmax; they received no feedback from the verifier and were evaluated without corrective post-processing, so they represent a baseline of forward

inference rather than the full spectrum of constraint-aware neural optimization. HIM-D performed an exact search and served as the formal reference. The hybrid system used the same attention network but added two untrainable components: an independent verifier and a backup HIM-D solver. Hybridity was implemented at the level of the computational flow rather than through parameter sharing or joint training. The comparison also includes five further systems, described where they are first used: three constraint-aware variants of the same attention network that operate without an exact solver, and two general-purpose exact solvers, CP-SAT [10] and mixed-integer programming [27], which serve as independent confirmations that the HIM-D implementation returns the true optimum, CP-SAT additionally serving as the backend against which the scaling behavior of the exact route is measured.

In the present experiment, the deterministic module is specified directly by the researchers. In a broader development workflow, its initial formal specification or executable implementation may be produced with the assistance of an AI model. However, before operational deployment, the resulting module must undergo independent domain validation, boundary-case testing, and version fixation. During operation, it remains functionally separate from the probabilistic model whose candidates it evaluates.

The neural-network candidate is denoted $x_N = N\theta(u)$; the independent test is $V(u, x_N) \in \{0, 1\}$; and the exact deterministic solver is $S(u) = \text{argmin } J(x)$ over the set of all feasible configurations. The final hybrid operator is defined by the following rule:

$$H(u) = x_N \text{ if } V(u,x_N) = 1; \quad H(u) = S(u) \text{ if } V(u,x_N) = 0 \tag{3}$$

The algorithm proceeds in six steps: (1) the attention network receives the full vector of attributes, thresholds, and compatibility relations; (2) a candidate is formed from five options by maximum likelihood; (3) the verifier recomputes cost, quality, risk, and all ten pairwise compatibility relations without using the neural network's probabilities; (4) a feasible candidate is accepted without re-optimization, so it can be feasible without being globally optimal; (5) if at least one condition is violated, a full enumeration of the 1024 configurations is launched and the feasible configuration with minimal $J(x)$ is selected; and (6) the final result is rechecked and logged together with the route taken, the list of violations (if any), and the rule version. This conditional switch between a probabilistic proposal and a formal solution constitutes the hybridity of the system studied here.

Table 6 summarizes the nine systems compared in Section 5 side by side—their internal mechanisms, the number of trainable parameters, the scope of the examination each output undergoes before being returned, and each system's role in testing the study's hypotheses.

Table 6. Comparison of the systems evaluated in this study.

System	Mechanism	Trainable Parameters	Examination Restrictions	Role
HIM-D	Explicit hierarchy; full search across 1024 configurations	0	Built-in, complete	Exact formal solver and reference upper bound
Residual MLP	Residual fully connected network; joint inference of 5 groups	471,572	None	Neural-network baseline model
Group-attention network	One token per group; single attention block; shared context	10,596	None	Compact structural neural network

Table 6. *Cont.*

System	Mechanism	Trainable Parameters	Examination Restrictions	Role
Attention network + verifier + HIM-D	$N\theta$ proposes x_N; V independently recomputes constraints; if $V = 0$, full enumeration via HIM-D	10,596 trainable; formal modules untrained	Full, up to issuance of the result	Procedural hybrid with mandatory verification and conditional backup
Attention network + soft-constraint penalty in the loss	Expected number of violated compatibility pairs added to the cross-entropy loss	10,596 trainable	None	Constraint-aware training baseline (Section 5.11)
Attention network + greedy feasibility repair	Iterative reassignment of the group whose change most reduces the number of violated constraints	10,596 trainable; repair rule untrained	Partial; no exact search	Constraint-aware correction baseline (Section 5.11)
Attention network + constraint-aware beam decoding	The 32 highest-probability configurations are screened by the verifier	10,596 trainable	Partial; limited to the beam	Constraint-aware decoding baseline (Section 5.11)
CP-SAT (OR-Tools)	Lazy clause generation, SAT search, and linear relaxation [10]	0	Built-in, complete	General-purpose exact solver (Sections 5.12 and 5.13)
MILP (CBC)	Branch and bound on a linear integer model [27]	0	Built-in, complete	General-purpose exact solver (Section 5.12)

The parameter counts in Table 6 (471,572 for the Residual MLP versus 10,596 for the attention network) recur in Section 5.6, where they are related to the measured latency of each system; the "examination restrictions" column anticipates the central empirical contrast of Section 5, between systems whose output is never checked and the hybrid, whose output is always checked before issuance.

4.4. Feature Preparation and Training

The input to each task was represented as five group tokens of 87 features each. The first 16 features are the cost, quality, risk, and preference-mismatch attributes of the four options belonging to the token's own group, each divided by a normalization constant fixed in advance. The next 64 are the binary compatibility entries between the four options of that group and the four options of each of the other four groups. The remaining seven are the three normalized global thresholds, a three-component one-hot indicator of the domain parameterization, and the incompatibility density of the instance. The Residual MLP received the concatenation of the five tokens as a flat vector of 435 features, projected it linearly onto 384 units with layer normalization, applied one residual block of two 384-unit layers with dropout of 0.08, and produced 5×4 logits through a linear head, giving 471,572 trainable parameters. The group-attention network projected each of the five tokens into a 32-dimensional space, applied one self-attention block over the five tokens followed by a position-wise feedforward layer of width 64 under residual connections, and produced four logits per token, giving 10,596 trainable parameters. Neither architecture contains a layer that encodes the requirement of exactly one option per group with all pairs compatible; the one-of-four structure is imposed only by the per-group softmax, and nothing in either model represents the ten intergroup relations as constraints rather than as input features.

Both networks were trained to reproduce the exact optimal configuration returned by HIM-D, with the mean of five per-group cross-entropy terms as the loss. No constraint term appears in this loss; the soft-constraint variant evaluated in Section 5.11 adds one. Optimization used AdamW with L2 regularization of 10^{-4}, implemented in PyTorch [28]. The Residual MLP used a learning rate of 10^{-3}, a batch size of 512, and five epochs; the group-attention network used a learning rate of 8×10^{-4}, a batch size of 512, and three epochs. The number of epochs was fixed in advance, and the validation split was used only to confirm that the loss had stopped improving, not to select among architectures or to tune hyperparameters against the test data. All pseudorandom processes, including corpus generation, weight initialization, and batch ordering, were controlled by the seed 20260711; Section 5.15 reports the variability that remains once this seed is varied. All computations, including the latency measurements of Section 5.6, were carried out in a single process on the central processing unit of a system with an Intel Core i7-1355U processor (10 physical cores, 1.7 GHz base clock) and 16 GB of RAM, running Windows 11 Pro (version 10.0.26200), with Python 3.12.10 and PyTorch 2.10.0+cpu; no graphics accelerator was used at any stage.

4.5. Metrics

The primary endpoint was full feasibility, equal to one only if the budget, quality, and risk constraints, the selection of exactly one option per group, and all pairwise compatibility conditions were simultaneously satisfied. A secondary composite metric additionally required that the relative objective-function gap not exceed 0.1%. Supplementary measures included exact agreement with the globally optimal configuration, the median and mean gap among feasible solutions, objective-function quality, the distribution of violation types, repeatability, and average computational latency.

Repeatability was assessed under two inference rules, so that the comparison between systems is not confounded by the decoding procedure. Under the deterministic rule, every system was run five times on the same subset of 500 test tasks with fixed inputs and settings, the two networks decoding their outputs by argmax; exact repeatability required all five returned configurations to coincide. Under the stochastic rule, five configurations were instead drawn from the softmax output distribution of each network on the same tasks. The deterministic rule is the one used for every other result reported in this paper; the stochastic rule is reported separately because it characterizes what happens when a probabilistic recommender is deployed with sampling enabled. Latency was measured as the mean CPU time per task on the machine described in Section 4.4, excluding input/output and network calls, and is therefore a property of this configuration space and this machine rather than of a deployed configurator; Section 5.13 reports how it grows with the size of the space.

Table 7 collects the operational definition of every indicator introduced above, together with the interval estimate or statistical criterion attached to it; it functions as a reference key for the result tables in Section 5, each of which reports one or more of these indicators.

Table 7. Operational definitions of the metrics and the statistical analysis plan.

Indicator	Definition	Interval Estimate or Criterion
Feasibility	All hard constraints satisfied	95% Wilson interval
Feasibility and optimality	Feasible and relative gap $\leq 0.1\%$	95% Wilson interval
Objective gap	$(J - J^*)/\lvert J^* \rvert \times 100\%$	Median and mean among feasible solutions

Table 7. *Cont.*

Indicator	Definition	Interval Estimate or Criterion
Exact configuration match	Coincidence of all five options with the optimum	Share of tasks
OOD degradation	ID feasibility minus OOD feasibility	Percentage points
Paired difference	HIM-D minus the compared system	Paired bootstrap, 2000 resamples
Repeatability	Identical result across all five runs	Share of tasks
Latency	Average CPU time per task	Milliseconds

4.6. Statistical Plan

Wilson 95% confidence intervals [29] were calculated for all proportions. Differences in feasibility were estimated by paired bootstrap resampling [30] on the same test instances, with 2000 resamples, which accounts for heterogeneity in task difficulty. Hypothesis H1 was considered supported if the lower bound of the 95% confidence interval for the HIM-D advantage exceeded 5 percentage points. For H4, a non-inferiority margin of 1 percentage point was used, following standard equivalence-testing practice [26]. OOD robustness was assessed from the difference in the proportions of feasible solutions between the ID and OOD subsamples. Because each hypothesis carries its own decision threshold and targets a logically distinct claim, no correction for multiple comparisons (for example, the Benjamini–Hochberg procedure [31]) was applied; all hypotheses, metrics, and decision rules were specified in advance and recorded in the analysis script before the final test set was examined. The study was not registered on a public platform, and the term pre-specified is used throughout in place of pre-registered for that reason.

4.7. Reproducibility and Data Provenance

All internal results were obtained by running the included code on a fixed synthetic corpus, consistent with current recommendations for reproducible machine learning research [32]. The reproducibility package described below includes the numerical-instance generator, model definitions, the exact hybrid-route protocol, a catalog of the twelve domain scenarios, the fixed random seed, the resulting JSON results file, and the figures. External metrics for large language models were taken from ConstraintBench [6] and are presented separately; they contextualize the formal feasibility problem but are not treated as a direct comparison, since they were obtained on a different problem set and under different computational conditions.

The complete reproducibility package, comprising the instance generator, the model definitions and training scripts, the verifier, the exact solver, the hybrid-route protocol, the catalog of the twelve domain scenarios, the fixed random seed, the analysis code that produces every table and figure in Section 5 and the appendices, and the resulting JSON results file, is available from the corresponding author on reasonable request, as stated in the Data Availability Statement. Regenerating the corpus and reproducing the reported values requires no accelerator and completes in minutes on a standard desktop processor.

Figure 1 illustrates the two solution paths. In the baseline mode (Panel A), HIM-D immediately performs an exhaustive search and returns a configuration whose feasibility and optimality are confirmed by the final check $V(u, x^*)$. In the procedural hybrid (Panel B), the attention network first proposes a candidate; an independent verifier $V(u, x_N)$

recomputes all constraints, and control passes to HIM-D for an exact backup search only if a violation is detected.

