REQUIREMENT-ALIGNED LLM MODELING FOR HOT ROLLING SCHEDULING IN STEEL MANUFACTURING

Anonymous authors
Paper under double-blind review

ABSTRACT

Steel manufacturing's high energy use and carbon emissions make effective production scheduling—particularly hot rolling scheduling—important for efficiency and sustainability. However, complex, dynamic, and highly specific requirements of steel production limit the flexibility of conventional optimization methods, leaving scheduling dependent on local human expertise. Large language models (LLMs) can construct and adapt models for the hot rolling scheduling problem (HRSP) from natural-language requirements, but incomplete or inconsistent formulations may be infeasible or yield schedules that violate the target production constraints. To address these limitations, we propose a requirement-aligned LLM-based framework for constructing and solving multi-objective HRSP models from natural-language descriptions and instance data. A knowledge base organizes expert formulations into reusable decision, constraint, and objective components with production semantics and dependency records. Component Selection matches components to target requirements through semantic comparison, evidence verification, and coverage checks that recover omissions. Component Composition aligns decision definitions and dependencies to assemble the selected components into a consistent constraint system and objective vector. Model and Solution Verification checks component dependencies and data references before solving, then independently validates target feasibility, objective values, and Pareto non-dominance. The framework achieves F1 scores of 0.952 for component selection and 0.966 for adaptation, with all returned schedules passing independent checks of the target constraints. Our code is available at `https://anonymous.4open.science/r/llm4hrsp`.

1 INTRODUCTION

Steel manufacturing's high energy use and carbon emissions make efficient production scheduling important for sustainability. Hot rolling is a key production stage in steel manufacturing, and the hot rolling scheduling problem (HRSP) coordinates order selection, rolling-round assignment, and slab sequencing (Tang et al., 2000; Chen et al., 2023). Effective scheduling requires models that accurately reflect production requirements, which vary across production lines and over time. Differences in equipment, orders, delivery commitments, operating priorities, uncertainty, and coordination with adjacent stages give rise to HRSP variants with different decisions, constraints, or objectives (Zhang et al., 2022; Jiang et al., 2025b). Existing methods develop formulations and exact, heuristic, or learning-based solvers for particular settings (Tan et al., 2019; Pan et al., 2019; Zhang et al., 2025). Their adaptability is limited because changing requirements may require structural reformulation, not just parameter updates.

Adapting scheduling models therefore still relies on planners' and operators' plant-specific knowledge, together with modeling experts who translate that knowledge into mathematical expressions. The resulting challenge is to support diverse HRSP variants without requiring expert reformulation for every new production setting.

Large language models (LLMs) offer a promising way to construct and adapt HRSP optimization models from natural-language production requirements. ORLM and LLMOPT generate mathematical formulations and solver code (Huang et al., 2025; Jiang et al., 2025a), while LLaMoCo focuses on optimization code generation (Ma et al., 2026). However, automatic generation does not by itself

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

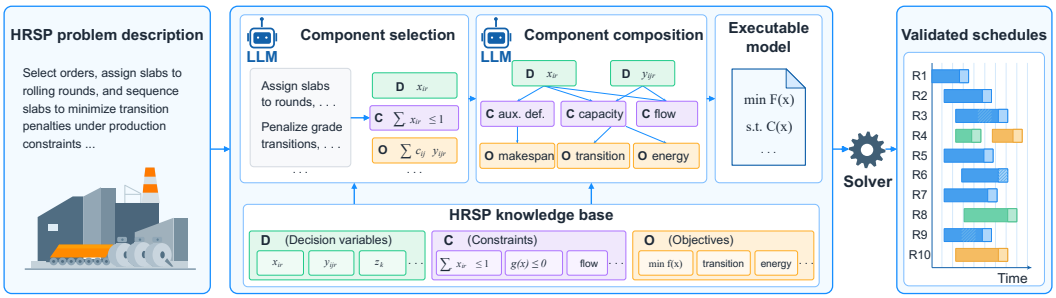


Figure 1: Overview of the proposed framework. An HRSP knowledge base supplies decision (D), constraint (C), and objective (O) components for requirement-grounded selection and dependency-consistent composition. The executable model is solved, and returned schedules are independently checked against the target production requirements. The schedule shown is illustrative.

ensure that the resulting formulation is complete or consistent with the target production requirements. In HRSP, even a localized requirement change may alter the model structure, for example by converting a preference into a hard restriction, introducing an additional decision, or requiring a new dependency between existing decisions. More importantly, a solver-executable model is not necessarily requirement-correct: a generated formulation may omit a production rule and still return solver-feasible solutions. Checking feasibility only against the generated constraints cannot reveal such an omission. Returned schedules, objective values, and Pareto non-dominance should therefore be evaluated independently against the target production requirements rather than only against the generated formulation. Therefore, reliable LLM-based HRSP modeling requires not only executable model construction, but also checks of requirement coverage, formulation consistency, and solution validity against the original production requirements.

To address these challenges, we propose a requirement-aligned LLM-based framework for modeling and solving HRSP variants from a natural-language description and instance data (Figure 1). The framework structures expert formulations into a knowledge base of reusable decision, constraint, and objective components with production semantics and explicit dependency records. *Component Selection* identifies target-supported components, verifies their supporting evidence, and checks requirement coverage to recover omissions. *Component Composition* aligns shared decision definitions, completes required decision dependencies, and assembles a consistent constraint system and objective vector. *Model and Solution Verification* checks component dependencies, symbol definitions, and data references before solving, returns missing components or inconsistent connections for correction, and then independently evaluates returned schedules against the target production constraints, objective definitions, and Pareto non-dominance conditions. These stages distinguish requirement alignment, formulation consistency, and target-level solution validity. Evaluation shows more accurate component selection and localized adaptation than the baselines, while producing target-feasible schedules with competitive Pareto and operational quality.

Our contributions are summarized as follows:

- We formulate LLM-based HRSP modeling as a requirement-alignment problem that distinguishes three aspects of correctness: whether the required modeling components are selected, whether the selected components form a consistent executable formulation, and whether the resulting schedules satisfy the target production requirements.
- We develop a component-based framework that organizes expert HRSP formulations into reusable decision, constraint, and objective components with explicit production semantics and dependency information. This structure supports localized adaptation to changing production requirements while preserving unaffected parts of the reference formulation.
- We provide an evaluation that separates solver feasibility, target feasibility, and solution quality across held-out HRSP cases and controlled requirement changes. The results reveal failures that remain hidden under solver feasibility alone, while the proposed framework produces schedules that pass all target-level checks.

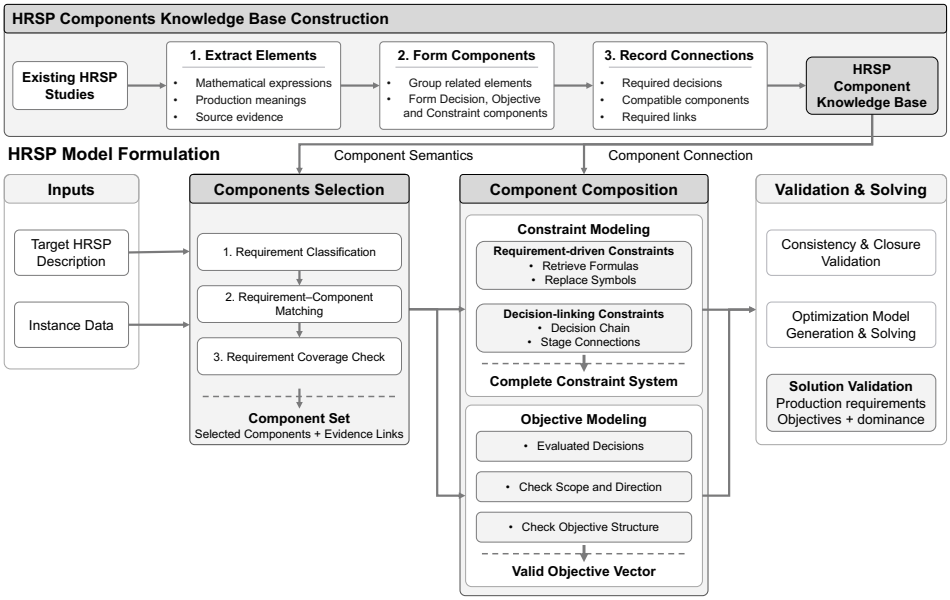


Figure 2: Overview of the proposed framework. Component selection interprets target requirements; component composition constructs the constraint system and objective vector; model and solution checks verify the formulation and its schedules.

2 Problem Setup

An HRSP schedule selects orders, assigns their slabs to rolling rounds, and sequences the slabs within each round. A rolling round is a group processed consecutively between roll changes. Depending on the setting, additional decisions concern processing times, furnace allocation, or arriving orders. Constraints can restrict round capacity, transitions between adjacent slabs, and the use of time or resources. Objectives evaluate quantities such as unselected orders, transition penalties, and energy consumption. Appendix A gives the full production setting and mathematical definitions.

The input consists of a natural-language target description D and instance data I. The description specifies the production requirements and preferences; the data provide the corresponding objects, attributes, and numerical parameters. The generated formulation is

$$\widehat{\mathcal{M}}_{D,I} = \big(\widehat{\mathcal{X}}_{D,I}, \boldsymbol{C}(\boldsymbol{x}), \boldsymbol{F}(\boldsymbol{x})\big), \tag{1}$$

where $\widehat{\mathcal{X}}_{D,I}$ defines the decision variables and domains, $\boldsymbol{C}$ the constraints, and $\boldsymbol{F}$ the objective vector. For multiple independent criteria, solving produces a finite set of non-dominated schedules.

Success requires three forms of agreement: selected components must express the target requirements; their variables, scopes, and dependencies must form a consistent executable model; and returned schedules must satisfy the target hard constraints under the intended objective definitions. Changes in production requirements may affect the decisions, constraints, or objectives of the target HRSP. The generated model should reflect these changes while remaining consistent with the unchanged requirements.

3 Method

The framework in Figure 2 constructs an HRSP optimization model from D and I using reusable components and connection records in $\mathcal{K}$. Selection identifies target-supported components together with their evidence and dependencies; composition connects their expressions under shared decision definitions. This separation distinguishes component applicability from consistency among selected expressions. Model verification checks the assembled formulation before solving, and independent solution checks assess returned schedules against target production rules.

