SEARCH BEFORE YOU OPTIMIZE: PROACTIVE RULE DISCOVERY FOR EVIDENCE-GROUNDED OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Constraints required by real-world optimization tasks may originate outside the problem description, and their applicability varies with jurisdiction, time, and task conditions. However, existing methods neither proactively assess whether the available information is sufficient for modeling nor actively search for missing external information, which can lead to modeling failures. To address this gap, we introduce SearchWorthyOR (SWOR), a benchmark built around real-world external rules with three paired conditions: rule inapplicable, applicable but omitted, and applicable and provided. SWOR evaluates rule discovery, applicability assessment, and information integration. We further propose SearchWorthy Agent (SWAgent), which uses an information–evidence table to link conditions requiring verification, supporting evidence, and modeling effects, organize on-demand search, and revise or retain the model according to verified findings. On SWOR, LLM answer accuracy is substantially lower in cases requiring proactive search than in those with rules provided, demonstrating the importance of this challenge for LLM-based automated modeling. SWAgent achieves 79.00% answer accuracy, the highest among the compared workflows, demonstrating its advantage in proactive external search. Our code is available[1].

1 INTRODUCTION

Operations research (OR) translates requirements into variables, objectives, and constraints (Bengio et al., 2021; NL4Opt, Ramamonjison et al., 2022; OptiMUS, AhmadiTeshnizi et al., 2026). Large language models (LLMs) automate this process (CoE, Xiao et al., 2024; ORLM, Huang et al., 2025a; LLMOPT, Jiang et al., 2025). Real-world requirements also arise from external rules whose applicability depends on jurisdiction, time, and task conditions.

Collaboration and solver feedback improve formulations (CoE, Xiao et al., 2024; OptiMUS, AhmadiTeshnizi et al., 2026); synthetic data and fine-tuning strengthen modeling knowledge (ORLM, Huang et al., 2025a; LLMOPT, Jiang et al., 2025; OptMATH, Lu et al., 2025). Opt-Miner adds active retrieval (Liu et al., 2026a). Evaluation reaches multi-file revision in OR-Space (Zhou et al., 2026) and end-to-end solving in ORAgentBench (Li et al., 2026a), motivating tests of external-rule discovery and use.

External information must become constraints and objectives governing feasible decisions. Unverified conditions can cause missing rules or unjustified restrictions of the feasible set. For example, omitting an applicable sales restriction can invalidate a procurement plan even when products, benefits, and budget are specified (Figure 1). Applicability verification must determine which mathematical relations require revision and when the formulation should be retained.

We introduce SearchWorthyOR (SWOR) to evaluate external-rule discovery and use in modeling. SWOR covers 10 optimization scenario categories with rules from public regulations and policies. Three paired conditions—rule inapplicable, applicable but omitted, and applicable and provided—test applicability and rule visibility through source evidence, model modifications, and reference decisions. We also propose SearchWorthy Agent (SWAgent), whose information–evidence table

[1] https://anonymous.4open.science/r/SearchWorthyOR-1425

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.

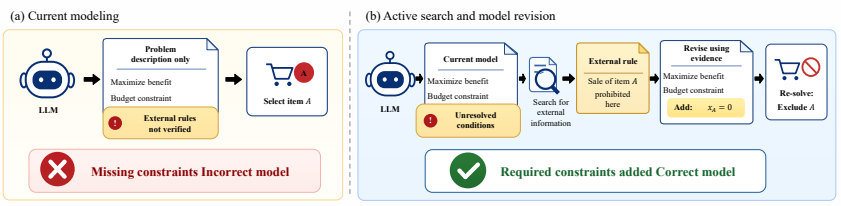

Figure 1: External rules can change the model. Direct modeling omits a necessary constraint; active discovery supplies evidence for revision. The sales restriction is hypothetical.

connects conditions, assessments, sources, and modeling effects. This shared state guides when to search, how to fuse evidence, and whether to revise or retain the model before solving.

Our main contributions are as follows: 1) We formulate proactive discovery of unstated external rules as an information-completeness problem and introduce **SWOR**, pairing rule-inapplicable, rule-omitted, and rule-provided conditions with disjoint optimal-action sets before and after applying an applicable rule; 2) We propose **SWAgent**, whose information–evidence table couples instance applicability and source support to constraint or objective effects, guiding search and justifying formulation revision or retention; 3) Across all 100 source tasks, SWAgent achieves **79.00%** answer accuracy, leading the compared workflows by **11.33** percentage points, while paired rule-visibility tests reveal a 36–65-point accuracy gap in base LLMs.