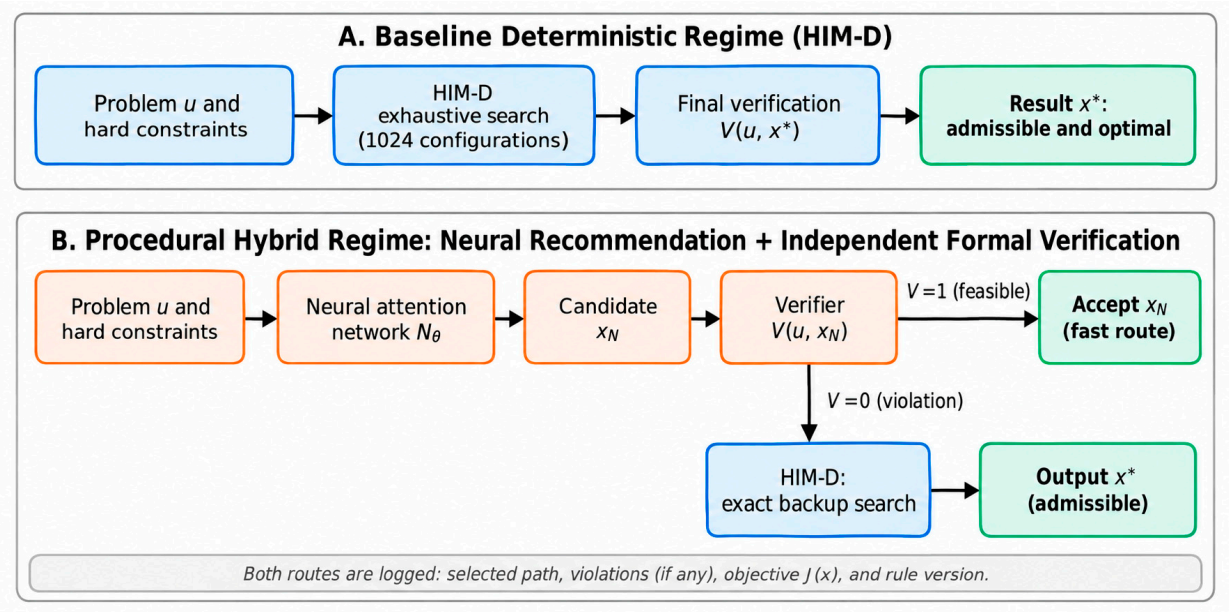


Figure 1. Deterministic and hybrid solution regimes. Panel (**A**): the baseline HIM-D regime performs an exhaustive search over the 1024-configuration space and returns a result confirmed by a final verification step, $V(u, x^*)$. Panel (**B**): the procedural hybrid regime, in which the neural attention network $N\theta$ proposes a candidate, an independent verifier evaluates $V(u, x_N)$, and control passes to HIM-D for an exact backup search only when the candidate violates at least one hard constraint ($V = 0$); otherwise, the candidate is accepted directly. Both routes are logged with the selected path, any violations, the objective value $J(x)$, and the applicable rule version.

5. Results

5.1. Experimental Workflow and Integrity Control

All 24,000 problems were successfully generated and passed a preliminary check confirming the existence of a feasible solution. There were no missing observations, no post hoc exclusions, and no changes to the test set after results were reviewed. All 6000 test problems were included in the final analysis, with each subject domain represented by 2000 problems (1000 ID and 1000 OOD).

Table 8 audits this workflow by comparing the planned and actual counts for every corpus element; the absence of any deviation confirms that the pre-specified protocol described in Section 4.1 was followed exactly, so the results reported below are not affected by post hoc sample changes.

Table 8. Task flow and adherence to the pre-specified protocol.

Element	Planned	Actual	Deviation
Training tasks	15,000	15,000	None
Validation tasks	3000	3000	None
ID test	3000	3000	None
OOD test	3000	3000	None
Full test set	6000	6000	None
Stochastic repetitions	5 × 500 tasks	5 × 500 tasks	None
Random seed	20260711	20260711	None

With the protocol in Table 8 confirmed to have been followed without deviation, the remainder of Section 5 reports the substantive comparison among HIM-D, the two pure neural-network models, and the hybrid system.

5.2. Main Results

Because HIM-D in this setting is defined as an exhaustive search over 1024 configurations, its 100.00% feasibility and optimality are an expected reference bound rather than an independent comparative finding. The main empirical result concerns the direct neural-network outputs and the hybrid route. The Residual MLP produced feasible configurations in 1044 of 6000 cases (17.40%), and the group-attention network in 1147 cases (19.12%). In the hybrid system, 1147 candidates were accepted via the fast route, while for the remaining 4853 problems (80.88%) control was transferred to the exact HIM-D solver. The hybrid's final feasibility was 100.00%, and its composite feasibility and optimality reached 83.08%, since the backup solver always returned the global minimum while a portion of the feasible neural-network candidates retained a gap greater than 0.1%. Formal verification thus eliminated all infeasible outputs but did not, by itself, guarantee the optimality of a candidate accepted without further search. These figures, together with the associated 95% confidence intervals, the exact match rate, and the objective-quality gap, are reported in full in Table 9.

Table 9. Main results on the final test corpus (n = 6000).

System	Feasibility, % [95% CI]	Exact Configuration, %	Feasibility and Optimality, % [95% CI]	Median Gap, %	Objective Quality, %
HIM-D	100.00 [99.94; 100.00]	100.00	100.00 [99.94; 100.00]	0.00	100.00
Residual MLP	17.40 [16.46; 18.38]	1.20	1.20 [0.95; 1.51]	21.55	81.54
Group-attention network	19.12 [18.14; 20.13]	2.17	2.20 [1.86; 2.60]	15.51	85.48
Attention network + deterministic verifier (hybrid)	100.00 [99.94; 100.00]	83.05	83.08 [82.11; 84.01]	0.00	97.22

CI = confidence interval; "objective quality" is the mean quality of J(x) among feasible solutions, expressed relative to the optimum.

The paired advantage of HIM-D over the Residual MLP in feasibility was 82.60 percentage points (95% CI [81.52; 83.53]); the advantage over the attention network was 80.88 percentage points (95% CI [79.97; 81.93]). For the hybrid system, the difference in feasibility relative to HIM-D was 0.00 points, satisfying the non-inferiority criterion within the pre-specified 1-point margin.

Figure 2 compares two indicators for each system on the full test corpus (n = 6000): the proportion of feasible configurations and the proportion that is both feasible and within 0.1% of the optimal objective value. HIM-D reaches 100.00% on both. The hybrid also reaches 100.00% feasibility, but 83.08% on the joint criterion, because a candidate that passes verification is accepted without re-optimization and can therefore be feasible without being optimal. The Residual MLP and the attention network reach 17.40% and 19.12% feasibility, and the shares additionally satisfying the optimality criterion fall to 1.20% and 2.20%.

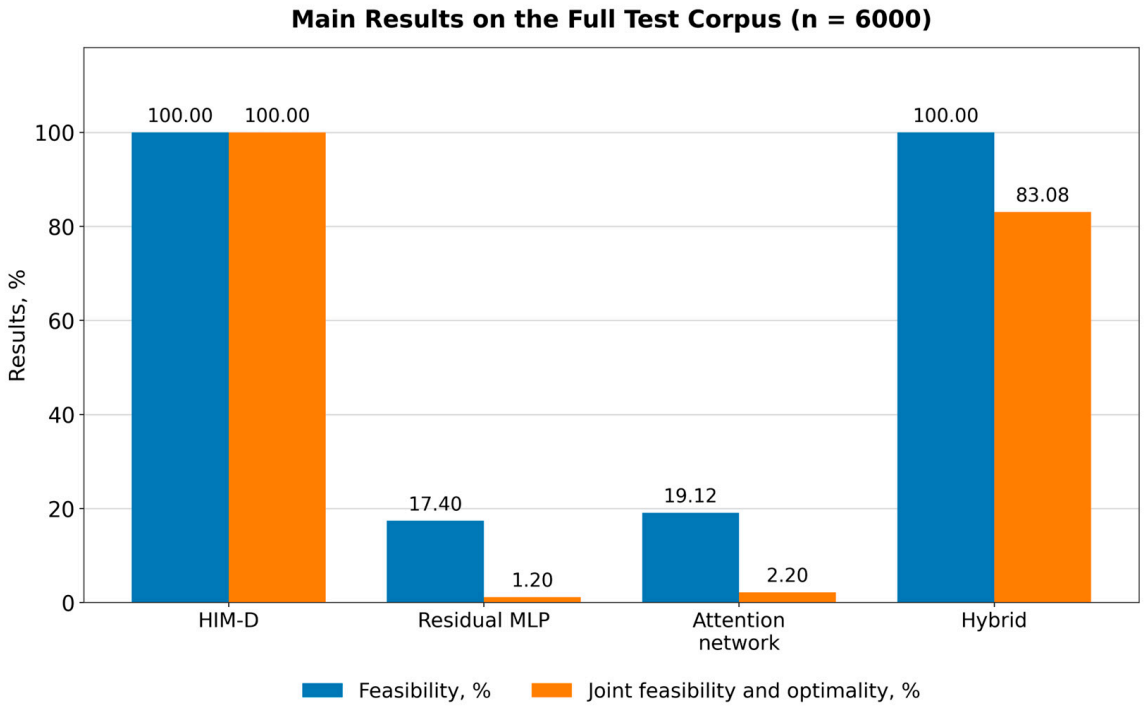


Figure 2. Feasibility and joint feasibility and optimality of the four compared systems on the full test corpus of 6000 configuration tasks, corresponding to the values in Table 9. A configuration counts as feasible when the budget, minimum-quality, maximum-risk, one-option-per-group, and all ten pairwise compatibility conditions hold simultaneously; it counts as jointly feasible and optimal when it is feasible and its objective value lies within 0.1% of the exact optimum found by exhaustive search. Blue bars report feasibility, orange bars the joint criterion. HIM-D attains 100.00% on both measures because it enumerates the entire 1024-configuration space. The procedural hybrid attains 100.00% feasibility, since no unverified candidate is ever returned, but 83.08% on the joint criterion, since a verified candidate is accepted without re-optimization. The Residual MLP and the group-attention network, evaluated without any verification or correction layer, remain below 20% on feasibility and below 3% on the joint criterion.

5.3. Generalization Outside the Training Distribution

Pure neural-network models showed marked sensitivity to distribution shift. Residual MLP feasibility fell from 27.20% on ID problems to 7.60% on OOD problems, a decline of 19.60 percentage points. For the attention network, the decline was 24.17 points, from 31.20% to 7.03%. HIM-D maintained 100.00% feasibility in both subsamples, and the hybrid architecture likewise maintained 100.00%, since every infeasible neural-network solution was replaced by the result of an exact search. Table 10 sets out the ID and OOD feasibility of each system side by side, together with the resulting decline in percentage points and the corresponding joint-optimality figures.

Table 10. Feasibility and joint optimality within (ID) and outside (OOD) the training distribution.

System	ID Feasibility, %	OOD Feasibility, %	Decline, p.p.	ID Joint Optimality, %	OOD Joint Optimality, %
HIM-D	100.00	100.00	0.00	100.00	100.00
Residual MLP	27.20	7.60	19.60	1.13	1.27
Group-attention network	31.20	7.03	24.17	2.93	1.47
Attention network + deterministic verifier (hybrid)	100.00	100.00	0.00	71.73	94.43

The hybrid's joint optimality was higher on the OOD partition (94.43%) than on the ID partition (71.73%). This does not indicate that the neural network generalized better to OOD conditions; on the contrary, it violated constraints more often under OOD conditions, which triggered the exact backup search more frequently and thereby returned the global optimum more often. This result illustrates an important property of hybrid systems: the final reliability indicator depends not only on the quality of the neural network but also on the rule governing control transfer to the formal module.

Figure 3, constructed from the data in Table 10, compares the feasibility of each system on the ID and OOD subsamples. HIM-D and the hybrid remain unchanged at 100.00% in both cases, whereas the Residual MLP declines from 27.20% to 7.60% and the attention network declines from 31.20% to 7.03%, clearly illustrating the vulnerability of direct neural-network inference to distribution shift.