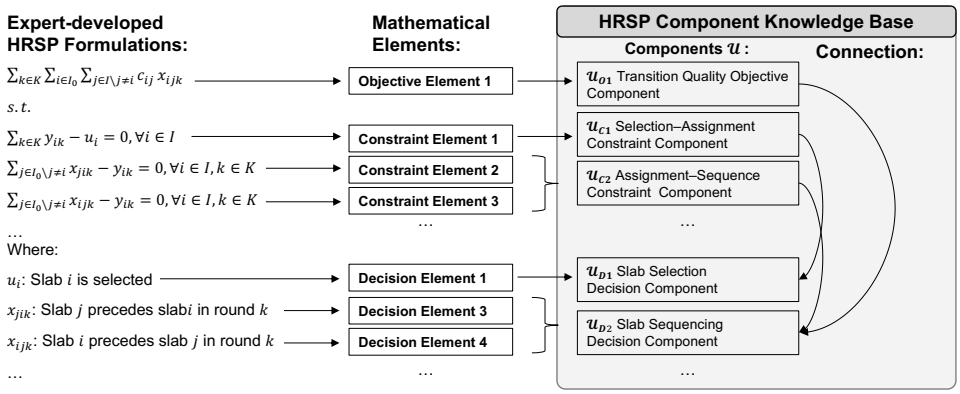


Figure 3: Elements, components, and connections extracted from an expert formulation. Expressions implementing the same production decision or rule remain together. The example uses slabs I, rounds K, selection variables u_i, assignment variables y_{ik}, and sequencing variables x_{ijk}.

3.1 HRSP COMPONENT KNOWLEDGE BASE CONSTRUCTION

The knowledge base $\mathcal{K}$ organizes expert formulations into reusable components and connection records for subsequent selection and composition. Components keep related mathematical expressions together because a single expression may not fully represent a production requirement. Each expression is an *element*; those implementing one decision, rule, or criterion form a *component*, the smallest unit selected for model construction. For component u, $\mathcal{E}_u = \{e_{u1}, \ldots, e_{un_u}\}$, with $n_u \geq 1$ (Figure 3).

Component records support requirement matching and formula instantiation by retaining production semantics and required mathematical definitions. Semantic fields include role, objects, scope, conditions, and restriction or optimization direction; objective records also retain evaluated decisions and preference structure. Variable domains, indices, required data, and source formulations support instantiation and interpretation. Notation is standardized, but components are merged only when their production meaning and mathematical representation are equivalent.

Connection records support composition by specifying dependencies through $\mathcal{G}_\mathcal{K} = (\mathcal{U}, E_\mathcal{K})$, where $\mathcal{U}$ is the component set. Constraint–decision and objective–decision relations identify required decisions; decision–decision relations specify shared objects and structural links. Variable correspondences, index domains, and application conditions determine whether a connection is reusable. Together, component semantics and connection records support target-specific selection and subsequent assembly.

3.2 COMPONENT SELECTION

Using $\mathcal{K}$, Component Selection identifies components supported by D and instantiable with I for subsequent composition. Classification and retrieval produce candidates; matching and evidence verification determine acceptance; coverage checking revisits omissions. The output retains evidence links and dependencies. Appendix B.2 details these operations.

Requirement Classification. The LLM organizes statements in D into decision, constraint, and objective requirements to guide candidate retrieval. These roles distinguish choices to make, restrictions to satisfy, and criteria to optimize. Using I to distinguish decisions from given quantities, it separates mixed statements while preserving scope and conditions, forming $\mathcal{R}(D, I)$. This distinction matters because the same quantity can appear in a restriction or a preference. Retrieval-augmented generation (RAG) then searches the corresponding categories in $\mathcal{K}$, forming $\mathcal{U}_{\text{cand}}$ for matching.

Requirement–Component Matching. Matching evaluates whether each retrieved candidate implements its associated target requirement. For each pair (r, u), the LLM compares production

meaning, objects, stages, index ranges, and application conditions. It distinguishes hard restrictions from penalties, bound directions, and minimization from maximization. Objective candidates must also match evaluated decisions, calculation scope, and independent or combined treatment of criteria; related wording alone is insufficient.

Evidence verification determines whether a semantically matched component is justified by the target problem. Source formulations define component meaning; D supports applicability, while I provides the required data directly or through explicit derivations. Acceptance requires both semantic agreement and target support. The retained evidence links each match to supporting statements and data references. Conflicting or uncertain candidates remain in the initially rejected pool with rejection reasons, enabling reassessment when a requirement is uncovered.

Requirement Coverage Check. This step checks whether the accepted matches collectively cover the parsed requirements, complementing the preceding pairwise judgments. With $\mathcal{A}$ denoting accepted, evidence-supported pairs, uncovered requirements are

$$\mathcal{R}_{\text{miss}} = \mathcal{R}(D, I) \setminus \{r : \exists u,\ (r, u) \in \mathcal{A}\}. \tag{2}$$

For each $r \in \mathcal{R}_{\text{miss}}$, the LLM reassesses initially rejected candidates using their semantics and recorded rejection reasons. Recovered matches enter $\mathcal{A}$ only after passing the same semantic and target-evidence checks. Coverage is reevaluated until all parsed requirements are covered or no further match can be recovered. Unresolved requirements prevent model acceptance and are reported without introducing unsupported assumptions.

The output of selection is

$$\widehat{\mathcal{U}} = \{u : \exists r,\ (r, u) \in \mathcal{A}\}, \tag{3}$$

accompanied by evidence links justifying inclusion and dependency records identifying the decisions required for composition.

3.3 Component composition

Component composition transforms the selected components and their dependency records into a constraint system $\boldsymbol{C}(\boldsymbol{x})$ and objective vector $\boldsymbol{F}(\boldsymbol{x})$. It first aligns shared decision definitions, then assembles objective and constraint expressions using $\mathcal{K}$ and I. The resulting formulation passes to the subsequent module for verification. Appendix B.3 details the construction.

Recorded component connection aligns component expressions with the selected decisions before objective and constraint construction. Dependency records identify required decisions. A recorded connection applies only when its participating components, shared objects, and conditions match the target. Variable references and index domains are aligned by production meaning rather than symbol names, establishing shared definitions for both modeling branches.

Objective Modeling. This step constructs the objective vector by connecting selected criterion formulas to the shared decisions and data in I. It matches each formula's required inputs to these definitions and reuses applicable connection formulas. Otherwise, the LLM derives logical or aggregation expressions from confirmed decision meanings and aligned indices. Each auxiliary variable receives a domain and defining constraints and cannot replace a missing production decision. Substituting the connected inputs preserves calculation scope and optimization direction, yielding

$$\boldsymbol{F}(\boldsymbol{x}) = \big(f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})\big). \tag{4}$$

Here, m counts the criteria and $\boldsymbol{x}$ includes auxiliary variables, whose defining constraints enter $\boldsymbol{C}_{\text{link}}$ below. Expressions defining one criterion remain together; scalar, Pareto, or lexicographic organization follows D.

Constraint Modeling. This step constructs $\boldsymbol{C}(\boldsymbol{x})$ from target production restrictions and the links required by the selected decision representations. Requirement-driven constraints $\boldsymbol{C}_{\text{req}}$ instantiate production rules by mapping their objects, stages, and parameters to shared variables, indices, and data. Bounds, directions, and activation conditions are preserved; calculated inputs use the auxiliary definitions introduced above. Decision-linking constraints reconcile the selected decision stages. In Figure 3, $\sum_{k \in K} y_{ik} = u_i$ links slab selection to round assignment. Missing relations are supplied

by compatible records or derived from selected decision meanings and index domains, even when not explicit in D. Only target-required stages are linked, without additional production restrictions. These links and auxiliary definitions form $\boldsymbol{C}_{\text{link}}$, completing the constraint system:

$$\boldsymbol{C}(\boldsymbol{x}) = \big(\boldsymbol{C}_{\text{req}}(\boldsymbol{x}), \boldsymbol{C}_{\text{link}}(\boldsymbol{x})\big). \tag{5}$$

3.4 Model and solution verification

Model and Solution Verification checks the assembled formulation before solving and subsequently validates returned schedules against target requirements. Model checks require every component's decision dependencies and expression definitions to be resolved. Connections are justified by compatible records or derivations from decision meanings and examined for isolated components or incompatible relations. Alternative records are distinguished by scope and conditions. Let $\Sigma_{C,F}$ contain symbols referenced by constraints and objectives, and $\Sigma_{X,I}$ those defined by the decision structure and instance data. Symbol coverage requires

$$\Sigma_{C,F} \subseteq \Sigma_{X,I}. \tag{6}$$

Beyond symbol coverage, reference checks require agreement with production objects, domains, and index ranges; calculated quantities must resolve to declared decisions and available data. Executable verification then constructs the solver model and checks instance-field access and values. Failures are recorded: missing required components return to selection; incorrect connections, definitions, or data references return to composition.

Independent solution checks evaluate returned schedules against target hard constraints rather than only the generated formulation, since solver feasibility cannot reveal an omitted target rule. Objectives are reevaluated from target definitions before non-dominated filtering. These checks support the reported schedules but do not establish equivalence between the generated and expert feasible regions. Appendix B details the records and checks.

4 Experiments

We evaluate the framework through three research questions, corresponding to the three results subsections: (1) Can it select the required components and adapt to requirement changes while preserving unaffected components? (2) Do the composed models yield target-feasible solutions with competitive Pareto quality? (3) How do the selected schedules perform on production indicators across HRSP variants?

4.1 Experimental settings

Dataset and separation. HRSP variants extend the standard problem in decision scope, constraints, objectives, or uncertainty representation. The 25 studies are grouped into these four extension categories and Standard HRSP. One distinct study is randomly held out from each category, yielding five test cases; the remaining 20 formulations supply the knowledge base. The original test formulations define the required components and target constraints, and all information from these studies is excluded from knowledge-base construction. Categories may overlap; their distribution appears in Appendix C.

Knowledge-base scale. The 20 training formulations yield 336 components: 82 decision, 202 constraint, and 52 objective components, with 526 recorded connections (Figure 4). Each component groups one or more mathematical elements representing a decision, production rule, or performance criterion; the components collectively contain 593 elements. These counts refer to stored component and connection records, retaining different mathematical representations of the same function as separate components.

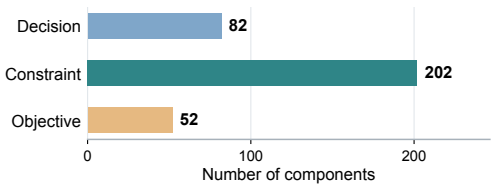


Figure 4: Component counts by type across the 20 training formulations.