2 Related Work

2.1 LLM-Based Optimization Modeling and Evaluation

LLM modeling uses prompting or domain-specific training. Prompted workflows include CoE's expert collaboration (Xiao et al., 2024), OptiMUS's modular correction (AhmadiTeshnizi et al., 2026), Opt-Verifier's model and solution checks (Liu et al., 2026b), Monte Carlo tree search (Astorga et al., 2025), OptiTree's hierarchical retrieval (Liu et al., 2025), Agora-Opt's memory-augmented debate (Lin et al., 2026a), and LEAN-LLM-OPT's structured workflow (Liang et al., 2026). ORLM (Huang et al., 2025a), LLMOPT (Jiang et al., 2025), and OptMATH (Lu et al., 2025) use fine-tuning and synthetic data. MIND (Liu et al., 2026c) and StepORLM (Zhou et al., 2025) explore error-driven learning and process supervision. ORPilot adds requirements interviews and data collection (Xie, 2026). Evaluation spans extraction and formulation in NL4Opt (Ramamonjison et al., 2022), multi-file modeling in OR-Space (Zhou et al., 2026), and end-to-end solving in ORAgentBench (Li et al., 2026a). SWOR evaluates external-rule discovery and applicability in optimization modeling.

2.2 Active Information Acquisition and Knowledge Use

ReAct (Yao et al., 2023) and IRCoT (Trivedi et al., 2023) interleave reasoning and acquisition. FLARE (Jiang et al., 2023), AdaSearch (Lin et al., 2026b), and S2G-RAG (Li et al., 2026b) study retrieval timing, query selection, and evidence sufficiency. Opt-Miner learns active retrieval through data synthesis and reinforcement learning (Liu et al., 2026a). MIRROR retrieves modeling and coding experience (Shi et al., 2026). Interactive acquisition includes ORPilot's interviews (Xie, 2026), preference elicitation (Drossman et al., 2026), and SAGE-Agent's uncertainty-guided clarification (Suri et al., 2026). CoE coordinates experts (Xiao et al., 2024), and the evaluated Codex-Native-Search workflow searches and solves autonomously (OpenAI, 2026b). SWAgent tracks condition-specific applicability, supporting evidence, and constraint or objective effects. Consistent, resolved entries with no pending conditions guide stopping and justify both revision and retention.

3 Problem Formulation

P comprises public task facts and a revisable initial formulation. Objects, jurisdiction, time, and activities define the modeling conditions:

$$\mathcal{C}(P) = \{c_1, \ldots, c_m\}. \tag{3.1}$$

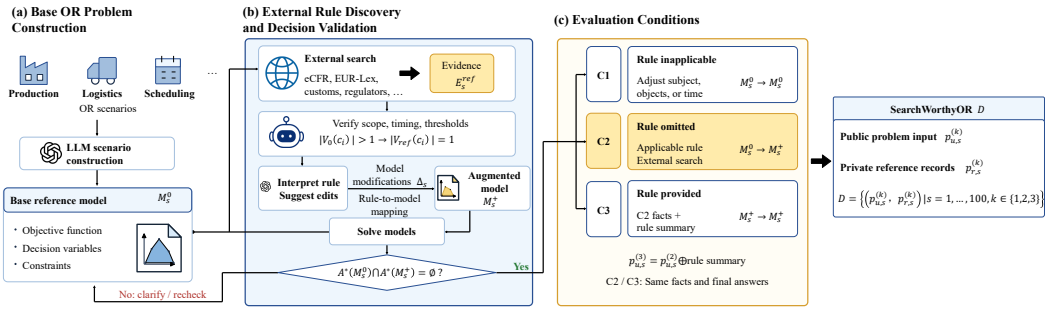

Figure 2: SWOR construction: verified rules change base models and optimal decisions. Applicability and visibility controls yield C1–C3. The clarify/recheck loop belongs to initial construction of the same task. Final evidence completion preserves the finalized problems, reference models, and answers.

Conditions may affect variables, constraints, or the objective. Let E_t contain external evidence and source records used through round t, with $E_0 = \varnothing$. Compatible answers are

$$V_t(c_i) = \{v \mid \text{Compatible}(c_i = v \mid P, E_t)\}\,. \tag{3.2}$$

Compatibility requires agreement with public facts, applicable evidence, and other conditions, allowing the initial formulation to change. Answers with the same modeling effect count as one element. V_t includes all compatible answers, even those not yet enumerated. Cardinalities 1, greater than 1, and 0 denote resolution, uncertainty, and conflict. Resolution may establish non-applicability. Completeness requires consistent information and resolution of every condition:

$$\text{Complete}_{\mathcal{C}(P)}(P, E_t) \iff \text{Consistent}(P, E_t) \wedge [\forall i,\ |V_t(c_i)| = 1]\,. \tag{3.3}$$

Consistency excludes unresolved factual or rule conflicts. It requires neither model feasibility nor a unique optimal solution. After checking existing information, the agent searches only for unresolved, publicly verifiable conditions:

$$\begin{aligned}(P, E_t, c_i) \xrightarrow{\text{generate query}} q_t \xrightarrow{\text{search and read}} D_t \\ \xrightarrow{\text{verify and update}} E_{t+1} \xrightarrow{\text{infer condition}} V_{t+1}(c_i).\end{aligned} \tag{3.4}$$

Queries q_t retrieve source text D_t. Verification checks objects, jurisdiction, time, and exceptions. Verified findings update assessments, may reveal further conditions, and guide revision. Once information is complete, the agent stops searching and solves the model.