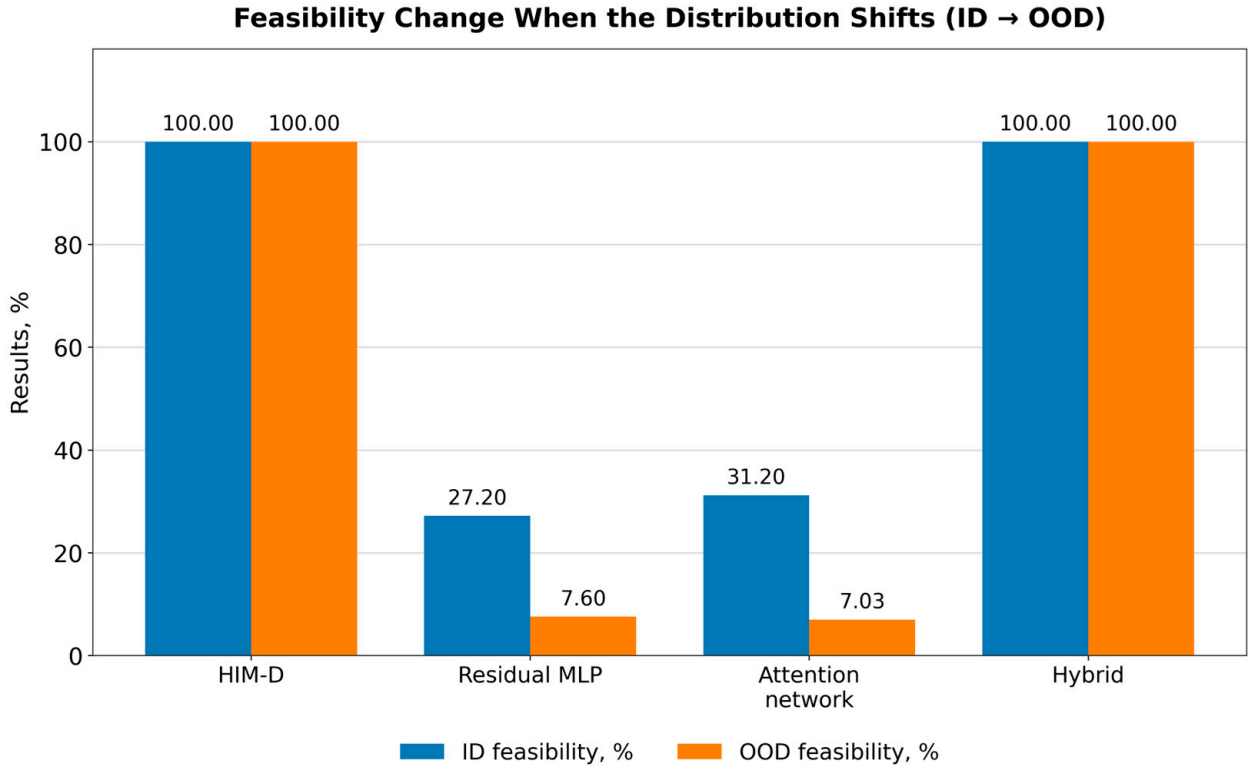


Figure 3. Feasibility of the four compared systems within the training distribution (ID, 3000 tasks) and under the controlled distribution shift (OOD, 3000 tasks), corresponding to the values in Table 10. The OOD partition tightens the budget, quality, and risk thresholds, raises the density of pairwise incompatibility from 5–13% to 16–28%, and shifts the cost and risk ranges, while every instance remains verified to admit at least one feasible solution. HIM-D and the procedural hybrid are unaffected, at 100.00% in both conditions. The Residual MLP falls from 27.20% to 7.60% and the group-attention network from 31.20% to 7.03%, each losing roughly three-quarters of an already low feasibility. Section 5.10 decomposes this decline into the contributions of the individual generation factors.

5.4. Results by Subject Domain

The deterministic system and the hybrid maintained full feasibility across all three numerical parameterizations. Among the pure networks, the attention network performed best in the engineering parameterization, at 20.15%. These figures cannot be read as a comparative evaluation of real educational, engineering, or commercial products. The twelve scenarios are semantic interpretations of one mathematical structure, the generator varies the numerical distributions only at the level of the three parameterizations, and no scenario draws on a real catalog, syllabus, or product line. Differences between domains in Table 11

therefore reflect differences between three sets of generator parameters, not differences between application areas, and the main conclusion concerns the presence or absence of formal verification rather than any system's advantage for a specific course, device, or catalog. Table 11 breaks the pooled figures of Table 9 down by domain, reporting feasibility, joint optimality, and the median objective gap for each system-domain combination.

Table 11. Results by subject domain.

System	Domain	Feasibility, %	Joint Optimality, %	Median Gap, %
HIM-D	Education	100.00	100.00	0.00
HIM-D	Engineering	100.00	100.00	0.00
HIM-D	Commerce	100.00	100.00	0.00
Residual MLP	Education	16.95	0.95	28.06
Residual MLP	Engineering	18.55	1.35	18.25
Residual MLP	Commerce	16.70	1.30	21.16
Group-attention network	Education	17.75	1.65	18.28
Group-attention network	Engineering	20.15	2.80	13.64
Group-attention network	Commerce	19.45	2.15	16.34
Hybrid	Education	100.00	83.90	0.00
Hybrid	Engineering	100.00	82.65	0.00
Hybrid	Commerce	100.00	82.70	0.00

Note: within each system, the domain-level feasibility values average to the overall figure reported in Table 9 (e.g., for the Residual MLP, (16.95 + 18.55 + 16.70)/3 = 17.40%), since the three domains contribute equal numbers of test instances.

5.5. Analysis of Violations

Infeasible neural-network configurations most often violated multiple requirements simultaneously: 2578 such instances occurred for the Residual MLP and 2382 for the attention network. Among single violations, pairwise incompatibility predominated. This result indicates that a locally plausible selection of individual components does not ensure global configuration consistency, and that budget, risk, quality, and compatibility cannot be reliably controlled through independent per-group predictions without formally constraining the solution space. Table 12 breaks these infeasible outputs down by violation type, distinguishing single-constraint violations (incompatibility, budget, quality, and risk) from cases in which several constraints were violated at once.

Table 12. Structure of constraint violations among infeasible configurations.

System	Incompatibility	Budget	Quality	Risk	Multiple Violations	Total Infeasible
Residual MLP	1309	224	429	416	2578	4956
Group-attention network	1483	218	376	394	2382	4853
Hybrid	0	0	0	0	0	0

Note: totals equal 6000 minus the number of feasible configurations reported in Table 9 (6000 − 1044 = 4956 for the Residual MLP; 6000 − 1147 = 4853 for the group-attention network); the hybrid produced zero infeasible outputs by construction.

5.6. Repeatability and Computational Efficiency

Repeatability was evaluated under the two inference rules defined in Section 4.5, because the two are not interchangeable. Under the deterministic rule, all four systems returned identical configurations on all five runs over the 500-task subset: the two networks

because a forward pass on fixed inputs and fixed weights is deterministic, and HIM-D and the hybrid because their search and control-transfer rules are deterministic. The separation between 0.00% and 100.00% therefore appears only under the stochastic rule, in which none of the 500 tasks yielded the same configuration five times for either network. Mean feasibility under stochastic sampling was 16.96% for the Residual MLP and 21.12% for the attention network, against 17.40% and 19.12% under argmax decoding. The defensible conclusion is accordingly narrower than a difference between architectures: exact repeatability is a property of the inference rule and not of the model class, and the contribution of the formal route is that it preserves repeatability without surrendering the feasibility guarantee, whereas a network is repeatable only when sampling is switched off, at which point it still fails the constraints on roughly four-fifths of the tasks.

Neural-network forward passes were faster than deterministic search, but absolute latency remained well below 1 ms per task in every case, given the small size of the configuration problem. HIM-D required 0.167 ms per task on average. The hybrid required 0.139 ms, since exact search was invoked in 80.88% of cases while a valid neural-network configuration was adopted directly in the remainder. These values apply only to the local CPU experiment and should not be compared with the network latencies of cloud-hosted language models. Table 13 collects the parameter counts, latency, exact repeatability, and stochastic-sampling feasibility for all four systems, allowing the speed–reliability trade-off discussed above to be read off directly.

Table 13. Repeatability and computational efficiency of the compared systems.

System	Parameters	Latency, ms	Repeatability, Stochastic Rule, %	Repeatability, Deterministic Rule, %	Feasibility Under Sampling, %	Note
HIM-D	0	0.167	100.00	100.00	100.00	1024 configurations checked
Residual MLP	471,572	0.005	0.00	100.00	16.96	Direct forward pass
Group-attention network	10,596	0.004	0.00	100.00	21.12	Direct forward pass
Hybrid	10,596 + formal module	0.139	100.00	100.00	100.00	Backup search invoked in 80.88% of tasks

Note: The deterministic rule decodes each network output by argmax and repeats the run with fixed inputs and weights; the stochastic rule draws each configuration from the softmax output distribution. All other results in this paper use the deterministic rule. Latency was measured on the machine described in Section 4.4 and applies to the 1024-configuration space only; Section 5.13 reports how it changes as the space grows.

5.7. Hypothesis Outcomes

Table 14 consolidates the pre-specified criterion, the observed result, and the resulting decision for each of the six hypotheses stated in Section 3.2, drawing on the figures reported in Tables 9, 10 and 13 and, for H6, in Section 5.11.

Table 14. Summary of hypothesis tests.

Hypothesis	Criterion	Observed Result	Outcome	Status
H1	HIM-D advantage $\geq$ 5 p.p.	82.60 p.p. vs. MLP; 80.88 p.p. vs. attention network; lower CI bounds > 79 p.p.	Confirmed	Structural check
H2	Smaller OOD decline for HIM-D	0.00 p.p. vs. 19.60 and 24.17 p.p.	Confirmed	Structural check
H3	Equal repeatability under a common deterministic rule; loss of repeatability only under sampling	100.00% for all four systems under argmax decoding; 0.00% for both networks under five stochastic draws	Confirmed	Empirical
H4	Hybrid not inferior to HIM-D within 1 p.p.	Difference of 0.00 p.p.	Confirmed	Structural check
H5	Hybrid improves joint optimality and keeps latency < 1 ms	83.08% vs. 2.20%; 0.139 ms	Confirmed	Empirical
H6	Constraint-aware correction raises feasibility but does not guarantee it	88.87% (best correction, ID) against 100.00% for verification with exact fallback; Section 5.11	Confirmed	Empirical

Note: A structural check is a statement whose direction follows from the design of the experiment, as explained in Section 3.2, and is reported to confirm that the implementation behaves as specified. An empirical statement is one whose outcome was not fixed by the design.

5.8. External Comparison with Modern Language Models

Published results from ConstraintBench [6] are examined here for external context. In that study, six state-of-the-art language models solved 200 problems across ten operations-research domains without access to a formal solver; for each answer, an independent verifier assessed constraint compliance and objective-function quality. These data were not included in the internal statistical tests and are presented solely as an external benchmark for the robustness of direct neural-network solutions to constrained problems. Table 15 reproduces the published per-model feasibility, objective quality, joint feasibility and optimality, and format/API error counts for all six models evaluated in that study. Figure 4 presents the same values graphically.

Table 15. External benchmark: published ConstraintBench results on 200 problems [6].

Model	Feasibility, %	Objective Quality Among Feasible, %	Joint Feasibility and Optimality, %	Format/API Errors
GPT-5.2-pro	65.00	95.20	30.50	1
GPT-5.2	64.00	93.80	21.00	1
Claude Opus 4.6	49.00	96.00	21.50	13
Claude Opus 4.5	50.50	94.00	12.50	5
o4-mini	50.50	89.00	8.50	2
Gemini 3 Pro Preview	44.50	94.80	8.50	0

A direct numerical ranking of HIM-D against the ConstraintBench models would be inappropriate, since the internal corpus uses synthetic problems with 1024 configurations while ConstraintBench spans ten different optimization types and remote commercial models. The qualitative conclusion is nonetheless consistent: without formal verification,

even strong neural-network systems frequently produce solutions that violate one or more conditions, and high objective quality among feasible solutions does not compensate for the risk of an infeasible configuration. ConstraintBench feasibility rates are substantially higher than those of the internal pure neural networks (44.50–65.00% versus 17.40–19.12%), yet the proportion of solutions that are both feasible and optimal remains low (8.50–30.50%), reinforcing the general conclusion regarding the gap between the plausibility of an answer and a guarantee of its feasibility.