Baselines and computation. We compare ORLM (Huang et al., 2025), LLMOPT (Jiang et al., 2025a), DeepSeek, DeepSeek + RAG, and *Reference*, the best result in the literature. DeepSeek +

Table 1: Component selection and adaptation on five cases and 15 requirement changes.

	Initial selection			Requirement changes			
Method	$P \uparrow$	$R \uparrow$	F1 $\uparrow$	$P_\Delta \uparrow$	$R_\Delta \uparrow$	$F1_\Delta \uparrow$	$S \uparrow$
ORLM[a,b]	0.750[a]	0.261[a]	0.357[a]	0.000[b]	0.000[b]	0.000[b]	0.600[b]
LLMOPT	0.878	0.442	0.536	0.036[c]	0.133[c]	0.057[c]	0.676[c]
DeepSeek	0.945	0.901	0.918	0.185[c]	0.667[c]	0.290[c]	0.865[c]
DeepSeek + RAG	0.914	0.907	0.908	0.250	**0.933**	0.394	0.878
Ours	**0.994**	**0.920**	**0.952**	**1.000**	**0.933**	**0.966**	**1.000**

Complete formulations: [a]2/5 initial cases; [b]4/15 requirement-change tests; [c]14/15 requirement-change tests for each method. Incomplete outputs remain in the evaluation. Best values before rounding, including ties, are bold.

RAG retrieves from the full texts of the 20 training studies; ORLM and LLMOPT use their released configurations. DeepSeek, DeepSeek + RAG, and our method share the same backbone and generation settings, so their comparison isolates differences in the modeling framework. Comparisons with ORLM and LLMOPT are at the system level because those methods use their released configurations. Reference uses the original reported results when a test study provides complete public results. Otherwise, its original formulation provides the reference model for Gurobi-based Pareto search. For Gurobi-based comparisons, we use a common Pareto-search procedure and a three-hour total budget per model. Published results retain their original computational settings; additional search settings are provided in Appendix C.

4.2 Component Selection and Adaptation

We evaluate initial selection on five cases and adaptation on 15 single-requirement changes. Three edits per case add, remove, or modify one decision, constraint, or objective requirement, leaving other requirements and instance data unchanged. Precision (P) is the proportion of selected components required by the target; recall (R) is the proportion of required components recovered; F1 is their harmonic mean. For adaptation, P_Δ, R_Δ, and $F1_\Delta$ compare predicted component additions and removals with the expected changes. Change precision measures how many proposed changes are required; change recall measures how many required changes are recovered. A component modification is represented by removing the original component and adding its replacement, following the reference annotations. Stability (S) is the fraction of unaffected reference components selected in both the original and modified descriptions. Scores are averaged over the five cases or 15 edits. Full definitions and component-type results are in Appendices C and D.

Table 1 compares component selection, adaptation, and stability. Our method achieves the highest initial-selection and change F1 while preserving all unaffected reference components in the tested edits. Its larger advantage under requirement changes addresses the need to adapt formulations without unnecessary changes elsewhere. DeepSeek + RAG matches our change recall but makes many unsupported revisions, indicating that the main distinction is excluding unnecessary changes rather than recovering more required updates. DeepSeek + RAG also slightly improves initial recall over DeepSeek while reducing precision, showing that related formulations alone do not establish component applicability. These findings are consistent with requirement-grounded reuse, although some required components remain unselected.

Figure 5 reports initial-selection and change F1 for decision, constraint, and objective components. Our method achieves the highest initial-selection F1 for decisions and constraints and correctly captures all tested objective and constraint updates. The contrast is particularly informative for objectives: DeepSeek and DeepSeek + RAG recognize all initial objectives but do not consistently identify the corresponding changes after requirement edits. Recognizing a criterion is therefore distinct from determining whether it should be retained, replaced, or removed. Our remaining decision-update omissions identify decision coverage as an unresolved issue. Because constraints and objectives depend on selected decisions, these omissions also motivate evaluating the assembled model rather than relying on component-level accuracy alone.

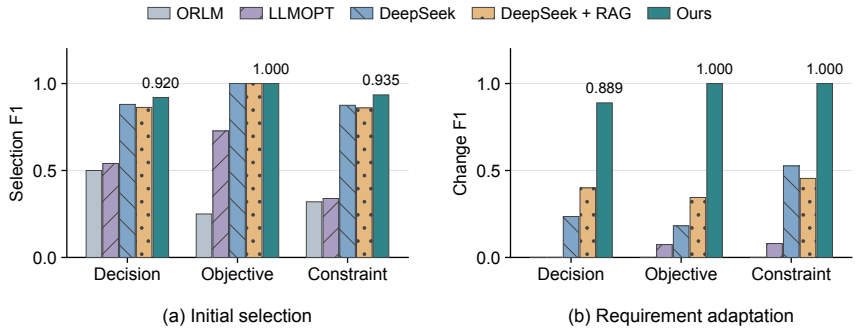


Figure 5: Component-type F1 for (a) initial selection and (b) adaptation to requirement changes.

Table 2: Target feasibility and Pareto quality across five HRSP cases.

	Standard			**Decision scope**			**Objective**			**Constraint**			**Uncertainty**		
Method	$T\uparrow$	HV $\uparrow$	IGD $\downarrow$	$T\uparrow$	HV $\uparrow$	IGD $\downarrow$	$T\uparrow$	HV $\uparrow$	IGD $\downarrow$	$T\uparrow$	HV $\uparrow$	IGD $\downarrow$	$T\uparrow$	HV $\uparrow$	IGD $\downarrow$
ORLM	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
LLMOPT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reference	**100.00**	0.777	**0.000**	**100.00**	1.114	**0.000**	**100.00**	**1.004**	**0.000**	**100.00**	**0.846**	**0.000**	**100.00**	0.636	**0.000**
DeepSeek	0.00	–	–	**100.00**	0.682	0.289	0.00	–	–	26.63	0.663^a	0.372^a	0.00	–	–
DeepSeek + RAG	0.00	–	–	0.00	–	–	0.00	–	–	36.61	0.616^a	0.399^a	0.00	–	–
Ours	**100.00**	**0.777**	0.001	**100.00**	**1.159**	0.038	**100.00**	0.909	0.096	**100.00**	0.832	0.085	**100.00**	**1.013**	0.361

–: no solver-feasible output or no target-feasible solution. Best values before rounding are bold. Reference IGD is zero by construction; its front is not assumed globally optimal. ^{a}Metrics use only the target-feasible subsets.

4.3 Model Validity and Pareto Quality

We compare solution quality from an optimization perspective. Solver feasibility checks a solution against the generated model; target feasibility (T) is the percentage of returned schedules satisfying independently checked target constraints. A rate of 100% means all returned schedules pass the target checker; an intermediate rate means partial feasibility, and 0% means none passes. We evaluate target-feasible, non-dominated solutions using hypervolume (HV) and inverted generational distance (IGD), with common objective normalization within each case. Definitions and case analyses appear in Appendices C and E.

Table 2 reports target feasibility and Pareto indicators across the five cases. Our method is the only evaluated LLM-based approach whose returned schedules satisfy all checked target constraints in every case, demonstrating its reliability advantage in these tests. By contrast, DeepSeek and DeepSeek + RAG frequently produce target-infeasible schedules despite successful solver execution, while ORLM and LLMOPT fail to obtain solver-feasible formulations. Together with the component-selection results, this exposes a gap between recognizing most required components and obtaining valid production schedules. Using raw paper texts in DeepSeek + RAG does not close this gap. The results support the combined modeling and verification approach and show why target checks are needed beyond solver execution. This evidence concerns returned schedules, not equivalence of the complete feasible regions.

The HV and IGD columns in Table 2 compare the quality of target-feasible solution sets. Our method closely matches Reference in the standard case and achieves higher HV and lower IGD than the available target-feasible fronts from the DeepSeek baselines in the decision-scope and constraint cases. Its advantage therefore extends beyond feasibility to the quality of the returned trade-offs. Comparison with Reference is more mixed: lower HV in the objective and constraint extensions leaves room to improve trade-off coverage despite full target feasibility. In the uncertainty case, higher HV coexists with nonzero IGD, so greater dominated volume does not establish uniform improvement over Reference. These results support competitive optimization outcomes under the reported computational settings, while motivating a separate examination of schedule-level production benefits.

Table 3: Operational performance of representative target-feasible schedules across five HRSP cases.

	Standard				Decision scope				Objective				Constraint				Uncertainty			
Method	Loss ↓	R ↓	RRU ↑	P ↓	Loss ↓	R ↓	RRU ↑	P ↓	Loss[a] ↓	R ↓	RRU ↑	P ↓	Loss ↓	R ↓	RRU ↑	P ↓	Loss ↓	R ↓	RRU ↑	P ↓
ORLM	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
LLMOPT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reference	**38.00**	**6**	**10.01**	**6.10**	**571.00**	**4**	47.86	**37475.90**	8300.00	**5**	**97.39**	3555.00	59.25	**4**	97.03	**259.50**	N/A	5	77.00	**5.12**
DeepSeek	–	–	–	–	573.50	**4**	49.10	55988.25	–	–	–	–	**59.10**[b]	**4**[b]	**98.55**[b]	274.50[b]	–	–	–	–
DeepSeek + RAG	–	–	–	–	–	–	–	–	–	–	–	–	84.53[b]	**4**[b]	92.70[b]	332.50[b]	–	–	–	–
Ours	**38.00**	**6**	**10.01**	**6.10**	578.00	**4**	**49.44**	46919.56	**1378.00**	**5**	95.87	**3335.00**	68.53	**4**	95.29	273.50	N/A	**4**	**96.25**	5.38

–: no target-feasible schedule; N/A: inapplicable indicator. Best values before rounding are bold. [a]Loss denotes total unscheduled slab length. [b]DeepSeek and DeepSeek + RAG are evaluated only on their target-feasible subsets.

4.4 Operational Performance

We assess the production performance of the selected schedules across the three HRSP decision stages. Loss measures unselected-order loss, except in the objective-extension case, where it is total unscheduled slab length. Rolling-round assignment is evaluated by the number of active rounds (R) and their mean capacity utilization (RRU, %). The transition penalty (P) sums penalties between consecutive slabs and serves as an indicator of roll wear. Round use is interpreted alongside Loss for a comparable workload; each case retains its own loss and penalty definitions. Loss and penalties are compared within cases because their numerical scales and physical meanings differ across variants. Some variants omit a decision stage, so its indicators are inapplicable.