4 SearchWorthyOR: Benchmark and Evaluation Framework

SWOR evaluates external-rule discovery and evidence-grounded modeling on 100 source tasks under three conditions, yielding 300 Chinese problems. Let $\mathcal{D} = \{(p_{u,s}^{(k)}, p_{r,s}^{(k)}) \mid s = 1, \ldots, 100,\ k \in \{1, 2, 3\}\}$, where p_u contains public problems and output requirements, and p_r private references. Source-task identifiers and condition labels serve only construction and scoring.

4.1 Benchmark Construction

Scenario backgrounds draw on NLP4LP (AhmadiTeshnizi et al., 2023), MAMO-ComplexLP (Huang et al., 2025b), and team scenarios across 10 categories. Business parameters, models, and answers are constructed anew, with external rules from eCFR, EUR-Lex, and regulatory agencies (Appendix A.7).

Omitting an applicable rule from an independently solvable scenario produces C2. Its public description $p_{u,s}^{(2)}$ and initial formulation constitute P (Section 3). Task actions, costs, benefits, resources, and capabilities define the base reference model:

$$M_s^0: \quad \text{opt}_{x \in \mathcal{X}_s^0} f_s^0(x). \tag{4.1}$$

Table 1: Evaluation conditions and reference-model relationships.

Condition	Rule and problem description	Initial to final model	Reference modification
C1: Rule inapplicable	Facts place the instance outside the rule's scope	$M_s^0 \to M_s^0$	$\varnothing$
C2: Rule omitted	The rule applies but is omitted from the description	$M_s^0 \to M_s^+$	Δ_s
C3: Rule provided	C2 facts with an additional rule summary	$M_s^+ \to M_s^+$	$\varnothing$

Here, x denotes decisions, $\mathcal{X}_s^0$ the feasible region, f_s^0 the objective, and opt minimization or maximization. LLM-assisted construction supplies business facts, interprets rules, and proposes modifications. Programmatic checks verify record consistency and changed optimal actions.

Queries use subjects, objects, jurisdiction, dates, and activities. Official texts and cross-references establish scope, effective dates, thresholds, and exceptions. Checks distinguish transaction-level from annual totals and strict from inclusive thresholds. Records retain URLs, authorities, passages, conditions, and access times. Reference evidence E_s^{ref}, distinct from runtime evidence E_t, resolves a publicly verifiable condition:

$$|V_0(c_i)| > 1 \quad \xrightarrow{E_s^{\mathrm{ref}}} \quad |V_{\mathrm{ref}}(c_i)| = 1. \tag{4.2}$$

With $E_0 = \varnothing$, V_{ref} evaluates Equation (3.2) using P and reference evidence, excluding private answers and labels. Nonpublic business facts remain explicit in the problem. Final evidence completion and cross-review cover all 100 tasks before evaluation, preserving their finalized problems, models, and answers.

Supported rules become modifications Δ_s to variables, constraints, or the objective. The augmented model must change optimal decisions:

$$\begin{aligned} M_s^+ &= \mathrm{Apply}(M_s^0, \Delta_s), \\ \mathcal{A}^\star(M_s^0) \cap \mathcal{A}^\star(M_s^+) &= \varnothing. \end{aligned} \tag{4.3}$$

Apply executes modifications linked to sources and task facts. $\mathcal{A}^\star(M)$ contains all optimal public-action combinations. Initial scenario design preserves rule meaning. Disjointness excludes every base optimum from the augmented optimum set (Appendix A.8).

4.2 Evaluation Conditions and Information Boundaries

From C2, we derive C1 by changing applicability facts while retaining the base optimization structure. C3 retains C2 facts and adds an original-source rule summary. Ignoring problem identifiers,

$$p_{u,s}^{(3)} = p_{u,s}^{(2)} \oplus \text{rule summary}_s. \tag{4.4}$$

The appended summary states scope, timing, thresholds, obligations, and exceptions, without model modifications or answers. Table 1 gives the reference relationships. The initial model represents all public information and is not supplied directly to the agent. In C3 the supplied rule is already part of P, with $E_0 = \varnothing$.

C1/C2 test applicability. C2/C3 share facts and answers, isolating rule visibility. Independent contexts receive only p_u, with task facts, output requirements, and condition-blind identifiers. Private p_r retains evidence, applicability, models, modifications, optimal actions and values, condition pairings, and scoring records. Construction checks their separation and consistency. Runtime inputs contain no missing-rule labels or task-specific search instructions. Acquisition and model updates are logged, with source tasks as paired statistical units.

4.3 Evaluation Protocol and Metrics

We report six answer, model, and search metrics, with answer accuracy primary. Let $\mathcal{I}$ contain N problems and $\mathcal{I}_{12}$ its N_{12} C1/C2 problems. Overall, per-condition, and C1/C2 denominators are 300, 100, and 200 (Section 6.1), including failures and missing outputs.