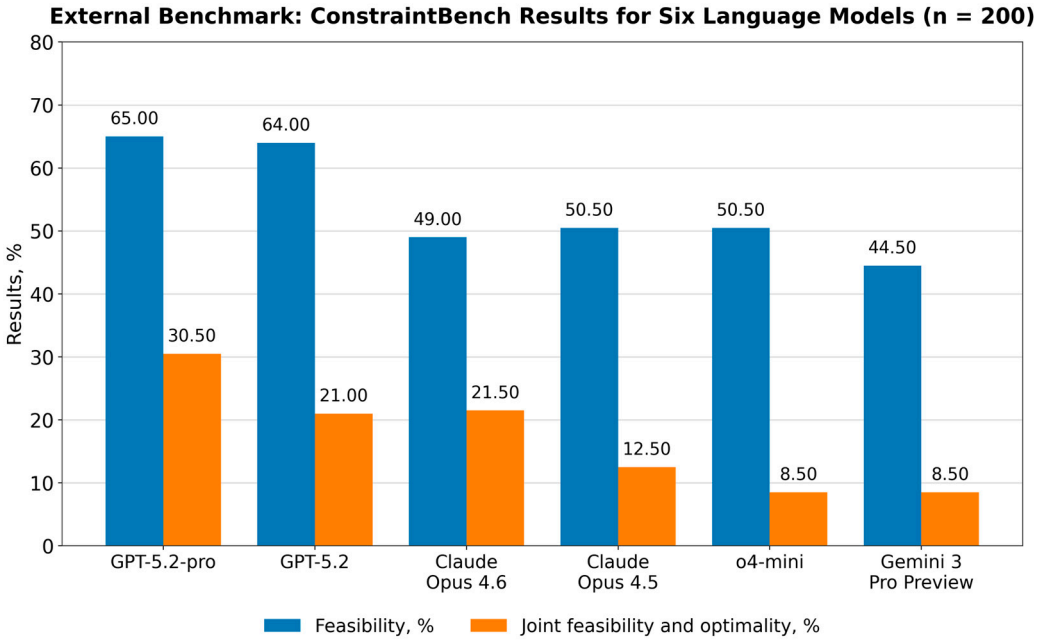


Figure 4. External benchmark: Published results for six contemporary large language models on the ConstraintBench suite of 200 constrained-optimization problems, reproduced from [6] and listed numerically in Table 15. In that study, each model received a fully specified problem without access to a formal solver, returned a structured solution, and had every constraint checked by an independent deterministic verifier. Blue bars show the share of feasible solutions, that is, those satisfying every stated constraint; orange bars show the share that is both feasible and optimal. Even the strongest model, GPT-5.2-pro, produced feasible solutions on 65.00% of problems and jointly feasible and optimal solutions on 30.50%, which indicates that feasibility rather than objective quality is the binding limitation of unaided language-model solvers on this task family. These values were obtained on a different problem set and in a different computational environment from the internal experiment and are not pooled with it.

5.9. Preliminary Note on the Extended Analyses

Sections 5.9–5.16 report analyses added after the primary evaluation had been completed. Each was carried out with the same generator, verifier, solver, and protocol, but on corpora regenerated independently of the pre-specified test set on which Tables 9–14 are based, so that the additional conditions could be introduced without altering it. Each table below therefore carries its own internal reference row or reference column, and its values should be read against that reference rather than against the pooled figures of Table 9; the two sets of numbers answer different questions and are not interchangeable. Latencies reported in Sections 5.12 and 5.13 were likewise measured in one environment and should be compared with one another rather than with Table 13.

5.10. Staged Ablation of the Distribution Shift

The out-of-distribution partition used in Section 5.3 changes five things at once: the density of pairwise incompatibility, the budget threshold, the minimum-quality threshold,

the maximum-risk threshold, and the ranges from which cost and risk are drawn. The combined design establishes that direct neural inference is fragile under such a shift, but it cannot say which component is responsible. Each factor was therefore moved to its OOD level in isolation, holding the others at their ID level, and the two trained networks were evaluated on the resulting corpora without retraining. Table 16 and Figure 5 report the outcome, together with the share of the 1024 configurations that remains feasible in each condition, which measures how far each factor contracts the feasible region.

Table 16. Staged ablation of the out-of-distribution shift: effect of each generation factor in isolation on the feasibility of direct neural inference (3000 instances per condition, evaluated without retraining).

Factor Moved to Its OOD Level	Feasible Configurations per Instance, % of 1024	Residual MLP Feasibility, %	Change, p.p.	Group-Attention Feasibility, %	Change, p.p.
None (ID reference)	11.81	29.23	reference	35.70	reference
Incompatibility density	3.17	11.60	−17.63	12.23	−23.47
Budget threshold	9.74	26.57	−2.66	33.60	−2.10
Minimum quality	9.62	24.33	−4.90	31.67	−4.03
Maximum risk	10.04	26.37	−2.86	32.90	−2.80
All three thresholds jointly	6.75	19.90	−9.33	27.40	−8.30
Cost and risk ranges	11.81	29.17	−0.06	36.53	+0.83
Full shift (all factors)	1.81	7.77	−21.46	8.83	−26.87

Note: p.p. denotes percentage points. Changes are measured against the ID reference row of this table, which was regenerated under the conditions described in Section 5.9 and is therefore not identical to the ID column of Table 10.

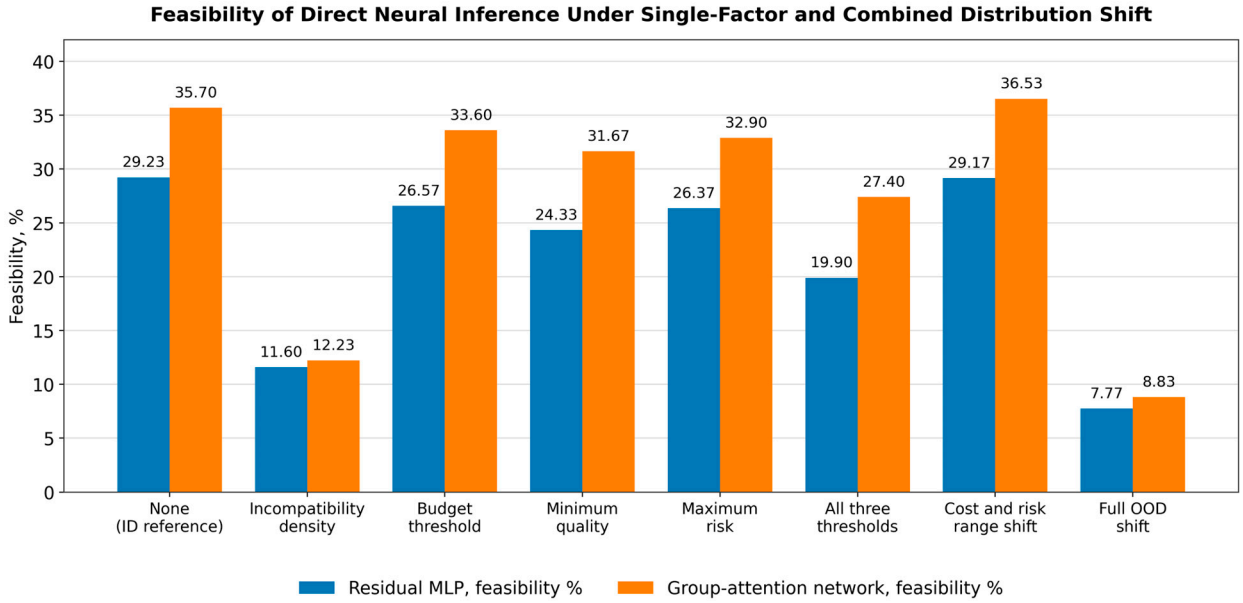


Figure 5. Feasibility of the two pure neural-network models when each generation factor is moved to its out-of-distribution level in isolation, with all remaining factors held at their in-distribution level (3000 instances per condition; values listed in Table 16). Blue bars report the Residual MLP, orange bars the group-attention network. The leftmost pair is the in-distribution reference and the rightmost pair the full shift used in Section 5.3. Raising the density of pairwise incompatibility alone reproduces most of the decline observed under the full shift, whereas shifting the cost and risk ranges alone leaves both networks unchanged, which identifies the combinatorial structure of the constraints rather than the covariate shift as the source of the degradation.

The decomposition is unambiguous. The density of pairwise incompatibility accounts for the greater part of the degradation on its own, costing 17.63 percentage points for the Residual MLP and 23.47 for the attention network, while the three numerical thresholds together account for 9.33 and 8.30 points. The shift of the cost and risk ranges, which is the only factor that constitutes a covariate shift in the usual sense of the term, has no measurable effect in isolation: the two networks change by −0.06 and +0.83 points, well inside the sampling variation of a 3000-instance sample. What breaks direct inference is therefore not that the inputs look unfamiliar but that the combinatorial structure of the feasible region becomes harder, and the attention network, whose architecture is designed to exploit relations between groups, is the more sensitive of the two to precisely that factor.

The factors are also strongly sub-additive. The individual declines sum to 28.11 percentage points for the Residual MLP and 31.57 for the attention network, against 21.46 and 26.87 under the full shift, which is the saturation to be expected once feasibility approaches the level attainable without any modeling at all. That comparison is worth making explicit. Expressed as a multiple of the share of configurations that are feasible in the space, the two networks operate at 2.48 and 3.02 times chance under ID conditions and at 4.29 and 4.88 times chance under the full shift. Their discriminative ability, measured in this relative way, does not deteriorate at all; what collapses is the absolute feasibility rate, and it collapses because the feasible region contracts from 11.81% to 1.81% of the space. This is the sense in which the OOD result should be read: it is a statement about how quickly an unverified predictor becomes useless as constraints tighten, not a claim that the networks fail to generalize in the representational sense.

5.11. Comparison with Constraint-Aware Correction of the Same Candidate

The two neural baselines of Section 5.2 are evaluated without any correction, which establishes the risk of unverified direct prediction but does not show that the formal route is necessary rather than merely sufficient. The literature reviewed in Section 2.3 offers three families of alternatives: penalizing violations during training, projecting or repairing the output afterward, and constraining the decoding itself. Differentiable projection layers such as LinSATNet [17] and HardNet [18] cannot be applied directly here, for the reason given in Section 2.3: an arbitrary binary incompatibility matrix combined with a global-minimum requirement is not a positive linear or convex constraint set. Three surrogates in the same spirit, and appropriate to this constraint class, were therefore implemented: two act on the identical attention candidate after inference, while the first retrains the network with an additional term in the loss. The first adds to the cross-entropy loss a differentiable penalty equal to the expected number of violated compatibility pairs under the predicted product distribution. The second repairs the argmax candidate by repeatedly reassigning the group whose reassignment most reduces the number of violated constraints, ties being broken by network log-probability, and stops when no single change improves matters. The third scores the 32 highest-probability configurations and returns the first the verifier accepts. None of the three invokes an exact solver. Table 17 places them beside the uncorrected candidate and the procedural hybrid.

Three findings follow. First, correction helps, and helps a great deal: screening the 32 most probable configurations raises in-distribution feasibility from 32.53% to 88.87%, and greedy repair achieves 86.67% at a lower computational cost. The soft penalty is by far the weakest of the three, raising feasibility to 39.00% while slightly lowering joint optimality, which is the behavior predicted in Section 2.3: a penalty changes the expected number of violations but places no constraint on any individual output. Second, none of the three attains a guarantee. The best of them still returns an infeasible configuration on roughly one instance in nine within the training distribution, and their advantage erodes

under shift, where beam decoding falls to 52.77% because the feasible region has contracted to a point at which a beam of 32 frequently contains no feasible member. Third, the gap in joint optimality is larger than the gap in feasibility: correction methods are steered by feasibility alone and settle on a nearby feasible configuration rather than the best one, so they remain below 16% on the joint criterion in every condition. H6 is therefore supported. The practical reading is not that these methods are inadequate, since a sixfold reduction in the violation rate at negligible cost is valuable, but that they change the probability of an error and not the class of outcomes, which is the distinction drawn in Section 2.3 between soft and hard constraints.

Table 17. Constraint-aware correction of the same attention-network candidate, compared with the uncorrected candidate and with the procedural hybrid (3000 in-distribution and 3000 out-of-distribution instances).