All operational indicators for a method come from one representative target-feasible, non-dominated schedule. Within each case, the common ideal point contains the best value of each objective across all compared methods' valid solutions. Using common normalization and equal objective weights, we choose the schedule with the smallest Euclidean distance to this point. For new solver runs, selection uses only solutions found within the shared three-hour budget; published results follow the Reference policy above. This common rule prevents selecting a different favorable schedule for each indicator. Detailed definitions appear in Appendix C.

Table 3 reports selection loss, round use, utilization, and transition penalty for these representative schedules. Our method matches Reference in the standard case, reduces unscheduled slab length and transition penalty in the objective extension, and uses fewer rounds with higher utilization in the uncertainty extension. Compared with DeepSeek + RAG in the constraint case, our schedule also combines lower loss and penalty with higher utilization. The benefits nevertheless involve trade-offs: the objective case has lower utilization than Reference, and the uncertainty case has a slightly higher penalty. The objective-case improvements despite lower HV further distinguish representative-schedule performance from front coverage. DeepSeek's lower loss and higher utilization in the constraint case describe a schedule selected from its limited feasible subset, rather than reliable performance across its outputs. Operational benefits should therefore be assessed jointly with target feasibility and the priorities assigned to competing production criteria.

5 Conclusion

We presented an LLM-based framework that aligns hot rolling scheduling models with production requirements. The framework reuses decision, constraint, and objective components from expert formulations, combining evidence-grounded selection and consistent composition with independent verification of the resulting schedules. Experiments show that the framework improves component selection and adaptation over the compared LLM baselines, with all returned schedules satisfying the checked target constraints. The Pareto and operational results show that target feasibility, front coverage, and the performance of an individual schedule capture distinct aspects of modeling outcomes. These findings support component-level knowledge reuse and independent verification as a basis for automating HRSP reformulation while preserving production requirements. The framework remains dependent on knowledge-base coverage, and the results do not establish equivalence to expert formulations. Future work will address requirements outside the current knowledge base through the construction and verification of new components and connections, and improve multi-objective search efficiency within practical planning budgets.

AI USE STATEMENT

LLM assistance was used to revise the writing and organization of this manuscript and to support literature retrieval and the discovery of related work. LLMs are also part of the proposed modeling framework and the experimental baselines, as described in Sections 3.2–4.

REFERENCES

Bahram Alidaee and Haibo Wang. On the integer programming formulation of production scheduling optimisation algorithm for the hot rolling processes. *International journal of production research*, 50(20):6036–6039, 2012.

A l Chen, GK Yang, and ZM Wu. Production scheduling optimization algorithm for the hot rolling processes. *International Journal of Production Research*, 46(7):1955–1973, 2008.

Lan Chen, Ling-ling Cao, Yao-min Wen, Hongsheng Chen, and Sheng-Long Jiang. A knowledge-based nsga-ii algorithm for multi-objective hot rolling production scheduling under flexible time-of-use electricity pricing. *Journal of Manufacturing Systems*, 69:255–270, 2023.

Wang Chen, Zhang Xiufeng, and Zhao Guohua. Research on hot rolling scheduling problem based on two-phase pareto algorithm. *Plos one*, 15(12):e0241077, 2020.

Wenchao Deng, Guodong Zhao, Qingxin Guo, and Yun Dong. Hybrid adaptive neighborhood multi-objective differential evolution algorithm improved by dbscan for hot rolling scheduling. *Expert Systems with Applications*, 309:131167, 2026.

Hamidreza Fardad, Faramarz Safi-Esfahani, and Behrang Barekatain. Optimizing production planning and sequencing in hot strip mills: an approach using multi-objective genetic algorithms. *The Journal of Supercomputing*, 81(1):88, 2025.

Chenyu Huang, Zhengyang Tang, Shixi Hu, Ruoqing Jiang, Xin Zheng, Dongdong Ge, Benyou Wang, and Zizhuo Wang. Orlm: A customizable framework in training large models for automated optimization modeling. *Operations Research*, 73(6):2986–3009, 2025.

Quirin Ilmer, Stefan Haeussler, and Hubert Missbauer. Slab assignment to non-identical reheat furnaces running in parallel mode. *Journal of the Operational Research Society*, 76(5):1034–1046, 2025.

Caigao Jiang, Xiang Shu, Hong Qian, Xingyu Lu, JUN ZHOU, Aimin Zhou, and Yang Yu. Llmopt: Learning to define and solve general optimization problems from scratch. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 101580–101606, 2025a.

Sheng-Long Jiang, Weigang Li, Xuejun Zhang, and Chuanpei Xu. An improved pareto local search for solving bi-objective scheduling problems in hot rolling mills. *Computers & Industrial Engineering*, 172:108561, 2022.

Sheng-Long Jiang, Lamei He, Lingling Cao, Liangliang Sun, and Gongzhuang Peng. A learning-enhanced ant colony optimization algorithm for integrated planning and scheduling in hot rolling production lines under uncertainty. *Swarm and Evolutionary Computation*, 93:101851, 2025b.

Sheng-Long Jiang, Qie Liu, Ling-Ling Cao, and Liangliang Sun. An online learning-based maco approach for hot rolling scheduling problems involving dynamic order arrivals. *IEEE Transactions on Automation Science and Engineering*, 2025c.

Xia Jiang, Yaoxin Wu, Minshuo Li, Zhiguang Cao, and Yingqian Zhang. Large language models as end-to-end combinatorial optimization solvers. *Advances in Neural Information Processing Systems*, 38:164787–164826, 2026.

Feng Li, Yanyan Zhang, Han Wei, and Xiaofan Lai. Integrated problem of soaking pit heating and hot rolling scheduling in steel plants. *Computers & Operations Research*, 108:238–246, 2019.

Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: towards efficient automatic algorithm design using large language model. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 32201–32223, 2024.

Zeyuan Ma, Yue-Jiao Gong, Hongshu Guo, Jiacheng Chen, Yining Ma, Zhiguang Cao, and Jun Zhang. Llamoco: Instruction tuning of large language models for optimization code generation. *IEEE Transactions on Evolutionary Computation*, 2026.

Manuj Malik, Jianan Zhou, Shashank Reddy Chirra, and Zhiguang Cao. Pyvrp+: Llm-driven metacognitive heuristic evolution for hybrid genetic search in vehicle routing problems. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems*, pp. 2800–2808, 2026.

Quan-Ke Pan, Qing-da Chen, Tao Meng, Bing Wang, and Liang Gao. A mathematical model and two-stage heuristic for hot rolling scheduling in compact strip production. *Applied mathematical modelling*, 48:516–533, 2017.

Quan-Ke Pan, Liang Gao, and Ling Wang. A multi-objective hot-rolling scheduling problem in the compact strip production. *Applied Mathematical Modelling*, 73:327–348, 2019.

Yarong Shi and Shixin Liu. Very large-scale neighborhood search for steel hot rolling scheduling problem with slab stack shuffling considerations. *IEEE Access*, 9:47856–47863, 2021.

Lulu Song, Ying Meng, Qingxin Guo, and Xinchang Gong. Improved differential evolution algorithm for slab allocation and hot-rolling scheduling integration problem. *Mathematics*, 11(9): 2050, 2023.

Yuanyuan Tan, MengChu Zhou, Yingying Wang, Xiwang Guo, and Liang Qi. A hybrid mip–cp approach to multistage scheduling problem in continuous casting and hot-rolling processes. *IEEE Transactions on Automation Science and Engineering*, 16(4):1860–1869, 2019.

Lixin Tang, Jiyin Liu, Aiying Rong, and Zihou Yang. A multiple traveling salesman problem model for hot rolling scheduling in shanghai baoshan iron & steel complex. *European Journal of Operational Research*, 124(2):267–282, 2000.

Nelson Torres, Gus Greivel, Joshua Betz, Eduardo Moreno, Alexandra Newman, and Brian Thomas. Optimizing steel coil production schedules under continuous casting and hot rolling. *European Journal of Operational Research*, 314(2):496–508, 2024.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. Reevo: Large language models as hyper-heuristics with reflective evolution. *Advances in neural information processing systems*, 37:43571–43608, 2024.

Biao Zhang, Quan-ke Pan, Liang Gao, Xin-li Zhang, and Qing-da Chen. A hybrid variable neighborhood search algorithm for the hot rolling batch scheduling problem in compact strip production. *Computers & Industrial Engineering*, 116:22–36, 2018.

Rui Zhang, Shiji Song, and Cheng Wu. The hot strip mill scheduling problem with uncertainty: Robust optimization models and solution approaches. *IEEE Transactions on Cybernetics*, 53(7): 4079–4093, 2022.

Zhuolun Zhang, Bailin Wang, Shuaipeng Yuan, Yiren Li, Xiqing Wang, and Tieke Li. Reinforcement-learning-based parallel genetic algorithm for the multi-objective hot-rolling scheduling problem of wide-thick slab. *Applied Soft Computing*, 179:113355, 2025.

Ziyan Zhao, Zikuo Bian, Jiaqi Liang, Shixin Liu, and MengChu Zhou. Scheduling and logistics optimization for batch manufacturing processes with temperature constraints and alternative thermal devices. *IEEE Transactions on Industrial Informatics*, 20(10):11930–11939, 2024.

APPENDIX

A HRSP SETTING AND TASK DEFINITIONS

A.1 PRODUCTION SETTING

An HRSP schedule selects orders, assigns their slabs to rolling rounds, and determines the sequence within each round. A round contains slabs processed consecutively between roll changes. Depending on the production setting, the model may also include start times, furnace assignments, or decisions for newly arriving orders. Figure 6 illustrates the principal decision stages, while Figure 7 shows the description and instance-data inputs.

Instance data specify slab attributes and the resources available to process them. Production constraints may bound the length or number of slabs in a round, restrict changes in width, thickness, or hardness between adjacent slabs, or impose time windows and resource limits. Dynamic settings may retain decisions already released to production, while uncertainty extensions change how feasibility or performance is evaluated under uncertain information. These conditions are included only when required by the target description. Objectives may penalize transitions, unselected orders, tardiness, or energy consumption; a preference expressed as a penalty must not be converted into a mandatory restriction.