Answers and models. Let $\hat{x}_i$ be the predicted full action vector, $\mathcal{A}_i^\star$ all reference-optimal actions, $\hat{z}_i$ the predicted objective, and $z_i^\star$ an accepted reference value:

$$\begin{aligned} A_i &= \mathbf{1}[\hat{x}_i \in \mathcal{A}_i^\star], \\ O_i &= \mathbf{1}[\text{correct direction and units } \wedge\ |\hat{z}_i - z_i^\star| \le 10^{-6}]. \end{aligned} \tag{4.5}$$

Here $\mathbf{1}$ is the indicator. Actions cover each public identifier once with integer, non-Boolean values. Invalid or missing actions give $A_i = 0$. Invalid or missing objective fields, including nonfinite values, give $O_i = 0$. With $Y_i = A_i O_i$, any reference-optimal combination earns credit independently of search.

After mapping to public action variables, $G_i = 1$ requires equal complete feasible action sets, objective directions, units, and values on every feasible action (absolute tolerance 10^{-7}). Matching only the optimal answer or the text of a modification is insufficient. Otherwise $G_i = 0$, retaining a separate flag for unresolved verification. Let $W_i = 1$ denote a confirmed model error and 0 otherwise. We compute

$$\begin{aligned} \mathrm{Acc}_{\mathrm{ans}} &= \frac{1}{N}\sum_{i\in\mathcal{I}} Y_i, \\ \mathrm{Acc}_{\mathrm{model}} &= \frac{1}{N}\sum_{i\in\mathcal{I}} G_i, \\ \mathrm{Err}_{\mathrm{model|ans}} &= \frac{\sum_{i\in\mathcal{I}} Y_i W_i}{\sum_{i\in\mathcal{I}} Y_i}. \end{aligned} \tag{4.6}$$

Conditional error uses per-instance intersections and is undefined without correct answers. Full-model verification combines exhaustive public-action evaluation, conditional solver checks for auxiliary variables, and independently checked complete-program enumeration (Appendix A.3). C1–C3 answer accuracies and unresolved-model counts are reported separately.

Search and evidence. Let n_i count issued logical retrievals, including failed operations but excluding duplicate transport retries. $S_i = 1$ denotes executed search and $S_i = 0$ confirmed absence. Reference behavior is $R_i = \mathbf{1}[k_i \in \{1, 2\}]$: C1/C2 require verification. C3 supplies the rule. For C1/C2, $C_i = 1$ requires material visible during the run to cover every reference quotation under the same official URLs. Matching normalizes layout and punctuation while retaining numbers and comparisons. Partial coverage or absent search gives $C_i = 0$. Thus,

$$\begin{aligned} \overline{n}_{\mathrm{search}} &= \frac{1}{N}\sum_{i\in\mathcal{I}} n_i, \\ \mathrm{Acc}_{\mathrm{search}} &= \frac{1}{N}\sum_{i\in\mathcal{I}} \mathbf{1}[S_i = R_i], \\ \mathrm{Joint}_{\mathrm{evidence,model}} &= \frac{1}{N_{12}}\sum_{i\in\mathcal{I}_{12}} C_i G_i. \end{aligned} \tag{4.7}$$

Search counts are per problem and rates are percentages. Dashes denote disabled search. Unknown execution earns no behavior credit. Incomplete counts are lower bounds with reported coverage. C3-provided rules are not acquired evidence. Trajectory review assesses semantic support beyond the joint textual-coverage criterion, including equivalent sources. Appendix A details scoring, coverage, and resources.

5 METHOD

SWAgent binds instance conditions to evidence and mathematical effects through a shared information–evidence table (Figure 3). The initial formulation guides acquisition and justified revision or retention.

5.1 INITIALIZATION AND THE INFORMATION–EVIDENCE TABLE

From public input, the LLM builds $\widehat{M}_0$ for checking, compilation, and Gurobi solving, yielding y_0. The model and description form P, with $E_0 = \varnothing$. Conditions capture objects, jurisdiction, time,

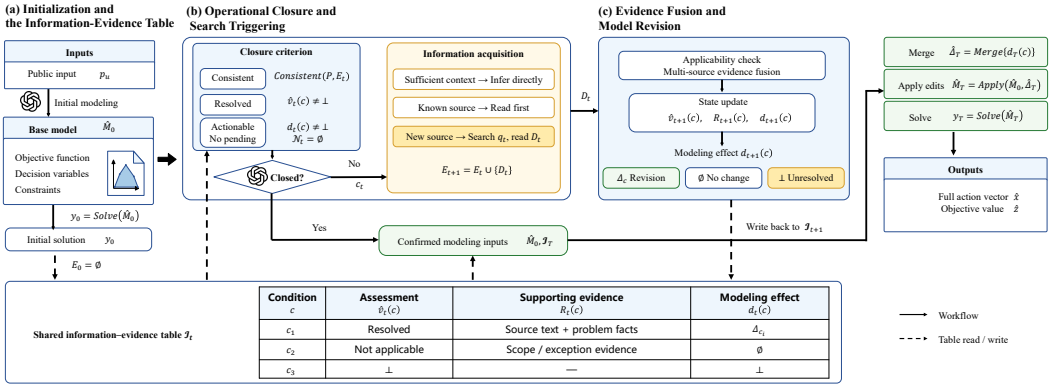

Figure 3: SWAgent workflow. (a) Initial modeling and solving establish the base formulation and shared information–evidence table. (b) Operational closure checks consistency and resolved records, directing inference, reading, or search. (c) Applicability verification supports revision or justified retention, leaving unsupported effects unresolved. After closure, effects are merged and applied before re-solving. New gaps return to the loop. Solid arrows show workflow and dashed arrows table access. Consistency is LLM-assessed (5.3).