System	ID Feasibility, %	ID Joint Feasibility and Optimality, %	OOD Feasibility, %	OOD Joint Feasibility and Optimality, %
Attention network, argmax, no correction	32.53	3.27	7.33	1.57
Attention network + soft-constraint penalty in the loss	39.00	2.70	11.57	2.00
Attention network + greedy feasibility repair	86.67	11.00	59.77	15.73
Attention network + constraint-aware beam decoding (k = 32)	88.87	11.13	52.77	13.10
Attention network + independent verifier + HIM-D fallback (procedural hybrid)	100.00	70.73	100.00	94.23

Note: All five rows decode the same trained attention network, except for the penalty row, which is retrained with the additional loss term. The corpora were regenerated as described in Section 5.9, so the first row is the internal reference for this table rather than the corresponding entry in Table 10.

5.12. Comparison with Constraint Programming and Mixed-Integer Programming

HIM-D is a purpose-written exhaustive solver, which raises two questions a general-purpose solver can answer: whether its output is in fact the global optimum, and how it compares with the standard tools of the field. The same 300 instances were therefore solved three ways: by HIM-D, by the CP-SAT solver of OR-Tools [10] restricted to a single search worker, and by a mixed-integer linear model solved with branch and bound in CBC [27]. The compatibility relations were encoded as pairwise exclusion constraints and the objective as a linear function of the selection indicators, so that all three solve the same problem exactly. Table 18 reports the agreement and latency of the three solvers.

The three solvers agree on the optimal objective value for every one of the 300 instances. This is the strongest available evidence that the HIM-D implementation is correct, since two independent and widely used solvers reproduce its answer exactly, and it is worth more than the internal consistency checks reported in Section 5.1, which could not have detected a systematic error in the enumeration itself. On latency, exhaustive enumeration is roughly ten times faster than CP-SAT and thirty times faster than CBC at this size, because a space of 1024 candidates is evaluated by a handful of vectorized array operations while both general-purpose solvers must first construct a model. That ordering is an artifact of the problem size and reverses quickly, as the next section shows.

Table 18. Agreement and latency of three exact solvers on the same 300 configuration instances.

Solver	Feasibility, %	Joint Feasibility and Optimality, %	Instances with the Same Optimal Objective as HIM-D, %	Mean Latency, ms
HIM-D, exhaustive enumeration of 1024 configurations	100.00	100.00	100.00 (by definition)	0.33
CP-SAT (OR-Tools), single worker	100.00	100.00	100.00	3.40
MILP (CBC), branch and bound	100.00	100.00	100.00	9.62

5.13. Scaling of the Exact Route Beyond 1024 Configurations

The sub-millisecond latency reported in Table 13 is a property of a space of 1024 candidates and should not be read as evidence about the response time of a deployed configurator. The number of groups and the number of options per group were therefore varied over a range spanning almost five orders of magnitude in the size of the configuration space, and the same two exact routes were timed on 60 instances of each size. Everything else in the generator, including the density of incompatibility and the rule by which thresholds are derived, was held fixed. Table 19 reports the resulting latencies.

Table 19. Mean latency per task of the two exact routes as the configuration space grows (60 instances per configuration size, single-threaded).

Groups G	Options O	Configuration Space O^G	HIM-D Latency, ms	CP-SAT Latency, ms	CP-SAT Proven Optimal, %
5	4	1024	0.27	3.07	100.00
6	4	4096	1.11	3.61	100.00
7	4	16,384	5.34	4.04	100.00
6	6	46,656	13.27	5.09	100.00
8	4	65,536	25.88	4.48	100.00
8	6	1,679,616	1713.69	7.26	100.00
10	6	60,466,176	not run	8.92	100.00

Note: The first row corresponds to the space used throughout Sections 5.1–5.8. Latencies here were measured on 60 instances per size and therefore differ slightly from the 300-instance measurements of Table 18. Exhaustive enumeration was not run at the largest size, where the extrapolated cost is on the order of a minute per task.

The two routes behave in entirely different ways. Exhaustive enumeration grows slightly faster than the size of the space, because the cost of checking the pairwise relations of a single candidate itself grows quadratically in the number of groups: a factor of 1640 in space size produces a factor of about 6400 in latency, and by 1.68 million candidates the route needs 1.71 s per task, which is three orders of magnitude beyond any interactive budget. CP-SAT grows by a factor of less than three across the same range and remains below 10 ms at 60 million candidates. The crossover lies between 4096 and 16,384 candidates, that is, at roughly one order of magnitude above the space studied here. Figure 6 shows both curves.

Two consequences follow for the interpretation of this paper. The latency advantage of HIM-D reported in Table 13 is real but local, and the sentence in H5 about sub-millisecond response should be understood as applying to this experimental setting only. The architectural claim, by contrast, is unaffected: the hybrid requires an exact backend but does not require this one, and substituting CP-SAT changes only the constant, since the verifier, the control-transfer rule, and the resulting guarantee are identical. Anyone applying the

pattern to a realistically sized catalog should therefore use a constraint solver as the fallback and treat exhaustive enumeration purely as the reference against which the solver was validated in Section 5.12.

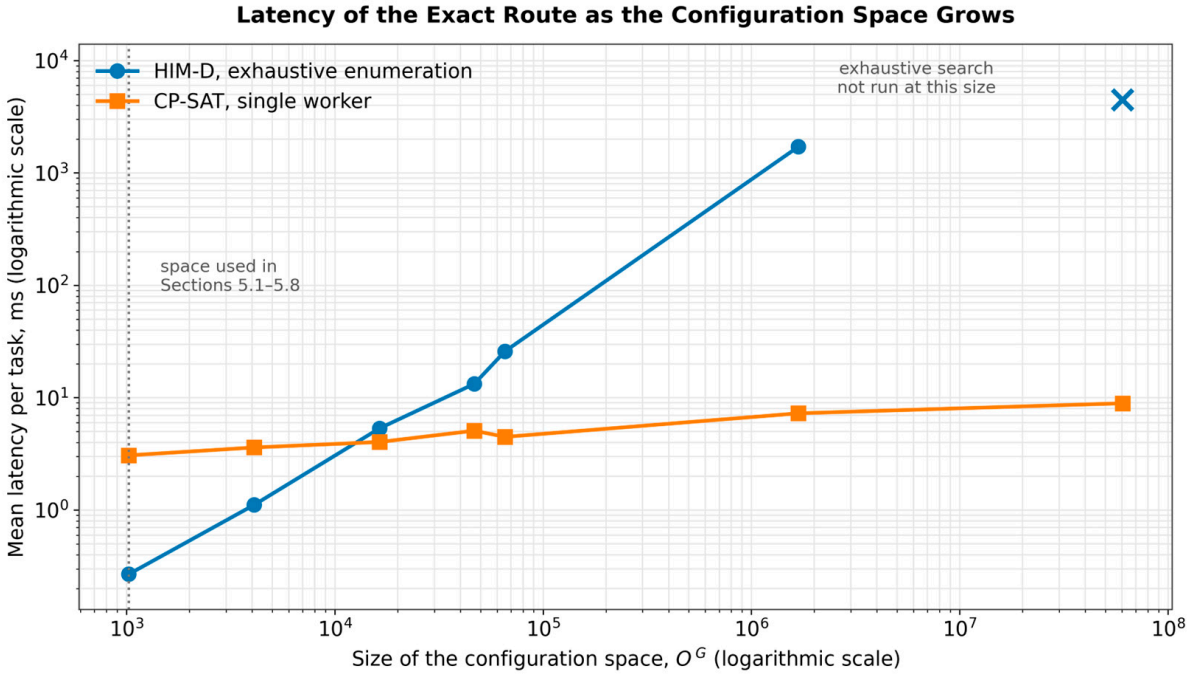


Figure 6. Mean latency per task of the two exact routes as a function of the size of the configuration space, on logarithmic axes (60 instances per size; values listed in Table 19). Blue circles show exhaustive enumeration over the whole space, and orange squares the CP-SAT solver restricted to a single worker; both return a proven global optimum in every case. The dotted vertical line marks the 1024-configuration space used in Sections 5.1–5.8. Enumeration is the faster route only below roughly 10^4 candidates and becomes impracticable beyond about 10^6, whereas the constraint solver stays below 10 ms across the entire range, which is why the architectural conclusion of this paper is stated independently of the choice of exact backend.

5.14. Sensitivity of the Guarantee to an Incomplete Rule Set

Section 2.4 states that the guarantee is conditional, and that the first condition is that the verifier implement every mandatory rule. In a real configurator, that condition is the least likely of the four to hold, since rule sets are compiled by hand and are rarely complete. The cost of violating it was therefore measured directly. A fraction of the ten pairwise compatibility rules, chosen at random, was withheld from the verifier, while the fallback solver continued to use the complete rule set, and the returned configurations were then evaluated against the complete rule set. Any loss of feasibility is therefore attributable entirely to candidates accepted on the fast route that violate a rule the verifier was not given. Table 20 reports the outcome.

The degradation is gradual and close to linear: withholding one of the ten pairwise rules costs about two points of true feasibility, and withholding half costs about eleven. The mechanism is visible in the acceptance column. As the verifier is weakened, it rejects fewer candidates, so the fast route is taken more often, and every additional acceptance is an opportunity to return a configuration that violates an unchecked rule. With half the rules missing, the verifier accepts 47.60% of in-distribution instances and returns a truly infeasible configuration on 10.80% of all instances. Two conclusions are worth stating plainly. The guarantee does not degrade catastrophically, because the fallback solver retains the complete rule set and is still invoked on the majority of instances, so an incomplete verifier remains substantially better than no verifier. But the guarantee is

genuinely conditional, and a deployment claiming full feasibility is making a claim about the completeness of its rule base, not about its architecture. This is the strongest practical caveat the present experiment can offer, and it argues for treating the rule set as a versioned, independently tested artifact of the kind described in Section 2.4.

Table 20. Effect of withholding compatibility rules from the verifier on the feasibility of the configurations the hybrid finally returns, evaluated against the complete rule set (3000 instances per condition).

Rules Withheld from the Verifier	Rules Checked, of 10	ID Final Feasibility, %	ID Fast-Route Acceptance, %	OOD Final Feasibility, %	OOD Fast-Route Acceptance, %
None	10	100.00	36.80	100.00	8.57
10%	9	97.70	39.10	98.77	9.80
20%	8	96.20	40.60	96.63	11.93
30%	7	93.73	43.07	95.53	13.03
50%	5	89.20	47.60	89.83	18.73

5.15. Stability Across Independent Replications

The results of Sections 5.2–5.7 come from a single seed, which leaves open whether the reported feasibility rates are stable properties of the architectures or accidents of one draw. The entire pipeline, from corpus generation through weight initialization, batch ordering, and evaluation, was therefore repeated five times with distinct seeds. Table 21 reports each replication and the resulting mean and standard deviation.

Table 21. Five independent end-to-end replications of the pipeline, each regenerating the corpus and retraining both networks (6000 test instances per replication).

Replication Seed	Residual MLP Feasibility, %	Group-Attention Feasibility, %	Hybrid Feasibility, %	Hybrid Joint Feasibility and Optimality, %	Fallback Rate, %
20260711	18.07	22.68	100.00	80.87	77.32
20260712	18.47	18.83	100.00	84.22	81.17
20260713	19.67	13.28	100.00	87.67	86.72
20260714	12.60	18.70	100.00	83.57	81.30
20260715	21.38	18.47	100.00	83.93	81.53
Mean ± SD	18.04 ± 3.30	18.39 ± 3.35	100.00 ± 0.00	84.05 ± 2.42	81.61 ± 3.35

Note: SD denotes the sample standard deviation over the five replications. The first replication reuses the primary generation seed; because the replication pipeline redraws the corpus and re-initializes both networks, its values differ from those of Table 9 by between 0.67 and 3.56 percentage points, which is on the order of the replication standard deviations given in the last row. When the partitions are separated, in-distribution feasibility is 28.52 ± 5.32% for the Residual MLP and 30.13 ± 5.12% for the group-attention network, against 7.55 ± 1.29% and 6.65 ± 1.63% out of distribution.