A.2 MATHEMATICAL TASK

For a target pair (D, I), the variable vector $\boldsymbol{x}$ and its domain $\mathcal{X}(D, I)$ depend on the required decisions. Let g_q be an inequality constraint, h_q an equality constraint, and $\mathcal{Q}_{\leq}$ and $\mathcal{Q}_{=}$ their respective index sets. The intended feasible region is

$$\Omega(D, I) = \left\{ \boldsymbol{x} \in \mathcal{X}(D, I) : \begin{array}{ll} g_q(\boldsymbol{x}; D, I) \leq 0, & q \in \mathcal{Q}_{\leq}, \\ h_q(\boldsymbol{x}; D, I) = 0, & q \in \mathcal{Q}_{=} \end{array} \right\}. \tag{7}$$

After reversing maximization objectives for notational convenience, the multi-objective problem is

$$\min_{\boldsymbol{x} \in \Omega(D,I)} \boldsymbol{F}(\boldsymbol{x}; D, I) = \big(f_1(\boldsymbol{x}; D, I), \ldots, f_m(\boldsymbol{x}; D, I)\big). \tag{8}$$

A feasible schedule $\boldsymbol{y}$ dominates $\boldsymbol{x}$ if it is no worse in every objective and strictly better in at least one. The Pareto-optimal set is

$$\mathcal{P}^*(D, I) = \{\boldsymbol{x} \in \Omega(D, I) : \nexists \boldsymbol{y} \in \Omega(D, I) \text{ that dominates } \boldsymbol{x}\}. \tag{9}$$

The task produces an executable formulation and a validated finite approximation of this set. Requirement interpretation determines the target decisions, rules, and criteria; model construction expresses and connects them using the data in I. A generated model can be executable without representing $\Omega(D, I)$ correctly. Consequently, returned schedules are checked against the independently implemented target constraints, not only against the generated formulation.

B COMPONENT RECORDS AND VERIFICATION DETAILS

B.1 COMPONENT AND CONNECTION RECORDS

A component u retains its mathematical role, element set $\mathcal{E}_u$, semantic description $\boldsymbol{s}_u$, and supporting expert formulation. The semantic description records the following fields:

- The decision, production restriction, or performance criterion represented by the component.
- The affected production objects, stages, and index domains.
- The mathematical function, restriction or optimization direction, and activation conditions.
- The calculation scope and objective organization, when applicable.

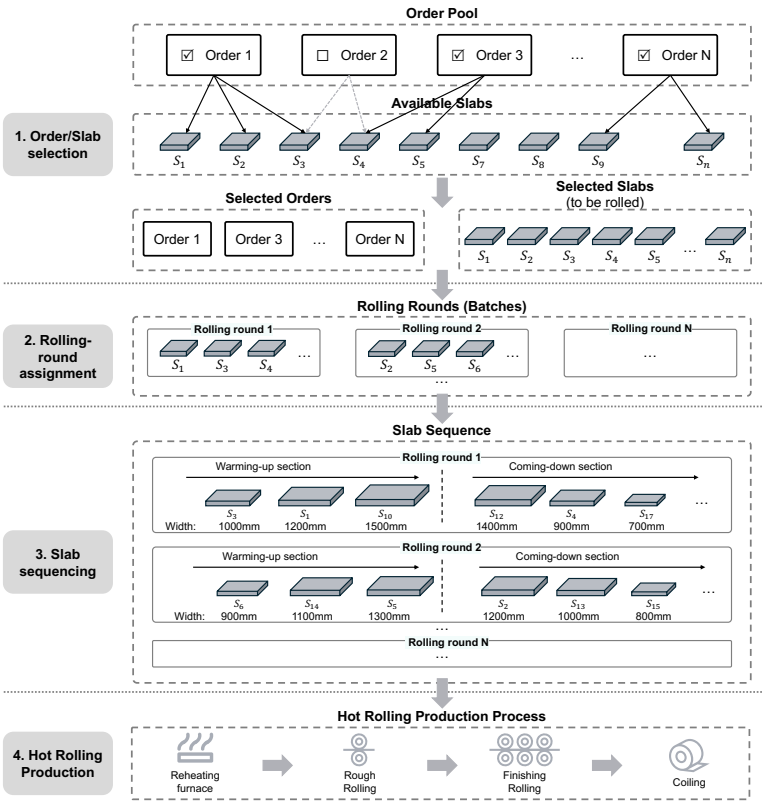


Figure 6: Selection, rolling-round assignment, and sequencing in HRSP.

Natural-Language HRSP Description D

At each event time t, select high-reward existing and newly arrived orders and group them into a fixed number of rolling rounds.

Each round contains a manually determined warming-up section and a coming-down section. A descending width profile is preferred; width increases are penalized rather than prohibited.

Round-length, same-width-run, and adjacent width/thickness/hardness jump limits must be satisfied. Fixed orders keep their assignments and sequences.

Minimize transition penalty and unselected-order loss simultaneously.

Instance Data I

4 rolling rounds | 120 orders | event times: 0, 27, 34, 47

```
[VEHICLE]
V     C     U
1   800    10
2   800    10
3   800    10
4   800    10

[CUSTOMER]
T    id    W   G   H   L     P
0     1   14   3   2  36   9.7
0     2    6   4   2  29   1.5
27  121    6   3   2  32   3.8
34  129    6   2   3  21   4.5
47  139    6   2   2  20   3.1
```

T arrival time | W width | G thickness | H hardness
L length | P unselected-order loss

Figure 7: An example of the natural-language description D and instance data I. The data excerpt is illustrative rather than the complete input file.

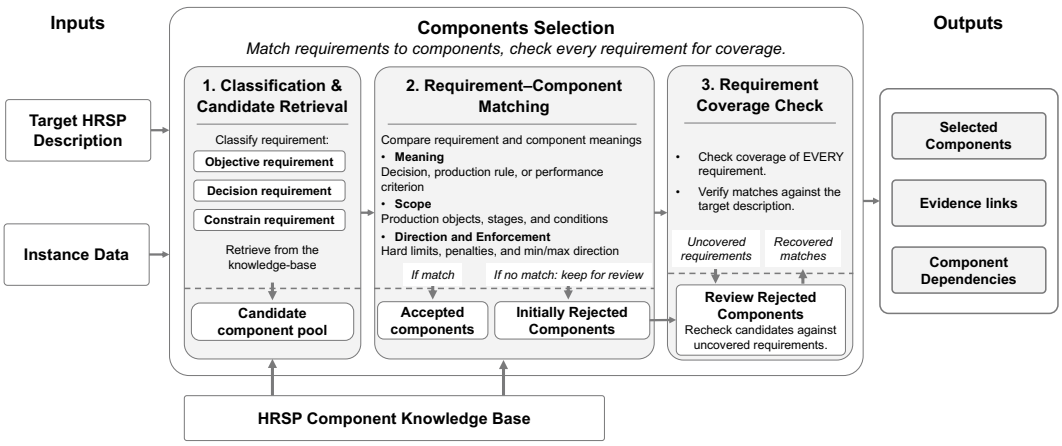


Figure 8: Component Selection combines requirement classification, semantic matching, and evidence-based coverage verification. Requirement classes determine the component category searched; initially rejected candidates remain available for reassessment when a requirement is uncovered.

Each element retains variable domains, quantifiers, boundary definitions, and its required data. Component identifiers distinguish differences in these fields even when their formulas use similar notation. Conversely, different symbol names do not create different components when the production meaning and mathematical representation are equivalent.

A connection record contains its endpoint components, shared production objects, variable and index correspondences, applicability conditions, and any expression needed to implement the relation. For a constraint or objective, the record identifies the decision quantities required by its elements. For two decisions, it identifies how their representations must agree. Only one relation set $E_{\mathcal{K}}$ is used; the endpoint types determine the interpretation. Structural constraints that express decision relations do not add production requirements.

B.2 Component Selection and Coverage Recovery

The component records described above support requirement classification, semantic matching, and coverage verification in Section 3.2. Figure 8 illustrates their relationship: matching identifies components for individual requirements, while the reverse coverage check reveals uncovered requirements and supports review of initially rejected candidates. The following paragraphs detail the evidence and dependency records retained for subsequent composition.

Requirement-guided retrieval. Each requirement in $\mathcal{R}(D, I)$ retains its supporting text in D and is classified as a decision, constraint, or objective requirement. A decision requirement specifies what must be chosen, assigned, or sequenced; a constraint requirement specifies a mandatory production rule; an objective requirement specifies a performance criterion or preference. The classification follows the meaning of the statement rather than isolated keywords. For example, prohibiting a width increase imposes a constraint, whereas penalizing it contributes to an objective. The instance schema supplies the meanings of referenced objects and parameters but does not turn an available data field into a production requirement.

RAG retrieves component records using each requirement's class and production meaning. The candidates for all requirements form the pool $\mathcal{U}_{\text{cand}}$, retaining the originating requirement for each retrieval. Retrieval does not exclude candidates because their scope, direction, conditions, or notation differ from the target; applicability depends on semantic correspondence. Retrieval supplies candidates and their recorded semantics, not a final decision about inclusion.

Semantic correspondence. The LLM compares each retrieved component with its associated requirement using the following fields:

- **Meaning:** whether both express the same production decision, mandatory rule, or performance criterion.
- **Scope:** whether they concern the same production objects, stages, index ranges, and application conditions.
- **Direction and enforcement:** whether they agree on hard restrictions versus penalties, bound directions, and minimization versus maximization. For an objective, the evaluated decisions, calculation scope, and independent or combined treatment must also agree.

The mathematical elements are read with their semantic descriptions, so identical symbols do not establish a match and different symbols do not prevent one. All applicability conditions required by a component must be supported by the target input; agreement on only the fields stated explicitly in a single requirement is insufficient. Related statements in D may resolve an omitted qualifier, but a condition from the source formulation cannot be imported without target support.

An accepted match records the component and the requirement supporting its use. The initially rejected pool retains non-matching components, the requirements considered, and the conflicting or unsupported fields. These records are maintained per requirement–component pair: rejecting a component for one requirement does not prevent its acceptance for another. The rejection reason provides a basis for distinguishing an overlooked qualifier or an incorrect interpretation from a genuine mismatch.

Evidence and requirement coverage. Membership in $\mathcal{A}$ requires evidence from the target input. Its evidence record identifies the supporting statement in D, the semantic fields it supports, and the data fields or explicit derivations in I required by the component's elements. A match that fails this check is returned to the rejected pool with the missing evidence recorded. Equation (2) evaluates coverage from the requirement side of this correspondence. One requirement may match several components, and one component may match several requirements. The coverage test counts a link only when it supports the requirement's full stated meaning, not merely a shared term or production object.