and activities:

$$\mathcal{I}_t = \{(c, \widehat{v}_t(c), R_t(c), d_t(c)) \mid c \in \widehat{\mathcal{C}}_t\}, \tag{5.1}$$

$$(\widehat{v}_{t+1}(c), R_{t+1}(c), d_{t+1}(c)) = \mathrm{LLM}(P, E_{t+1}, \mathcal{I}_t, c). \tag{5.2}$$

Each record stores an assessment $\widehat{v}_t(c)$, evidence $R_t(c)$, and revision or retention effect $d_t(c)$. Joint LLM updates undergo programmatic citation checks, with $\perp$ marking unresolved entries.

5.2 Operational Closure and Search Triggering

Operational closure implements Equation (3.3)'s consistency and resolution requirements on identified conditions $\widehat{\mathcal{C}}_t$. Each round registers new conditions $\mathcal{N}_t$ from actions, definitions, exceptions, and dependencies. The LLM assesses consistency and resolves effects using facts or evidence. The program checks recorded states and citations:

$$\begin{aligned} \widehat{\mathrm{Closed}}_t \iff \widehat{\mathrm{Consistent}}_t \wedge \mathcal{N}_t = \varnothing \\ \wedge \left[\forall c \in \widehat{\mathcal{C}}_t,\ \widehat{v}_t(c) \neq \perp \wedge d_t(c) \neq \perp\right]. \end{aligned} \tag{5.3}$$

For unresolved c_t, the LLM identifies gaps and public sources from existing materials. The program prioritizes unread sources. A new search uses

$$\begin{aligned} q_t &= \mathrm{LLM}(P, E_t, \mathcal{I}_t, c_t), \\ D_t &= \mathrm{Read}(\mathrm{Search}(q_t)). \end{aligned} \tag{5.4}$$

Successfully read text and provenance enter $E_{t+1} = E_t \cup \{D_t\}$. Queries verify regime, scope, timing, thresholds, and exceptions. Snippets only locate sources. Missing nonpublic facts or exhausted budgets leave conditions unresolved.

5.3 Evidence Fusion and Model Revision

The LLM checks applicability and reconciles source versions, times, and scopes through Equation (5.2). Candidate effects relative to the initial model are

$$d_{t+1}(c) = \begin{cases} \widehat{\Delta}_c, & \text{evidence supports a specific model modification,} \\ \varnothing, & \text{evidence supports retaining the model,} \\ \perp, & \text{the answer or modeling effect remains unresolved.} \end{cases} \tag{5.5}$$

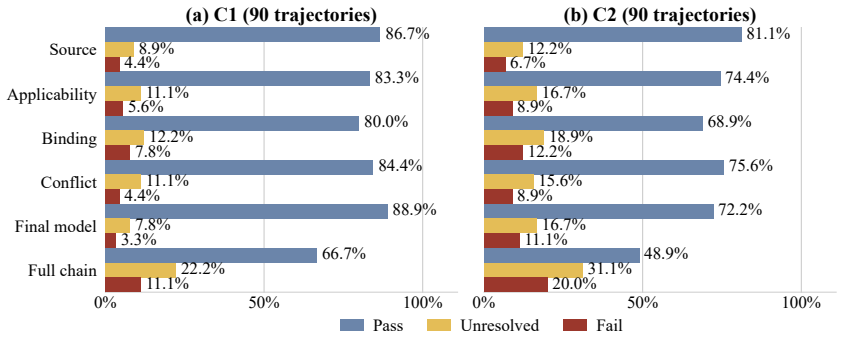

Figure 4: Semantic verification of SWAgent trajectories. Each panel comprises $30 \times 3 = 90$ trajectories, from the same 30 source tasks across three runs. Bars show percentages for source support, applicability, binding, conflict handling, final-model follow-through, and their conjunction. Exact mathematical correctness is assessed separately (Section 4.3).

Modifications add, replace, or remove constraints or adjust the objective. Binary action 2 requiring action 1 gives $x_2 \leq x_1$. Retention requires evidence of inapplicability or correct representation. Updates return to closure before joint revision.

At round T, the LLM merges valid effects, deduplicates, and handles dependencies. The program applies, checks, compiles, and solves:

$$\begin{aligned} \widehat{\Delta}_T &= \mathrm{Merge}\{d_T(c) : c \in \widehat{\mathcal{C}}_T\}, \\ \widehat{M}_T &= \mathrm{Apply}(\widehat{M}_0, \widehat{\Delta}_T), \\ y_T &= \mathrm{Solve}(\widehat{M}_T). \end{aligned} \tag{5.6}$$

New gaps reopen the loop. y_T contains solver status, available $\hat{x}, \hat{z}$, and condition/source records. Infeasibility alone does not invalidate information closure.