The variability of direct neural feasibility is substantial: a standard deviation of about 3.3 percentage points on a mean of about 18 means that roughly a fifth of the reported value is run-to-run noise, and the two architectures are not reliably distinguishable from one another at this sample size, since their intervals overlap almost completely. That is itself a finding worth stating, because it means the ranking of two unverified predictors on this task is not a stable quantity, and any comparison of such predictors that rests on a single run should be treated with caution. The values reported in Table 9, namely 17.40% and 19.12%, and the hybrid's 83.08% on the joint criterion, all lie within one standard deviation of the corresponding replication means. Hybrid feasibility, in contrast, has zero variance across every replication, which is what the structural argument of Section 3.2 requires and

is the point of the architecture: the quantity that is guaranteed does not fluctuate, while the quantity that depends on the network, the joint criterion, inherits the network's variability through the fast-route acceptance rate.

5.16. Sensitivity to the Objective-Function Coefficients

The coefficients of Equation (2) were fixed in advance, and Section 3.1 explains how they were chosen. Because the hard constraints do not involve the objective, feasibility is invariant to them by construction, and only two quantities can move: which configuration is optimal, and how often an accepted fast-route candidate happens to be that configuration. Each weight was halved and doubled in turn, the optima were recomputed, and the hybrid was re-evaluated. Table 22 reports the result.

Table 22. Sensitivity of the optimum and of the hybrid to the objective-function weights (6000 instances per setting).

Weight Setting	Instances Whose Optimal Configuration Is Unchanged, %	Hybrid Joint Feasibility and Optimality, %
Reference: 18, 0.6, 0.15	100.00	80.87
Preference weight halved, 9	83.23	80.57
Preference weight doubled, 36	60.57	81.18
Risk weight halved, 0.3	70.68	80.60
Risk weight doubled, 1.2	60.93	80.82
Quality weight halved, 0.075	71.48	80.67
Quality weight doubled, 0.3	79.97	81.12

The weights are consequential for the answer and immaterial for the conclusion. Halving or doubling any one of them changes the optimal configuration on between 16.77% and 39.43% of instances, so the objective is not a formality and a deployment could not adopt these values without justification. Yet the hybrid's joint feasibility and optimality stays within 0.6 percentage points of the reference in every setting, because the fast-route acceptance rate is governed by the constraints rather than by the objective, and the fallback returns whatever optimum the current weights define. The architectural finding is therefore robust to the choice of coefficients, while the recommendation delivered to a user is not; in an application, the weights would have to be elicited from the domain rather than assumed, and the present values should be read as an experimental setting.

6. Discussion

6.1. Main Conclusion

The principal conclusion concerns a specialized architectural setting. The experiment does not establish that HIM-D or the procedural hybrid can solve arbitrary configuration problems. It shows that, when a bounded task family can be represented by explicit variables, hard constraints, and a verifiable objective, separating candidate generation from formal acceptance control prevents infeasible neural outputs from being treated as valid final decisions.

The experiment supports one claim, and it is an architectural rather than a comparative one: mandatory formal verification changes the class of outcomes a system can return, not merely the average quality of what it returns. Everything else in the results follows from that distinction. HIM-D was an exact reference by construction and its perfect feasibility is not a finding; the two networks performed direct inference without a constraint-aware layer and their low feasibility is a finding only in its magnitude. What the procedural hybrid contributes is that no unverified result is ever presented as final,

and Section 5.11 shows that this is not a distinction without a difference: constraint-aware beam decoding, the strongest correction available without an exact solver, reached 88.87% feasibility in distribution, and no correction of any kind exceeded 59.77% under shift, whereas verification with fallback reached 100.00% in both. The gap between those two figures is the whole content of the paper.

Three qualifications follow immediately, and none of them was resolved by this experiment. The reference bound established here holds in a finite space in which all constraints are known and exhaustive search is cheap, and Section 5.13 shows that the second of those properties fails about one order of magnitude above the space studied. The guarantee is conditional on the completeness of the rule set, and Section 5.14 quantifies what it costs when that condition is relaxed. The reliability of the hybrid is a property of the control-transfer contract rather than of the network, which means not only that it can be reasoned about but also that improving the network does not improve the guarantee, only the frequency with which the expensive route is avoided.

6.2. Why Neural Networks Lost Feasibility

The pure neural-network models received all threshold and compatibility information but were trained with cross-entropy on individual group decisions. That loss rewards matching the optimal class within each group and places no cost on globally incompatible combinations, and the attention mechanism improved matters only moderately. The predominance of simultaneous violations shows why: the choice of one element affects the budget, the risk, the quality, and the compatibility relations with several other elements at once, so errors are coupled rather than independent. Section 5.10 sharpens the diagnosis by showing that what degrades performance is the density of the incompatibility relations rather than any shift in the numerical attributes, which locates the difficulty squarely in the combinatorial structure of the problem rather than in the input distribution.

The improvements available at the training level can be ordered by how directly they attack that structure, and the evidence of Section 5.11 bears on each. Weakest is the addition of a violation penalty to the loss, which raised feasibility from 32.53% to 39.00% here; it shifts the expected number of violations but constrains no individual output, and it slightly degraded joint optimality by trading objective quality for feasibility mass. Next comes a change of learning target: training the network to predict the feasible region, the active constraints, or a calibrated probability that a candidate will violate a rule, rather than to reproduce the optimal configuration. This does not by itself produce feasible outputs but it produces a usable acceptance signal, and it is the direction most likely to reduce the fallback rate of 80.88% observed here, since the network currently supplies no information about its own reliability. Third is architectural encoding of the constraints, through differentiable projection or a satisfiability layer [13,17,18,22]; for the constraint class studied here, this is only partially available, since an arbitrary incompatibility matrix is neither a positive linear nor a convex set, but a hybrid encoding that handles the three additive thresholds architecturally and leaves the pairwise relations to the verifier is a realistic intermediate design. Fourth, and most direct, is putting a solver in the training loop, whether by differentiating through it [14,15] or by using it to generate corrective supervision, so that the network learns from the solver's decisions rather than only from its answers. Finally, the corrective decoding of Section 5.11 requires no retraining at all and delivered the largest single improvement measured here, which suggests that in practice the cheapest gains lie in inference-time search rather than in the loss function. None of these removes the need for verification before output; they change how often the fallback is invoked, which is an efficiency question rather than a correctness one.

6.3. Practical Role of the Hybrid Architecture

The hybrid system eliminated all infeasible outputs through a clear separation of responsibilities: the neural network neither validated its own decision nor modified the rules, and its only function was to generate a candidate. The verifier independently recomputed the constraints, and HIM-D received control only when verification failed. The guarantee of feasibility therefore derived not from any property of the network, but from the fact that no result was returned to the user before formal verification. The hybrid did not always reach the global optimum, since a valid neural-network candidate was accepted without re-optimization. In applications where the cost of error or of computation is particularly high, the balance can be tightened—for example, by running HIM-D for every instance, or by accepting a neural-network candidate only after a provable optimality-gap estimate.

The proportion of fallback invocations was 80.88%, meaning that the fast neural route was used for only 19.12% of tasks. This value coincides with the feasibility of the pure attention network, since the switching condition depends solely on whether its argmax candidate satisfies every constraint, and it is stable across replications at $81.61 \pm 3.35\%$ (Section 5.15). The compact network has therefore reduced the load on the formal kernel only marginally, and the corrective decoding of Section 5.11 suggests a cheap improvement: screening a small beam before verification would raise the acceptance rate substantially without altering the guarantee, because the verifier still decides. The more fundamental improvement is for the network to predict the feasible region and its own probability of violation rather than only the optimal configuration, so that the system can tell in advance which instances need the solver.

6.4. Implications for Education, Engineering, and Commerce

The following observations are conditional on the architectural evidence presented above and do not rest on any domain evaluation, since the corpus contains no educational, engineering, or commercial data. Read in that light, the pattern is straightforward. Where a neural component interprets a request, adapts an explanation, or ranks options, its output is a proposal, and any property that must hold with certainty—mathematical correctness in a teaching system, compatibility and resource limits in an engineering selection, price and availability in a commercial configurator—has to be established by a separate mechanism before the result is shown. The broader shift from predictive to prescriptive analytics makes the same point in a different vocabulary, by taking decision quality rather than predictive accuracy as the object of evaluation [33]. Table 23 sets out the implication as a decision rule: it pairs five representative application contexts with the minimum evidentiary requirement each imposes and the architecture that meets it, and should be read as a hypothesis about where the boundary between proposal and guarantee ought to lie, derived from the present results and requiring domain validation before use.

The educational, engineering, and commercial scenarios used in the experiment are semantic parameterizations of a common numerical configuration structure rather than independent validations on real domain datasets. They illustrate how the same specialized architecture may be interpreted in different application contexts, but they do not by themselves demonstrate operational effectiveness in those sectors. A real deployment would require a separately constructed configurational module, domain-specific constraints, representative data, and independent validation.

As Table 23 is read from top to bottom, the recommended architecture shifts systematically from network-only at the low-risk end to a deterministic core at the critical-system end, mirroring the feasibility gap between the unaided networks and the hybrid documented in Table 9.

Table 23. Recommended architecture by application risk level.

Context	Minimum Evidentiary Requirement	Recommended Architecture
Low-risk recommendation	Acceptable preference match, visible warning	Neural network with uncertainty reporting
Educational explanation	Verified mathematical correctness, solution log	Neural network plus a symbolic or deterministic verifier
Commercial equipment configurator	Full compatibility, accurate price, availability	Dialogue neural network plus a formal configurator
Engineering selection	100% coverage of mandatory rules, version control	HIM-D or a formal solver with a neural-network interface
Critical system	Certified limits, independent verification, fail-safe behavior	Deterministic core; neural network only as an auxiliary module

6.5. Comparison with the Targeted Literature

These results are consistent with the typology of machine learning for combinatorial optimization [4,5]. The feedforward networks used here solved the problem as a multi-output classification task with no formal algorithm in the loop, so their low feasibility does not contradict studies in which learning strengthens a solver, differentiates through it, or projects the output onto the feasible set [13–18]. Section 5.11 makes the relationship concrete rather than rhetorical: implementing the correction families of Section 2.3 in forms appropriate to this constraint class recovers a large part of the feasibility gap, which is precisely the reason such methods exist, and still leaves the residual that only verification closes.

The procedural hybrid is closer to solver-backed systems than to jointly trained neural symbolism. Like OptiMUS it separates the interpretive component from the formal solver [9], but the input here is already machine-readable, so what is tested is the quality of the final candidate rather than the formulation step. Unlike ConstraintBench [6], the formal module does not merely score the error after the fact; it changes the route before the result is issued, which is why the hybrid reaches full feasibility despite the low feasibility of the network inside it. The high fallback rate shows how little of the formal work the current network avoids. Three directions remain the most promising, and Section 5.11 indicates their likely order of return: constrained decoding at inference time, which is cheap and immediately effective; learning the feasible region and a calibrated violation probability, which would make the fallback selective; and solver-in-the-loop training, which is the most expensive and the most likely to change the network's behavior fundamentally.

6.6. Theoretical Implications and Scientific Novelty

This study clarifies the role of hierarchical imitation models within modern AI architecture. Their advantage manifests not as a universal learning capability, but as a mechanism for explicitly representing the feasible domain, applying rules reproducibly, logging decisions, and enabling independent verification. The novelty of this work lies in operationalizing this principle within a single corpus and in separately measuring three levels: the quality of the direct candidate, the correctness of the verifier, and the behavior of the conditional fallback. This separation makes it possible to analyze the reliability of the hybrid without attributing a formal guarantee to the neural network itself.

The potentially transferable contribution is not the particular exhaustive HIM-D implementation but the procedural pattern by which a specialized formal module constrains probabilistic inference. The domain model may be manually specified or constructed with AI assistance; nevertheless, its acceptance as an operational verifier

requires independent validation. This creates a development-operation distinction: an AI model may participate in constructing its future constraint mechanism during development, whereas the validated and versioned mechanism remains external to probabilistic candidate generation during operation.