An uncovered requirement provides the basis for reassessing the initially rejected pool, including candidates originally considered for another requirement. The LLM compares the recorded rejection reasons with the target text and data to distinguish actual incompatibilities from misread conditions or overlooked correspondences. Recovery requires the same semantic agreement and target evidence as initial acceptance. Each recovered match changes $\mathcal{A}$ and the corresponding coverage, allowing further review until all requirements are covered or no additional supported match is found. Unresolved requirements remain explicit; neither a rejected component nor an assumption from another variant is accepted merely to complete coverage.

Evidence and dependency records. Equation (3) gives the selected component set after review. Each selected component is accompanied by its verified requirement-to-component evidence links and the dependency records identifying the decisions required by its elements. The corresponding connection records remain available for composition. A dependency on a decision not yet selected is retained for checking rather than removed from the record; composition returns such missing decisions to Component Selection. These outputs support the subsequent assembly of the constraint system and objective vector without performing component connection during selection.

B.3 Component composition and expression assembly

The selected components and their dependency records provide the input to component composition in Section 3.3. Figure 9 shows how applicable connection records support objective and constraint modeling, followed by validation of the assembled connections. A missing required decision component is returned to Component Selection before the affected expressions are assembled.

Assembly preserves the meaning of the selected elements while replacing their symbols with the target variables, sets, and parameters. The replacement is determined by shared production objects and decision definitions, not by equal symbol names. For a recorded connection, its conditions must hold before its expression is used. For a derived connection, the derivation must identify the selected decisions being related and the index ranges over which they describe the same production objects.

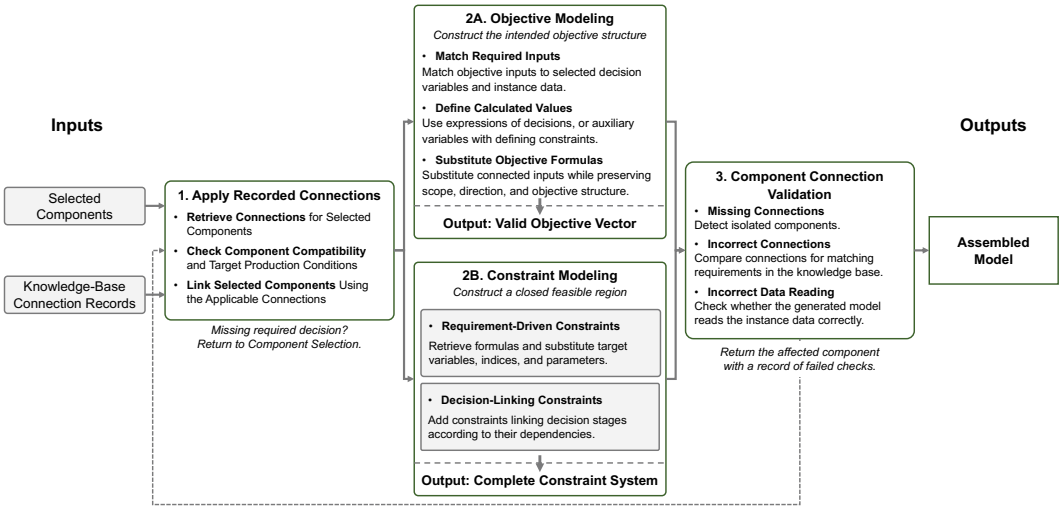


Figure 9: Component composition. Applicable connection records link selected components; objective and constraint definitions are assembled and checked for missing connections, incompatible relations, and incorrect data references.

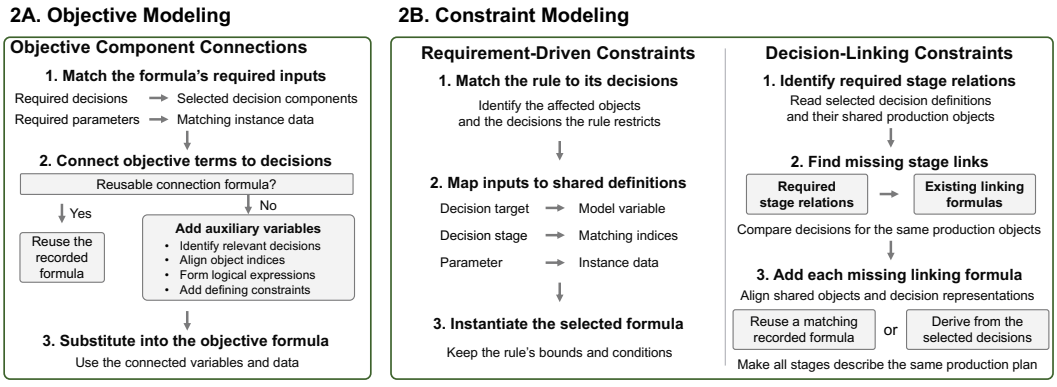


Figure 10: Connecting objective and constraint components. Objective terms and production rules are matched to target decisions and data. Decision-linking constraints reconcile the selected stages; auxiliary quantities require defining constraints.

Within this shared decision structure, objective terms and production rules are connected to target variables and data, while decision-linking constraints reconcile the selected stages. Figure 10 details these connections, including when to reuse a recorded formula or derive a relation from the selected decision definitions.

Objective Modeling identifies the relevant decisions and data, constructs the required expression, and preserves its calculation scope and optimization direction. Independent criteria remain separate for Pareto optimization. Weights or priority levels are used only when specified by the target input. Any auxiliary definition created for an objective is included in the constraint system and checked with the other decision-linking constraints.

Constraint Modeling separates target production restrictions from decision-linking constraints, including selection–assignment links, assignment–sequence links, and auxiliary definitions. Each auxiliary variable requires a domain and all defining constraints needed for the intended quantity, even when a logical or aggregated expression spans several elements. Missing data or undefined quantities cannot be replaced with invented parameters.

B.4 Model closure and solution checks

The connection validation in Figure 9 checks the assembled model and its evidence records against the following closure conditions:

1. **Requirement coverage:** every accepted production requirement contributes its complete component elements; unresolved requirements prevent acceptance.
2. **Decision connections:** every required relation among selected decisions is represented, without links to absent decisions or incompatible production scopes.
3. **Expression references:** every symbol, index, and parameter in a constraint or objective has a compatible definition and an available data source.
4. **Connection justification:** each link agrees with an applicable knowledge-base record or a checked derivation from the selected decision meanings; alternative records are distinguished by their conditions.
5. **Objective structure:** all intended criteria are computable with their specified directions, scopes, and preference organization.

These checks distinguish semantic judgments about production meaning from checks of explicit symbol, index, and data references. Missing required components return to selection; errors in the relations among present components return to assembly. No repair may add an unsupported production requirement.

Executable verification checks whether the accepted expressions can be constructed in the solver model. Solution validation then checks the returned schedules against independently implemented target constraints, reevaluates their objectives, and removes dominated or duplicate objective points. Passing these checks provides evidence for the reported schedules; it is not a proof of equivalence between the generated and expert feasible regions.

C Evaluation protocols and metric definitions

C.1 Dataset and comparison conditions

The 25 studies include 5 standard HRSP formulations, 16 decision-scope extensions, 18 constraint extensions, 16 objective extensions, and 6 uncertainty extensions. Counts overlap because a study can extend several aspects of the standard problem. One distinct paper is sampled from each category, producing five held-out test papers and 20 knowledge-base papers. All information from each test paper is excluded from knowledge-base construction. The formulations in the 20 knowledge-base studies are decomposed into decision, constraint, and objective components. The original formulations in the five test studies provide the reference components and target hard constraints; no expert reconstruction of these models is required.

The methods receive the same target descriptions and instance data. DeepSeek uses DeepSeek V4 without retrieved context; DeepSeek + RAG retrieves from the full texts of the 20 knowledge-base studies; our method performs component selection and composition. These three methods share the backbone and generation settings. ORLM (Huang et al., 2025) and LLMOPT (Jiang et al., 2025a) use their released configurations.

The best result in the literature is labeled *Reference* in the result tables and plots. For each test case, it uses the original reported results when complete experimental results are publicly available. Otherwise, the study's original formulation provides the reference model for Gurobi-based Pareto search. Original published results retain their reported computational settings. Generation completeness, solver feasibility, and target feasibility are distinct evaluation outcomes.

Pareto-search settings. The evaluation protocol allocates a three-hour budget per model to a common Pareto-search procedure. Within this budget, up to 10 minutes are allocated to finding an initial feasible solution, 30 minutes to optimizing each objective separately to obtain anchor solutions, and two minutes to each subsequent search subproblem. All searches are subject to the remaining total budget. For each subproblem, the solver maintains a pool of up to 1,024 feasible candidates within a 0.25% relative objective gap of the best solution found. All other Gurobi settings remain at their defaults. Independent feasibility verification fixes the decisions of each returned schedule and allows a separate two-minute checking period.

C.2 Component-selection measures

For test case i, let $\mathcal{U}_i^*$ denote the reference component set from the test formulation and $\widehat{\mathcal{U}}_i$ the set selected by a method. Two types of selection error are considered. A component in $\widehat{\mathcal{U}}_i \setminus \mathcal{U}_i^*$ is an incorrectly selected component because it is not required by the target problem, whereas a component in $\mathcal{U}_i^* \setminus \widehat{\mathcal{U}}_i$ is an omitted component. These errors are measured by component precision, recall, and F1:

$$P_i = \frac{|\widehat{\mathcal{U}}_i \cap \mathcal{U}_i^*|}{|\widehat{\mathcal{U}}_i|}. \tag{10}$$

$$R_i = \frac{|\widehat{\mathcal{U}}_i \cap \mathcal{U}_i^*|}{|\mathcal{U}_i^*|}. \tag{11}$$

$$F1_i = \frac{2P_iR_i}{P_i + R_i}. \tag{12}$$

Precision decreases when unsupported components are retained, while recall decreases when required components are omitted. The three measures are first calculated for each case and then averaged across the five cases. To determine which modeling roles are most difficult to identify, F1 is also calculated separately for decision, objective, and constraint components.

C.3 Single-requirement updates

For each of the five original descriptions, three controlled variants each change one decision, constraint, or objective requirement. An edit adds, removes, or modifies the requirement while preserving all other requirements and the instance data. Each method is evaluated on all 15 original–modified pairs.