6 Experiments

6.1 Experimental Setup

Setup. SWAgent uses GPT-5.6-luna/xhigh, web search, source reading, and Gurobi on 100 source tasks (300 problems, Appendix C). Metrics average three runs, with source tasks grouping C1–C3 as paired units. Appendix D gives configurations, process metrics, and run-level variation (Table 13).

Baselines. GPT-5.6-luna (OpenAI, 2026c), GPT-6 Astra (OpenAI, 2026a), Claude Opus 5 (Anthropic, 2026), and DeepSeek Flash (DeepSeek, 2026) model directly without retrieval and call Gurobi (Gurobi Optimization, LLC, 2026). Workflows share the GPT-5.6-luna/xhigh label and retain their tool protocols. Standard (formerly Direct) models and solves before optional search and revision. CoE (Xiao et al., 2024) and OptiMUS (AhmadiTeshnizi et al., 2026) do not search. OptiMiner (Liu et al., 2026a) retrieves only arXiv papers without additional training. Codex-Native-Search (OpenAI, 2026b) autonomously searches, reads, and solves.

Metrics. Section 4.3 defines six metrics with their stated denominators. Rates are percentages. Logical searches exclude retries. Failures remain in task-level denominators. Unresolved models earn no correctness credit.

6.2 RQ1: Does Real-World External Information Affect LLM Modeling?

Effect of external-rule availability. Supplying rules in C3 raises all four models' answer accuracy by 36–65 points over C2 (Table 2): GPT-5.6-luna improves from 31.00% to 87.00%, and Claude Opus 5 from 64.00% to 100.00%. C2/C3 share business facts and reference decisions, isolating rule visibility from task difficulty. Moreover, construction requires disjoint optimal-action sets before and after applying the rule: the missing information changes which decisions are optimal. The

Table 2: Base-model performance on 100 paired source tasks (300 problems). C1/C2/C3 report answer accuracy for inapplicable, omitted, and provided rules. C2/C3 share facts and reference answers. $\mathrm{Acc}_{\mathrm{model}}$ checks complete formulations. $\mathrm{Err}_{\mathrm{model|ans}}$ measures confirmed model errors among correct answers. All entries are percentages: $\mathrm{Acc}_{\mathrm{ans}}$ is three-run mean $\pm$ sample SD. Other entries are three-run means.

Base model	$\mathrm{Acc}_{\mathrm{ans}}$	C1	C2	C3	$\mathrm{Acc}_{\mathrm{model}}$	$\mathrm{Err}_{\mathrm{model|ans}}$
GPT-5.6-luna	62.67 ± 1.15	70.00	31.00	87.00	52.33	16.49
GPT-6 Astra	77.33 ± 0.88	79.00	57.00	96.00	67.33	12.93
Claude Opus 5	81.00 ± 0.58	79.00	64.00	100.00	67.67	16.46
DeepSeek Flash	68.33 ± 1.15	76.00	32.00	97.00	59.67	12.68

Table 3: End-to-end workflow comparison on the same 300 problems using GPT-5.6-luna/xhigh and each method's tool protocol. Accuracy columns are percentages. $\overline{n}_{\mathrm{search}}$ counts logical searches per problem, and $\mathrm{Acc}_{\mathrm{search}}$ scores search in C1/C2 and abstention in C3. $\mathrm{Acc}_{\mathrm{ans}}$ is three-run mean $\pm$ sample SD. Other entries are three-run means. Bold marks the highest accuracies. Dashes denote disabled search.

Method	$\mathrm{Acc}_{\mathrm{ans}}$	C1	C2	C3	$\overline{n}_{\mathrm{search}}$	$\mathrm{Acc}_{\mathrm{search}}$	$\mathrm{Acc}_{\mathrm{model}}$
Standard	62.33 ± 0.88	71.00	22.00	94.00	0.610	55.67	57.67
CoE	64.33 ± 1.00	81.00	24.00	88.00	—	—	41.67
OptiMUS	64.00 ± 0.58	78.00	22.00	92.00	—	—	54.33
OptiMiner	64.67 ± 0.33	83.00	17.00	94.00	0.040	36.00	37.67
Codex-Native-Search	67.67 ± 2.00	77.00	41.00	85.00	0.523	59.00	59.67
SWAgent	$\mathbf{79.00 \pm 0.67}$	**93.00**	**45.00**	**99.00**	1.353	**75.33**	**61.33**

gains therefore establish external-rule availability as an OR modeling requirement with measurable decision consequences across model families.

Joint answer and model evaluation. Claude Opus 5 answers a mean 64 C2 problems correctly, with 13 corresponding models incorrect (20.31%). In C3, all 100 answers are correct but 12 models are incorrect. An omitted or overrestrictive constraint can leave the reported optimum unchanged while altering other feasible decisions. Checking the complete feasible action set and objective therefore exposes formulation errors that answer matching alone cannot distinguish. Together, the paired conditions and dual evaluation measure both the need for external rules and their faithful mathematical incorporation.