7. Limitations of the Study

The first limitation concerns the synthetic nature of the corpus and the way subject content was defined. The twelve explicitly described scenarios make the composition of the problems transparent, but they do not make the corpus educational, engineering, or commercial data. Component names are semantic interpretations of variant indices, and the numerical distributions were defined at the level of three general parameterizations. Nothing in the corpus reproduces ambiguous textual requirements, incomplete catalogs, temporary unavailability, specification errors, regulatory documents, or expert disagreement, and the generator guarantees that a feasible solution exists, which real configuration problems do not. The consequences for the reported numbers should be stated concretely rather than in general terms. The feasibility rates of the neural baselines would not transfer, because they depend on the density of the incompatibility relations, which Section 5.10 identifies as the dominant factor and which is a free parameter here. The perfect feasibility of the formal route would transfer only insofar as the rule set is complete, which Section 5.14 shows is a strong assumption. What does transfer is the ordering of the architectures and the mechanism behind it. Establishing effectiveness in any real setting would require a separate study on anonymized domain problems, with subject-matter review of the rules and criteria for external validity fixed in advance.

The second limitation concerns the size of the configuration space. Exhaustive search over 1024 candidates guaranteed the global optimum at negligible latency, and Section 5.13 measures where that ceases to be true: the exhaustive route is overtaken by a constraint solver at roughly 10^4 candidates and becomes impracticable, at more than a second per task, by 1.7×10^6. Larger spaces therefore require branch and bound, constraint programming, mixed-integer optimization, decomposition, or approximation, and the full correctness achieved here is not evidence that exhaustive search scales. The architectural conclusion is insulated from this, since Section 5.12 confirms that a general-purpose solver returns the identical optimum and can replace HIM-D in the fallback position without changing anything else, but the latency figures in Table 13 are specific to this setting and should not be quoted as response times for a deployed configurator.

The third limitation concerns the neural-network baseline models, which were intentionally compact and trained for a limited number of epochs on a CPU. Larger models, constraint-aware loss functions, differentiable optimization layers, and solver-feedback learning could substantially improve feasibility. However, external results from Constraint-Bench show that even considerably more powerful language models do not eliminate the formal-robustness problem [6].

The fourth limitation is the indirect nature of the external comparison. The language-model metrics are drawn from another study and do not constitute a randomized comparison on a single corpus. A useful next step would be to run selected open-source and commercial models on the same 6000 tasks, using the same inference scheme, token budget, and independent verifier.

The fifth limitation is the absence of a user study. This work evaluates machine correctness and computational properties but does not measure trust, ease of interaction, or the system's influence on human decisions. Assessing the educational and commercial utility of the system would require a separate study with human participants, pre-approved by an ethics committee.

The sixth limitation concerns the comparative baselines. Section 5.11 adds constraint-aware correction of the same candidate and Section 5.12 adds CP-SAT and mixed-integer programming, but the comparison remains incomplete in two respects. Differentiable projection layers such as LinSATNet and HardNet were not evaluated in their native form, because the constraint class studied here falls outside the sets for which they provide guarantees, so the surrogates used in Section 5.11 test the idea rather than those specific architectures. Solver-in-the-loop training was not evaluated at all. In addition, because exhaustive HIM-D search simultaneously supplies both the solution and the reference bound, this paper measures the effect of architectural verification and does not establish superiority over modern constraint-aware learning. A full treatment would train a projection-based or solver-differentiated model on this corpus and compare it against the hybrid on all of feasibility, joint optimality, and latency.

The seventh limitation concerns the external validity of the synthetic generator. Guaranteed decidability and thresholding relative to a reference configuration can create regularities that are absent from real catalogs. A useful next step would include expertly labeled real or semi-synthetic configurations, inter-expert agreement checks, and separate validation of rule correctness. General principles for the verification and validation of simulation models require distinguishing among implementation correctness, conformity with the conceptual model, and applicability to the real system [34].

An eighth limitation concerns the statistical basis of the neural results. Section 5.15 reports five replications and a standard deviation of roughly 3.3 percentage points on the feasibility of each network, which is large relative to the difference between the two architectures and means that the ordering of the Residual MLP and the group-attention network in Table 9 should not be relied upon. Five replications are enough to bound the variability but not to estimate it precisely, and a study whose object was the comparison between neural architectures rather than between architectural classes would need considerably more.

Specialization reduces the requirement for unrestricted cross-domain generalization but does not eliminate the scalability problem. A local configuration family may itself contain a very large solution space, making exhaustive fallback impractical. The present experiment therefore validates the procedural principle only for a bounded finite space and does not determine which exact, approximate, or decomposition-based solver is preferable for larger specialized systems.

Formal verification guarantees consistency with the encoded constraint model, not automatic correspondence between that model and all requirements of the real application. Missing, incomplete, or incorrectly specified constraints may lead to formally feasible but practically unacceptable configurations. Consequently, validation of the domain model is a separate requirement from verification of individual outputs.

8. Conclusions

This study experimentally evaluates a procedural architecture for specialized configurational AI systems rather than a universal configuration solver. Within a predefined task family, probabilistic candidate generation was combined with an independently validated deterministic module responsible for feasibility control and conditional exact fallback. The results demonstrate that local neural predictions cannot be assumed to form a globally feasible configuration, whereas architectural separation of generation and verification provides a reliable acceptance boundary.

This study set out to test one architectural principle experimentally: that in configuration problems with hard constraints, the generation of a candidate and the guarantee of its correctness are different functions and should be carried out by different compo-

nents. Where the earlier work proposed the hierarchical imitation architecture at the methodological level, the present paper converts it into a measurable object, with a reproducible corpus of 24,000 problems, a fixed evaluation protocol, pre-specified hypotheses and decision thresholds, and a uniform comparison of deterministic, neural-network, and procedural-hybrid architectures under identical conditions.

Three results carry the argument. Direct neural inference, given complete information about every threshold and every compatibility relation, produced globally feasible configurations on fewer than one fifth of the tasks, and the shortfall is not an artifact of weak models but a consequence of optimizing a per-component objective in a problem whose correctness is a property of the whole. Correction applied to those same candidates without an exact solver recovers a large part of the gap but not the guarantee, which is the empirical form of the distinction between a soft and a hard constraint. Verification with conditional exact fallback closes the gap completely, and does so without any property of the network, because the guarantee is a property of the control flow.

The contribution is therefore as much methodological as empirical. The protocol assembled here—a reproducible generator, fixed training and test partitions, separate in-distribution and out-of-distribution evaluation, operational definitions of every indicator, paired interval estimation, and an explicit decision log—is independent of the particular network and the particular solver used inside it. New architectures, whether differentiable optimization layers, solver-backed language models, or agent-based systems, can be substituted into it and measured on the same scale without changing the experimental conditions.

The boundaries of the claim should be stated as plainly as the claim itself. The evidence comes from a synthetic corpus with a fixed structure and a space of 1024 configurations, in which the deterministic module is exhaustive enumeration. The extended analyses locate three specific limits: the degradation under distribution shift is driven by the density of the incompatibility relations rather than by covariate shift, so it will differ in any setting with a different relational structure; exhaustive enumeration is overtaken by a constraint solver about one order of magnitude above the space studied and ceases to be practicable about three orders above it, although substituting that solver leaves the architecture intact; and the feasibility guarantee holds only to the extent that the verifier's rule set is complete, degrading by roughly two percentage points for each pairwise rule withheld. The paper demonstrates the effect of architectural principles on reliability, reproducibility, and robustness within a fully controlled environment, and does not demonstrate scalability or effectiveness in any applied domain.

Four lines of further work follow directly from these limits. The first is external validity: replicating the protocol on anonymized real configuration data, with expert review of the rules and inter-rater agreement on what counts as an acceptable configuration. The second is scale: replacing exhaustive enumeration with constraint programming or mixed-integer optimization as the fallback and re-measuring the fallback rate, the latency, and the joint optimality on spaces several orders of magnitude larger. The third is the neural component: training it to predict the feasible region, the active constraints, and a calibrated probability of violation rather than only the optimal configuration, so that the fallback becomes selective rather than reflexive, and comparing that against constraint-aware architectures trained on the same corpus. The fourth is the human element, which this study does not address at all: whether an auditable decision log of the kind produced here actually improves trust and decision quality among the people who use such systems is a question for a separate study with human participants.

Future implementations may use AI models to assist in constructing domain-specific configurational modules, translating expert requirements into formal constraints, and

generating verification tests. Such AI-assisted construction does not remove the need for independent validation: before deployment, the resulting module must be tested, approved, versioned, and separated from the probabilistic component whose behavior it constrains.

Author Contributions: Conceptualization, R.Y. and V.R.; methodology, R.Y. and V.R.; software, R.Y.; validation, R.Y. and V.R.; formal analysis, R.Y.; investigation, R.Y.; resources, V.R.; data curation, R.Y.; writing—original draft preparation, R.Y.; writing—review and editing, R.Y. and V.R.; visualization, R.Y.; supervision, V.R.; project administration, V.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable. This study did not involve humans or animals; all numerical instances were produced by a synthetic, seed-controlled generator (Section 4.2) and contain no personally identifiable information.

Informed Consent Statement: Not applicable. This study did not involve human participants.

Data Availability Statement: The complete reproducibility package supporting the reported results, comprising the configuration-instance generator, the model definitions and training scripts, the independent verifier, the exact solver, the hybrid-route protocol, the catalog of the twelve domain scenarios, the fixed random seeds, the analysis code that produces every table and figure, and the resulting JSON results file, is available from the corresponding author on reasonable request. No third-party or restricted data were used; the corpus is generated deterministically by the included code and contains no personal data. The external language-model figures quoted in Table 15 and Figure 4 are taken from the published ConstraintBench results [6] and are not redistributed here.

Acknowledgments: The authors thank the editors and reviewers for their time and constructive comments.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial intelligence
HIM	Hierarchical imitation model
HIM-D	Deterministic hierarchical imitation model (exact solver used in this study)
CSP	Constraint satisfaction problem
MILP	Mixed-integer linear programming
SAT	Boolean satisfiability problem
MLP	Multilayer perceptron
LLM	Large language model
ID	In-distribution
OOD	Out-of-distribution
CI	Confidence interval
AdamW	Adam optimizer with decoupled weight decay
CBC	COIN-OR branch-and-cut solver
CP	Constraint programming
CP-SAT	Constraint-programming solver based on satisfiability search
p.p.	Percentage points
RQ	Research question
SD	Standard deviation

Appendix A. Precise Implementation Parameters

Table A1 records every implementation detail needed to reproduce the neural-network models and the hybrid rule described in Sections 4.3 and 4.4—architecture sizes, parameter

counts, optimizer settings, the decoding and hybrid-switching rules, and the computing environment—so that the results in Section 5 can be regenerated exactly from the code of the reproducibility package described in Section 4.7.

Table A1. Hyperparameters and implementation settings.

Component	Value
Random seed (generation and training)	20260711
Number of tasks	24,000, including 6000 test tasks
Configuration space	5 groups, 4 options per group; $4^5 = 1024$ configurations
Residual MLP architecture	Input 435; hidden layer 384; residual block 384; dropout 0.08; output 5×4
Residual MLP parameters	471,572
Residual MLP training	AdamW; lr 0.001; batch size 512; 5 epochs; weight decay 0.0001
Attention network architecture	5 tokens $\times$ 87 features; projection dimension 32; one attention block; FFN width 64; output 4 per token
Attention network parameters	10,596
Attention network training	AdamW; lr 0.0008; batch size 512; 3 epochs; weight decay 0.0001
Neural-network decoding rule	Per-group argmax over the softmax outputs of each network, with no verification or correction
Hybrid rule—neural step	x_N = argmax of the attention network, then independently checked against all hard constraints
Hybrid rule—fallback step	On any violation, exhaustively search the 1024 configurations and select the feasible option minimizing $J(x)$
Optimality criterion	Feasible and relative gap $\leq 0.1\%$
Computing environment	CPU; Python; NumPy; PyTorch 2.10.0+cpu

Appendix B. Minimum Reproducibility Log Schema

Table A2 specifies the minimum schema for the per-instance log referenced in Sections 2.4 and 4.7: each field's name, data type, and description are given so that the decision path, the verification outcome, and the objective-function values recorded for every task can be audited independently of the code that produced them.

Table A2. Fields of the reproducibility log.