For each pair, a method is first applied to D_i^{ori} to obtain $\widehat{\mathcal{U}}_i^{\mathrm{ori}}$ and then to the modified description D_i^{mod} to obtain $\widehat{\mathcal{U}}_i^{\mathrm{mod}}$. The generated change set records both additions and removals:

$$\widehat{\mathcal{E}}_i = \left\{ +u : u \in \widehat{\mathcal{U}}_i^{\mathrm{mod}} \setminus \widehat{\mathcal{U}}_i^{\mathrm{ori}} \right\} \cup \left\{ -u : u \in \widehat{\mathcal{U}}_i^{\mathrm{ori}} \setminus \widehat{\mathcal{U}}_i^{\mathrm{mod}} \right\}. \tag{13}$$

A modified component is represented by removal of its original version and addition of its revised version. The expected signed change set $\mathcal{E}_i^*$ is obtained from the expert requirement-to-component annotations for the original and modified descriptions. The experiment compares $\widehat{\mathcal{E}}_i$ with $\mathcal{E}_i^*$.

Three error types are considered. An unnecessary change occurs when an addition or removal appears in $\widehat{\mathcal{E}}_i$ but not in $\mathcal{E}_i^*$; a missed change occurs when an expected addition or removal is absent from $\widehat{\mathcal{E}}_i$; and unstable selection occurs when a component unrelated to the changed requirement is not preserved across the two runs. Component-change precision, recall, and F1 are defined as

$$P_{\Delta,i} = \frac{|\widehat{\mathcal{E}}_i \cap \mathcal{E}_i^*|}{|\widehat{\mathcal{E}}_i|}. \tag{14}$$

$$R_{\Delta,i} = \frac{|\widehat{\mathcal{E}}_i \cap \mathcal{E}_i^*|}{|\mathcal{E}_i^*|}. \tag{15}$$

$$F1_{\Delta,i} = \frac{2P_{\Delta,i}R_{\Delta,i}}{P_{\Delta,i} + R_{\Delta,i}}. \tag{16}$$

Let $\mathcal{H}_i^* = \mathcal{U}_i^{*,\mathrm{ori}} \cap \mathcal{U}_i^{*,\mathrm{mod}}$ denote the expert components that should remain unchanged. Their preservation is measured by unchanged-component stability:

$$S_i = \frac{|\mathcal{H}_i^* \cap \widehat{\mathcal{U}}_i^{\mathrm{ori}} \cap \widehat{\mathcal{U}}_i^{\mathrm{mod}}|}{|\mathcal{H}_i^*|}. \tag{17}$$

Change precision measures whether the generated changes are necessary, change recall measures whether all required changes are made, and stability measures whether components outside the changed requirement are preserved. The metrics are averaged across the 15 tests and are also calculated separately for decision, objective, and constraint components.

C.4 Feasibility and Pareto-front measures

The first evaluation determines whether a generated model is executable and whether its solutions satisfy the target problem rather than only the generated formulation. Solver feasibility records whether Gurobi can construct and solve the generated model and return at least one feasible solution under that model. Models that fail this check are not evaluated further.

For a solver-feasible model, every returned schedule is independently checked against the hard constraints of the expert reference formulation. Let $\mathcal{P}_i$ denote the non-dominated schedules returned for case i and $\mathcal{X}_i^*$ the feasible region defined by the expert hard constraints. The feasible subset is

$$\mathcal{P}_i^{\mathrm{valid}} = \mathcal{P}_i \cap \mathcal{X}_i^*. \tag{18}$$

The problem feasibility status is *Pass* when every returned schedule is target-feasible and *Fail* when none satisfies the expert constraints. It is *Partial* when the target-feasible set is nonempty but does not contain every returned schedule. This distinction detects schedules that satisfy the generated model but violate the target problem; identifying the responsible component or connection requires inspecting the formulation.

Pareto-front indicators are computed only from $\mathcal{P}_i^{\mathrm{valid}}$. All objectives are normalized using common bounds across the compared methods. For a normalized m-objective front $\mathcal{P}$ and a common reference point $\boldsymbol{r} = (r_1, \ldots, r_m)$, hypervolume is

$$\mathrm{HV}(\mathcal{P}) = \lambda_m\left(\bigcup_{\boldsymbol{x}\in\mathcal{P}} \prod_{q=1}^{m} [\widehat{f}_q(\boldsymbol{x}), r_q]\right), \tag{19}$$

where λ_m denotes m-dimensional Lebesgue measure and a larger value indicates broader objective-space coverage. The inverted generational distance from the reference front $\mathcal{P}^{\mathrm{ref}}$ is

$$\mathrm{IGD}(\mathcal{P}, \mathcal{P}^{\mathrm{ref}}) = \frac{1}{|\mathcal{P}^{\mathrm{ref}}|} \sum_{\boldsymbol{y}\in\mathcal{P}^{\mathrm{ref}}} \min_{\boldsymbol{x}\in\mathcal{P}} \left\| \widehat{\boldsymbol{f}}(\boldsymbol{x}) - \widehat{\boldsymbol{f}}(\boldsymbol{y}) \right\|_2, \tag{20}$$

where a smaller value indicates better coverage of the reference-front points. The reference front uses the publicly available results of the test study or, when those results are incomplete or unavailable, the front obtained by solving its original formulation within the three-hour budget. Neither source is assumed to provide a certified optimal front.

C.5 Operational measures

Operational performance is assessed separately for order/slab selection, rolling-round assignment, and slab sequencing; a variant need not include all three stages. The four reported indicators are Loss, R, RRU, and P, matching Table 3. All operational indicators use one representative schedule from each method's target-feasible, non-dominated set. For newly solved models, this set contains only solutions found within the common three-hour budget; published reference results follow the baseline policy above. With common objective normalization and equal weights, the schedule closest to the common ideal point is selected:

$$\boldsymbol{x}_i^{\mathrm{rep}} = \arg\min_{\boldsymbol{x}\in\mathcal{P}_i^{\mathrm{valid}}} \sqrt{\sum_{q=1}^{m} w_q \widehat{f}_q(\boldsymbol{x})^2}, \qquad w_q = \frac{1}{m}. \tag{21}$$

The common ideal point contains the best value of each target objective across all compared methods' target-feasible solutions within a case. Maximization criteria are converted to minimization, and the normalized objective origin represents this common ideal point. The same bounds and weights apply to every method, so indicators cannot be drawn from different favorable schedules. For cases with multiple decision events, the same selection rule is applied at each event before averaging the reported operational indicators.

Loss: order/slab selection. Loss measures the cost of leaving orders or slabs unscheduled, using the definition applicable to each case:

$$\mathrm{Loss} = \begin{cases} \displaystyle\sum_{j\in J\setminus J_s} p_j, & \text{Standard, decision-scope, or constraint case,} \\ \displaystyle\sum_{j\in J\setminus J_s} l_j, & \text{objective-extension case.} \end{cases} \tag{22}$$

Here, J denotes the candidate orders or slabs evaluated in the case, and J_s is the scheduled subset. The coefficient p_j is the loss associated with excluding order j, whereas l_j is slab length in the instance's length unit. Thus, the Loss column reports unselected-order loss in the Standard, decision-scope, and constraint cases, but total unscheduled slab length in the objective-extension case. Lower Loss is preferred; it is not applicable to the uncertainty-extension case and is reported as N/A.

R: rolling rounds used. Let K^+ denote the set of active rolling rounds in the selected schedule. The round count is

$$R = |K^+|. \tag{23}$$

This indicator measures resource use. Fewer rounds are preferable for a comparable production workload, so R must be interpreted together with Loss.

RRU: rolling-round utilization. RRU is the mean fraction of capacity used across the R active rounds:

$$\mathrm{RRU} = \frac{1}{R}\sum_{k\in K^+} \frac{\sum_{j\in S_k} l_j}{U_k} \times 100\%, \qquad R > 0. \tag{24}$$

Here, S_k is the set of slabs assigned to round k, and U_k is its capacity in the same length units as the assigned load. RRU is reported as a percentage; higher values are preferred.

P: slab-sequencing transition penalty. The transition penalty P is used as an indicator of roll wear and sums the penalties for consecutive slabs under the production rules of the target case:

$$P = \sum_{(i,j)\in A} \pi_{ij}, \tag{25}$$

where A contains consecutive slab pairs within the scheduled rounds and π_{ij} is their transition penalty. Lower P is preferred. Each case retains its own Loss and P definitions and scales, so these indicators are compared within cases, not across variants. A dash denotes the absence of a target-feasible schedule, whereas N/A denotes an inapplicable indicator.

Table 4: Component-selection F1 by mathematical role.

Method	Decision F1	Objective F1	Constraint F1
ORLM	0.500	0.250	0.320
LLMOPT	0.541	0.727	0.339
DeepSeek	0.880	**1.000**	0.875
DeepSeek + RAG	0.863	**1.000**	0.860
Ours	**0.920**	**1.000**	**0.935**

Table 5: Component-change F1 and unchanged-component stability by mathematical role.

	Change F1			Stability		
Method	Decision	Objective	Constraint	Decision	Objective	Constraint
ORLM	0.000	0.000	0.000	0.600	0.600	0.600
LLMOPT	0.000	0.074	0.080	0.771	0.686	0.571
DeepSeek	0.235	0.182	0.526	0.921	0.742	0.933
DeepSeek + RAG	0.400	0.345	0.455	0.900	0.844	0.889
Ours	**0.889**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**

D ROLE-SPECIFIC SELECTION AND UPDATE RESULTS

Table 1 in the main text compares initial component selection with adaptation to single-requirement changes; Figure 5(a) and (b) report the respective F1 scores by component type using a shared legend. The tables below provide the exact component-type scores and unchanged-component stability. For selection, ORLM produces complete formulations for 2 of the 5 cases. For requirement updates, ORLM produces complete formulations for 4 of the 15 tests; LLMOPT and DeepSeek do so for 14 of the 15 tests. Incomplete outputs remain in the evaluation rather than being excluded from the reported averages.

Our method achieves decision and constraint selection F1 of 0.920 and 0.935, respectively, as shown in Table 4. DeepSeek and DeepSeek + RAG match its objective F1 but are less accurate for decisions and constraints. For updates, our method attains perfect objective and constraint change F1 and preserves all unchanged components, as reported in Table 5. The remaining decision-update error is an omission, not an unnecessary change to another component.

The requirement-change results differ from initial selection: DeepSeek's objective F1 is only 0.182, despite its perfect initial objective-selection F1. Whole-paper RAG raises its decision and objective change F1 from 0.235 and 0.182 to 0.400 and 0.345, respectively, but reduces constraint change F1 from 0.526 to 0.455. Overall stability improves only slightly, from 0.865 to 0.878; by type, objective stability improves, while decision and constraint stability decline. Thus, retrieval does not consistently identify which components should change and which should be preserved. ORLM has zero change F1 across all three types, while LLMOPT attains at most 0.080 and has constraint stability of 0.571.