6.3 RQ2: Can SWAgent Discover and Integrate External Rules?

End-to-end performance. SWAgent reaches 79.00% answer accuracy, leading Codex-Native-Search by 11.33 points (Table 3). Model correctness also ranks first at 61.33%. Its C1/C2/C3 accuracies of 93.00/45.00/99.00% lead that workflow by 16/4/14 points. C1 tests whether a plausible rule should be excluded, C2 requires discovering and incorporating an omitted rule, and C3 tests formulation from supplied evidence. Gains across all three conditions support the combined ability to retain justified models and make consequential revisions.

Evidence acquisition and use. SWAgent attains 75.33% search behavior accuracy, searching in 92.00% of C1 and 97.00% of C2 cases while refraining in 37.00% of C3 cases. Its 1.353 searches per problem accompany condition-dependent acquisition: evidence can justify excluding a rule as well as adding one. C2 accuracy rises from Standard's 22.00% to 45.00%, with 43 mean repairs involving search and accepted nonempty updates (Appendix D.5). This connects acquisition to changes in optimization decisions. The information–evidence table makes the applicability and mathematical effect of retrieved rules explicit.

Semantic evidence-chain verification. All 100 finalized reference packages pass the five-dimensional cross-review before evaluation (Appendix D.8). Separately, two LLMs review 180 C1/C2 SWAgent trajectories from 30 sources, with 104 complete-chain passes (Figure 4). Passes require agreed dimension-wise majorities across three assessments per model (Appendix A.6). The trajectory diagnostic retains its fixed denominator. Main results cover all 100 source tasks.

Table 4: Contributions of evidence state, base modeling, and closure. Variants share the backbone and retain search access (Table 10). Rates are percentages, with search count per problem. $\mathrm{Acc}_{\mathrm{ans}}$ is three-run mean $\pm$ sample SD. Other entries are three-run means. Bold marks the highest rates. The joint rate requires coverage of all reference quotations at reference URLs and exact model correctness on 200 C1/C2 problems per run.

Variant	$\mathrm{Acc}_{\mathrm{ans}}$	C1	C2	C3	$\overline{n}_{\mathrm{search}}$	$\mathrm{Acc}_{\mathrm{search}}$	$\mathrm{Joint}_{\mathrm{evidence,model}}$
SWAgent	**79.00 $\pm$ 0.67**	**93.00**	**45.00**	**99.00**	1.353	**75.33**	**3.00**
w/o Information–Evidence Table	62.33 $\pm$ 0.88	71.00	22.00	94.00	0.610	55.67	0.00
w/o Base Modeling	68.00 $\pm$ 0.88	73.00	35.00	96.00	0.820	63.00	1.00
w/o Closure Criterion	64.33 $\pm$ 1.20	79.00	31.00	83.00	1.030	66.67	2.50

From source support to mathematical effects. Complete-chain passes reach 66.7% for C1 retention and 48.9% for C2 integration, compared with source-support rates of 86.7% and 81.1%. Binding has the lowest individual pass count (134/180): binding must identify the affected variables, constraints, or objective. The distinction explains why retrieving a relevant source and completing an evidence-grounded model are separate achievements. The benchmark spans six rule families and ten categories (Figure 6). Within the 30-source diagnostic, the audit makes justified retention and revision inspectable at the condition level.

6.4 RQ3: How Do the Workflow Mechanisms Affect Performance?

We compare three variants (Table 4): **1) w/o Information–Evidence Table** removes persistent records of conditions and evidence. **2) w/o Base Modeling** acquires information before formulation. **3) w/o Closure Criterion** replaces UPDATE's stopping prompt with an LLM sufficiency judgment. The third variant retains the table, initial modeling, updates, re-solving, and tool budget, adding no judgment call. Appendix D.3 details all configurations.

Persistent state and formulation-guided acquisition. Without the information–evidence table, accuracy trails SWAgent by 16.67 points overall and 23 in C2, the largest overall gap. Persistent records preserve each condition's source, applicability, and mathematical effect across rounds. Without base modeling, the gaps are 11.00 points overall and 20/10 in C1/C2. An initial formulation supplies concrete variables, constraints, and objectives for evaluating evidence, supporting both justified exclusion and necessary revision.

Closure and productive revision. With the table and update machinery retained, the variant without closure trails SWAgent by 14.67 points overall and 16 in C3. Closure checks resolved assessments and mathematical effects on identified conditions. The paired diagnostic records 47 mean repairs and four degradations, a net gain of 43 correct answers (Table 11). Together, these results support the complementary OR roles of formulation-guided acquisition, persistent evidence binding, and justified stopping.

Interpreting the process metrics. The 3.00% joint rate requires quotation coverage and exact model correctness on 200 C1/C2 instances per run (Table 4). The semantic audit checks supported interpretation and binding in 180 diagnostic trajectories (Figure 4, Appendix D.8). These endpoints assess complementary textual, semantic, and mathematical requirements.

7 Conclusion

SearchWorthyOR tests external rules through their applicability and consequences for optimal decisions. SWAgent organizes acquisition around constraints, objectives, and evidence-supported mathematical effects, guiding closure and justified formulation revision or retention. Its leading answer accuracy among the compared workflows and semantic-audit results support persistent records of conditions and evidence for connecting external rules to optimization decisions.