Field	Type	Description
instance_id	string	Unique task identifier
domain	categorical	Education, engineering, or commerce
split	categorical	Training, validation, ID test, or OOD test
system_id	string	Algorithm or neural-network version
seed	integer	Random seed used for generation and the run
predicted_configuration	array	Five selected options

Table A2. *Cont.*

Field	Type	Description
feasible	Boolean	Result of independent verification
violated_constraints	array	List of violated conditions
objective_value	number	Recomputed value of the objective function
optimal_objective	number	Reference (optimal) objective value
objective_gap	number	Relative gap, %
latency_ms	number	Execution time
model_hash	string	Checksum of the model version
scenario_code	string	Code for one of the 12 subject scenarios (E1–E4, I1–I4, or C1–C4)
scenario_name	string	Human-readable name of the configuration task
hybrid_route	categorical	neural_accept or him_fallback

Appendix C. Generation Procedure for a Configuration Instance

Table A3 states the procedure by which a single instance is produced, in the order in which the steps are executed. Symbolic names denote the parameters whose numerical values are given in Table 5 or, where an interval is specific to the attribute distributions, in the configuration file of the reproducibility package (Section 4.7). The procedure is deterministic given the seed and the instance index.

Table A3. Generation procedure for one configuration instance.

Step	Operation
1. Domain	Draw the domain $d \in$ {education, engineering, commerce} so that the three domains are balanced within every split, and select the associated attribute-spread factors.
2. Attributes	For each of the $G = 5$ groups and each of the $O = 4$ options, draw the cost, quality, risk, and preference-mismatch attributes independently and uniformly from the domain-specific intervals; under the OOD setting, the cost and risk intervals are additionally displaced by the shift factors of Table 5.
3. Density	Draw the instance incompatibility density δ uniformly from the interval listed for the split in Table 5.
4. Compatibility	For each unordered pair of distinct groups (g, h), draw an $O \times O$ binary matrix in which each entry is 0 with probability δ and 1 otherwise, and store it symmetrically as $K^{(g,h)}$ and $K^{(h,g)}$.
5. Reference configuration	Draw a reference configuration x^{ref} uniformly over the O^G candidates and set to 1 the compatibility entry of every pair of options it selects, so that x^{ref} is pairwise compatible by construction.
6. Thresholds	Compute the cost, quality, and risk sums of x^{ref} and set B, $Q_{\min}$, and $R_{\max}$ as those sums multiplied by factors drawn uniformly from the intervals listed for the split in Table 5. The instance is feasible because x^{ref} satisfies every condition of Equation (1).

Table A3. *Cont.*

Step	Operation
7. Exhaustive labeling	Evaluate all O^G candidates, record the feasible set, the minimum objective value, and the optimal configuration, and regenerate the instance if the feasible set is empty. No instance in the corpus required regeneration.
8. Scenario tag	Assign one of the four scenario codes of the drawn domain (Table 4) for reporting purposes only; the tag does not affect any numerical value.
9. Features	Assemble the five group tokens of 87 features described in Section 4.4, using normalization constants that are fixed in advance and identical for every split.

Note: The partitions differ only in the intervals used in steps 2, 3, and 6. The training, validation, and ID-test partitions take the ID levels listed in Table 5 and the OOD-test partition takes the OOD levels. Steps 1, 4, 5, 7, 8, and 9 are identical for all four partitions.

References

1. Yavich, R.; Malev, S.; Volinsky, I.; Rotkin, V. Configurable Intelligent Design Based on Hierarchical Imitation Models. *Appl. Sci.* **2023**, *13*, 7602. [CrossRef]
2. Mittal, S.; Frayman, F. Towards a Generic Model of Configuration Tasks. In Proceedings of the 11th International Joint Conference on Artificial Intelligence, Detroit, MI, USA, 20–25 August 1989; pp. 1395–1401. Available online: https://www.ijcai.org/Proceedings/89-2/Papers/087.pdf (accessed on 15 September 2026).
3. Sabin, D.; Weigel, R. Product Configuration Frameworks—A Survey. *IEEE Intell. Syst. Their Appl.* **1998**, *13*, 42–49. [CrossRef]
4. Bengio, Y.; Lodi, A.; Prouvost, A. Machine Learning for Combinatorial Optimization: A Methodological Tour d'Horizon. *Eur. J. Oper. Res.* **2021**, *290*, 405–421. [CrossRef]
5. Cappart, Q.; Chételat, D.; Khalil, E.B.; Lodi, A.; Morris, C.; Veličković, P. Combinatorial Optimization and Reasoning with Graph Neural Networks. *J. Mach. Learn. Res.* **2023**, *24*, 1–61. Available online: https://www.jmlr.org/papers/v24/21-0449.html (accessed on 15 September 2026).
6. Tso, J.; Schmittou, P.; Huynh, Q.; Hutchins, J. ConstraintBench: Benchmarking LLM Constraint Reasoning on Direct Optimization. *arXiv* **2026**, arXiv:2602.22465. [CrossRef]
7. Chen, J.; Wei, Z.; Ren, Z.; Li, Z.; Zhang, J. LR^{2}Bench: Evaluating Long-Chain Reflective Reasoning Capabilities of Large Language Models via Constraint Satisfaction Problems. In *Findings of the Association for Computational Linguistics: ACL 2025*; Association for Computational Linguistics: Vienna, Austria, 2025; pp. 6006–6032. [CrossRef]
8. Yang, Z.; Wang, Y.; Huang, Y.; Guo, Z.; Shi, W.; Han, X.; Feng, L.; Song, L.; Liang, X.; Tang, J. OptiBench Meets ReSocratic: Measure and Improve LLMs for Optimization Modeling. In Proceedings of the International Conference on Learning Representations (ICLR 2025), Singapore, 24–28 April 2025.
9. AhmadiTeshnizi, A.; Gao, W.; Udell, M. OptiMUS: Scalable Optimization Modeling with (MI)LP Solvers and Large Language Models. In *Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria, 21–27 July 2024*; PMLR: Cambridge, MA, USA, 2024; Volume 235, pp. 577–596.
10. Perron, L.; Didier, F.; Gay, S. The CP-SAT-LP Solver. In *Proceedings of the 29th International Conference on Principles and Practice of Constraint Programming (CP 2023), Toronto, Canada, 27–31 August 2023*; LIPIcs: Wadern, Germany, 2023; Volume 280, pp. 3:1–3:2. [CrossRef]
11. Gorishniy, Y.; Rubachev, I.; Khrulkov, V.; Babenko, A. Revisiting Deep Learning Models for Tabular Data. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 18932–18943.
12. Huang, X.; Khetan, A.; Cvitkovic, M.; Karnin, Z. TabTransformer: Tabular Data Modeling Using Contextual Embeddings. *arXiv* **2020**, arXiv:2012.06678. [CrossRef]
13. Amos, B.; Kolter, J.Z. OptNet: Differentiable Optimization as a Layer in Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017*; PMLR: Cambridge, MA, USA, 2017; Volume 70, pp. 136–145.
14. Vlastelica, M.; Paulus, A.; Musil, V.; Martius, G.; Rolínek, M. Differentiation of Blackbox Combinatorial Solvers. In Proceedings of the International Conference on Learning Representations (ICLR 2020), Virtual, 26 April–1 May 2020.
15. Paulus, A.; Rolínek, M.; Musil, V.; Amos, B.; Martius, G. CombOptNet: Fit the Right NP-Hard Problem by Learning Integer Programming Constraints. In *Proceedings of the 38th International Conference on Machine Learning, Virtual, 18–24 July 2021*; PMLR: Cambridge, MA, USA, 2021; Volume 139, pp. 8443–8453.

16. Liang, E.; Chen, M.; Low, S.H. Low Complexity Homeomorphic Projection to Ensure Neural-Network Solution Feasibility for Optimization over (Non-)Convex Set. In *Proceedings of the 40th International Conference on Machine Learning, Honolulu, HI, USA, 23–29 July 2023*; PMLR: Cambridge, MA, USA, 2023; Volume 202, pp. 20623–20649. Available online: https://proceedings.mlr.press/v202/liang23a.html (accessed on 15 September 2026).
17. Wang, R.; Zhang, Y.; Guo, Z.; Chen, T.; Yang, X.; Yan, J. LinSATNet: The Positive Linear Satisfiability Neural Networks. In *Proceedings of the 40th International Conference on Machine Learning, Honolulu, HI, USA, 23–29 July 2023*; PMLR: Cambridge, MA, USA, 2023; Volume 202, pp. 36605–36625.
18. Min, Y.; Azizan, N. HardNet: Hard-Constrained Neural Networks with Universal Approximation Guarantees. *arXiv* **2024**, arXiv:2410.10807. [CrossRef]
19. Rudin, C. Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. *Nat. Mach. Intell.* **2019**, *1*, 206–215. [CrossRef] [PubMed]
20. Sculley, D.; Holt, G.; Golovin, D.; Davydov, E.; Phillips, T.; Ebner, D.; Chaudhary, V.; Young, M.; Crespo, J.-F.; Dennison, D. Hidden Technical Debt in Machine Learning Systems. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 2503–2511. Available online: https://proceedings.neurips.cc/paper_files/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf (accessed on 15 September 2026).
21. Manhaeve, R.; Dumančić, S.; Kimmig, A.; Demeester, T.; De Raedt, L. DeepProbLog: Neural Probabilistic Logic Programming. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 3749–3759.
22. Wang, P.-W.; Donti, P.L.; Wilder, B.; Kolter, J.Z. SATNet: Bridging Deep Learning and Logical Reasoning Using a Differentiable Satisfiability Solver. In *Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019*; PMLR: Cambridge, MA, USA, 2019; Volume 97, pp. 6545–6554.
23. Garcez, A.d.; Lamb, L.C. Neurosymbolic AI: The 3rd Wave. *Artif. Intell. Rev.* **2023**, *56*, 12387–12406. [CrossRef]
24. Kambhampati, S.; Valmeekam, K.; Guan, L.; Verma, M.; Stechly, K.; Bhambri, S.; Saldyt, L.P.; Murthy, A.B. Position: LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks. In *Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria, 21–27 July 2024*; PMLR: Cambridge, MA, USA, 2024; Volume 235, pp. 22895–22907.
25. Mitchell, M.; Wu, S.; Zaldivar, A.; Barnes, P.; Vasserman, L.; Hutchinson, B.; Spitzer, E.; Raji, I.D.; Gebru, T. Model Cards for Model Reporting. In Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT* 2019), Atlanta, GA, USA, 29–31 January 2019; pp. 220–229. [CrossRef]
26. Lakens, D. Equivalence Tests: A Practical Primer for t Tests, Correlations, and Meta-Analyses. *Soc. Psychol. Personal. Sci.* **2017**, *8*, 355–362. [CrossRef] [PubMed]
27. Forrest, J.; Lougee-Heimer, R. CBC User Guide. In *Emerging Theory, Methods, and Applications*; INFORMS TutORials in Operations Research; INFORMS: Catonsville, MD, USA, 2005; pp. 257–277. [CrossRef]
28. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 8024–8035.
29. Wilson, E.B. Probable Inference, the Law of Succession, and Statistical Inference. *J. Am. Stat. Assoc.* **1927**, *22*, 209–212. [CrossRef]
30. Efron, B.; Tibshirani, R.J. *An Introduction to the Bootstrap*; Chapman and Hall/CRC: New York, NY, USA, 1994. [CrossRef]
31. Benjamini, Y.; Hochberg, Y. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *J. R. Stat. Soc. Ser. B* **1995**, *57*, 289–300. [CrossRef]
32. Pineau, J.; Vincent-Lamarre, P.; Sinha, K.; Larivière, V.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; Larochelle, H. Improving Reproducibility in Machine Learning Research: A Report from the NeurIPS 2019 Reproducibility Program. *J. Mach. Learn. Res.* **2021**, *22*, 1–20. Available online: https://www.jmlr.org/papers/volume22/20-303/20-303.pdf (accessed on 15 September 2026).
33. Bertsimas, D.; Kallus, N. From Predictive to Prescriptive Analytics. *Manag. Sci.* **2020**, *66*, 1025–1044. [CrossRef]
34. Sargent, R.G. Verification and Validation of Simulation Models. *J. Simul.* **2013**, *7*, 12–24. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.