E ADDITIONAL SOLUTION-QUALITY ANALYSIS

Tables 2 and 3 separate the operations-research evaluation from the production indicators. The first table retains all six methods and reports target-constraint pass rates, HV, and IGD. The second table retains all six methods in every case; a dash explicitly marks unavailable operational results. The discussion below provides further comparisons for each case without duplicating the tables. Reference IGD is zero by construction, and reference results are not assumed to be globally optimal.

Figure 11 visualizes the two Pareto indicators; comparisons are made within cases because objective definitions and normalization differ across cases.

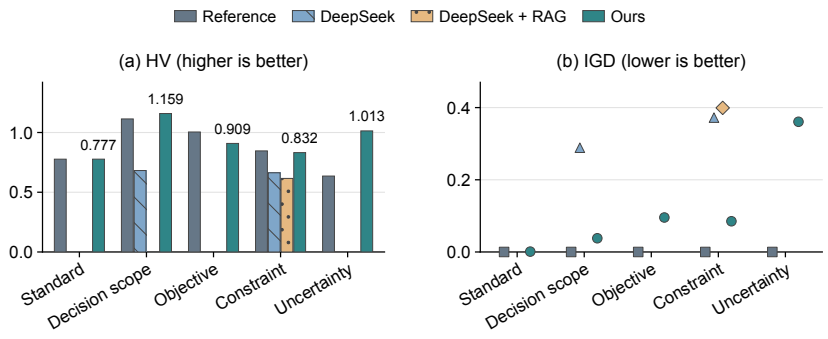


Figure 11: Pareto quality of target-feasible solutions. Missing bars or points denote unavailable results, not zero. ORLM and LLMOPT have no solver-feasible output and therefore no plotted values. Reference IGD is zero by construction.

E.1 Standard HRSP

Our model and the reference both return only target-feasible schedules. Their HV values are 0.777 after rounding, although the proposed model's value is slightly higher before rounding; IGD is 0.001. The representative schedules also coincide on all reported production indicators: loss 38.000, six rolling rounds, utilization 10.010%, and transition penalty 6.100. DeepSeek and DeepSeek + RAG return solver-feasible schedules, but none satisfies the target constraints. The observed agreement supports reconstruction of the core decision relations on this instance, without establishing equivalence of the complete feasible regions.

E.2 Decision-scope extension

Both our method and DeepSeek achieve full target feasibility, while DeepSeek + RAG fails the target checker. Our HV of 1.159 exceeds the reference value of 1.114 and DeepSeek's 0.682; the corresponding IGD values are 0.038 and 0.289 for the two generated models. All valid representative schedules use four rolling rounds. Our utilization is highest at 49.440%, but loss is 578.000 compared with 571.000 for the reference and 573.500 for DeepSeek. Our transition penalty of 46 919.560 lies between the reference's 37 475.900 and DeepSeek's 55 988.250. Thus higher HV or utilization does not imply improvement in every production measure. These differences alone do not establish whether a specific constraint is unnecessarily restrictive.

E.3 Objective extension

Only our method and the reference yield target-feasible schedules. Our HV is lower than the reference (0.909 versus 1.004), with IGD 0.096. However, our representative schedule reduces unscheduled slab length from 8 300.000 to 1 378.000 and transition penalty from 3 555.000 to 3 335.000. Both schedules use five rolling rounds, while utilization decreases from 97.390% to 95.870%. The improvement in these selected production measures can coexist with a weaker overall Pareto front because the operational evaluation concerns one schedule rather than the entire set of trade-offs.

E.4 Constraint extension

Our method passes the target checker for every returned schedule, with HV 0.832 compared with 0.846 for the reference. Only 26.630% of DeepSeek schedules and 36.610% of DeepSeek + RAG schedules pass; all their reported quality metrics use these valid subsets. All representative schedules use four rolling rounds. Our loss, utilization, and transition penalty are 68.530, 95.290%, and 273.500, respectively, each better than DeepSeek + RAG's corresponding value. DeepSeek attains lower loss (59.100) and higher utilization (98.550%) on its selected valid schedule, but its transition penalty of 274.500 is slightly higher than ours. The reference has the lowest transition penalty,

259.500. Strong indicators for one verified schedule should therefore be distinguished from the fraction of all returned schedules that satisfy production requirements.

E.5 Uncertainty extension

Our method remains fully target-feasible and raises HV from 0.636 to 1.013. Its IGD of 0.361 indicates a different distribution of trade-offs, not uniform superiority to the reference front. For the representative schedule, the number of rolling rounds decreases from five to four and utilization increases from 77.000% to 96.250%. Transition penalty rises slightly from 5.120 to 5.380, while selection loss is not applicable. This case illustrates a production trade-off between round use and transition quality.

F Extended related work

F.1 HRSP formulations

Existing HRSP studies have commonly formulated hot rolling scheduling as routing-type optimization problems. However, these formulations are variant-specific and require expert modeling. In this paradigm, slabs or orders are treated as nodes, rolling rounds or rolling turns are represented as routes, and transition penalties between adjacent slabs are modeled as arc costs. Tang et al. (2000) formulated hot rolling scheduling as a multiple traveling salesman problem. Chen et al. (2008) and Alidaee & Wang (2012) subsequently modeled it as a Vehicle Routing Problem (VRP), in which route-capacity constraints represent the production limits of each rolling round. More recent studies have further extended this routing-based perspective to capture variant-specific industrial requirements. Chen et al. (2020) and Chen et al. (2023) considered energy-oriented hot rolling scheduling through prize-collecting VRP formulations. Jiang et al. (2022) incorporated temporal constraints into a prize-collecting capacitated VRP with time windows, and Jiang et al. (2025c) modeled dynamic order arrivals as a dynamic VRP. Jiang et al. (2025b) further integrated hot-rolling planning with reheating-furnace scheduling using a VRP with resource thresholds, while Zhang et al. (2025) modeled wide-thick slab scheduling as a multi-objective VRP in which the transition cost between two slabs depends on their rolling order. These studies indicate that VRP-based formulations provide a flexible mathematical foundation for HRSP, but the resulting models remain highly dependent on variant-specific production assumptions and expert-defined mathematical structures.

F.2 HRSP solution methods

Exact methods can certify feasibility or optimality when the corresponding mathematical conditions and termination criteria are met. As the scope of steel-production scheduling expanded across production stages, Tan et al. (2019) developed a hybrid exact framework for integrated continuous-casting and hot-rolling scheduling. Within a related integrated setting, Li et al. (2019) formulated the combined scheduling of pre-rolling heating and subsequent hot rolling as a mixed-integer program solved by branch-and-bound. Exact formulations were subsequently adapted to more specific operational requirements: Shi & Liu (2021) incorporated slab-stack shuffling through integer programming and branch-and-bound repair, whereas Zhang et al. (2022) addressed processing-time uncertainty using robust optimization and Benders decomposition. For integrated casting-and-rolling scheduling, Torres et al. (2024) used variable reduction and cutting planes to reduce computational complexity. At the equipment-assignment level, Ilmer et al. (2025) combined mixed-integer optimization with iterative parameter adjustment for non-identical reheating furnaces. Together, these studies show the strength of exact optimization, but also its dependence on expert formulations and problem-specific solution enhancements.

As HRSP models became larger and more tightly coupled, research increasingly traded optimality guarantees for the scalability of heuristic search. Early work exploited problem structure through decomposition: Pan et al. (2017) separated compact-strip scheduling into assignment and sequencing subproblems, while Zhang et al. (2018) designed hybrid variable-neighborhood search for the batch-scheduling variant. When multiple production criteria became important, Pan et al. (2019) combined a scheduling heuristic with a multi-objective evolutionary algorithm to approximate Pareto schedules. Along this direction, Jiang et al. (2022) developed Pareto local search for temporal and process

constraints, and Chen et al. (2023) embedded domain knowledge in NSGA-II for energy-oriented scheduling under time-of-use electricity pricing. The application scope then broadened beyond rolling sequences alone. Song et al. (2023) applied improved differential evolution to integrated slab allocation and hot-rolling scheduling, whereas Zhao et al. (2024) used enhanced simulated annealing for temperature-constrained batch production and logistics. At the production-planning level, Fardad et al. (2025) combined heuristic search with NSGA-II to coordinate planning and sequencing. These methods improve scalability, but their encodings, neighborhoods, repair rules, and parameter settings remain closely tied to the HRSP variant for which they were designed.

To reduce manual search control in changing production environments, recent studies have embedded learning components within metaheuristics. These methods use learning to adapt the search strategies and operators applied to an already formulated scheduling problem. Zhang et al. (2025) used Q-learning to tune a parallel genetic algorithm for wide-thick slab scheduling. For dynamic order arrivals, Jiang et al. (2025c) combined heuristic selection and pheromone transfer with multi-objective ant-colony optimization. For scheduling under resource uncertainty, Jiang et al. (2025b) used learning mechanisms to select search operators and guide the search toward better solutions. Deng et al. (2026) further developed a hybrid multi-objective differential-evolution method to improve convergence and solution diversity. Learning therefore improves search adaptability, but still assumes that the correct decisions, objectives, and constraints have already been formulated.

F.3 LLMs FOR OPTIMIZATION

Because exact, heuristic, and learning-based solvers all assume that the optimization problem has already been specified, recent LLM research has moved automation upstream toward model and algorithm construction. For optimization modeling, Huang et al. (2025) proposed ORLM to generate mathematical formulations and solver code with open-source LLMs. Addressing the same stage, Jiang et al. (2025a) developed LLMOPT to improve formulation definition and code generation from natural-language descriptions. Moving from problem definitions to executable implementations, Ma et al. (2026) introduced LLaMoCo for instruction-tuned optimization code generation. A second research direction uses LLMs to design the solution algorithm itself. Liu et al. (2024) proposed Evolution of Heuristics to jointly evolve heuristic ideas and executable code. Building on reflection during evolutionary search, Ye et al. (2024) developed ReEvo for automatic hyper-heuristic generation. Within vehicle routing, Malik et al. (2026) used LLM-driven metacognitive evolution to improve hybrid genetic search components. At the direct-solution level, Jiang et al. (2026) investigated whether LLMs can map natural-language combinatorial problems directly to solutions. These studies demonstrate the potential of LLMs in optimization modeling, code generation, and heuristic design. However, existing LLM-based methods cannot reliably identify the small but important differences among HRSP variants. These differences may change the required decisions, constraints, objectives, or their connections; overlooking them can produce an incorrect model and solutions that are infeasible for the target production setting. Expert review is therefore still required to adapt and validate the generated formulation.