REFERENCES

Ali AhmadiTeshnizi, Wenzhi Gao, and Madeleine Udell. OptiMUS: Optimization Modeling Using MIP Solvers and large language models, 2023. URL https://arxiv.org/abs/2310.06116.

Ali AhmadiTeshnizi, Wenzhi Gao, Herman Brunborg, Shayan Talaei, Connor Lawless, and Madeleine Udell. OptiMUS-0.3: Using Large Language Models to Model and Solve Optimization Problems at Scale, 2026. URL https://arxiv.org/abs/2407.19633.

Anthropic. Introducing Claude Opus 5, 2026. URL https://www.anthropic.com/news/claude-opus-5.

Nicolás Astorga, Tennison Liu, Yuanzhang Xiao, and Mihaela Van Der Schaar. Autoformulation of Mathematical Optimization Models Using LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 1864–1886, 2025. URL https://proceedings.mlr.press/v267/astorga25a.html.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2): 405–421, 2021. doi: 10.1016/j.ejor.2020.07.063. URL https://doi.org/10.1016/j.ejor.2020.07.063.

DeepSeek. Responses API, 2026. URL https://api-docs.deepseek.com/api/create-response/.

Joshua Drossman, Alexandre Jacquillat, and Sébastien Martin. Let's Have a Conversation: Designing and Evaluating LLM Agents for Interactive Optimization, 2026. URL https://arxiv.org/abs/2604.02666.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2026. URL https://docs.gurobi.com/projects/optimizer/en/current/index.html.

Chenyu Huang, Zhengyang Tang, Shixi Hu, et al. ORLM: A Customizable Framework in Training Large Models for Automated Optimization Modeling, 2025a. URL https://doi.org/10.1287/opre.2024.1233.

Xuhan Huang, Qingning Shen, Yan Hu, Anningzhe Gao, and Benyou Wang. LLMs for Mathematical Modeling: Towards Bridging the Gap between Natural and Mathematical Languages, 2025b. URL https://arxiv.org/abs/2405.13144.

Caigao Jiang, Xiang Shu, Hong Qian, et al. LLMOPT: Learning to Define and Solve General Optimization Problems from Scratch, 2025. URL https://arxiv.org/abs/2410.13213.

Zhengbao Jiang, Frank Xu, Luyu Gao, et al. Active Retrieval Augmented Generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7969–7992, 2023. doi: 10.18653/v1/2023.emnlp-main.495. URL https://doi.org/10.18653/v1/2023.emnlp-main.495.

Jiajun Li, Mingshu Cai, Yixuan Li, et al. ORAgentBench: Can LLM Agents Solve Challenging Operations Research Tasks End to End?, 2026a. URL https://arxiv.org/abs/2606.19787.

Minghan Li, Junjie Zou, Xinxuan Lv, Chao Zhang, and Guodong Zhou. S2G-RAG: Structured Sufficiency and Gap Judging for Iterative Retrieval-Augmented QA. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25846–25862, 2026b. doi: 10.18653/v1/2026.acl-long.1185. URL https://doi.org/10.18653/v1/2026.acl-long.1185.

Kuo Liang, Yuhang Lu, Jianming Mao, et al. Large-Scale Optimization Model Auto-Formulation: Harnessing LLM Flexibility via Structured Workflow, 2026. URL https://arxiv.org/abs/2601.09635.

Jianghao Lin, Zi Ling, Chenyu Zhou, et al. From Soliloquy to Agora: Memory-Enhanced LLM Agents with Decentralized Debate for Optimization Modeling, 2026a. URL https://arxiv.org/abs/2604.25847.

Tzu-Han Lin, Wei-Lin Chen, Chen-An Li, Hung yi Lee, Yun-Nung Chen, and Yu Meng. AdaSearch: Balancing Parametric Knowledge and Search in Large Language Models via Reinforcement Learning, 2026b. URL https://arxiv.org/abs/2512.16883.

Haoyang Liu, Jie Wang, Yuyang Cai, Xiongwei Han, Yufei Kuang, and Jianye Hao. OptiTree: Hierarchical Thoughts Generation with Tree Search for LLM Optimization Modeling, 2025. URL https://arxiv.org/abs/2510.22192.

Haoyang Liu, Yuyang Cai, Jie Wang, et al. Opt-Miner: Empowering Information-Seeking Agent with Tree-Guided Data Synthesis for Optimization Modeling. In *Proceedings of the 43rd International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2026a. URL https://miralab.ai/publication/icml-2026-opt-miner/.

Haoyang Liu, Jie Wang, Boxuan Niu, et al. Opt-Verifier: Unleashing the Power of LLMs for Optimization Modeling via Dual-Side Verification, 2026b. URL https://arxiv.org/abs/2605.29556.

Weiting Liu, Han Wu, Yufei Kuang, et al. Automated Optimization Modeling via a Localizable Error-Driven Perspective, 2026c. URL https://arxiv.org/abs/2602.11164.

Hongliang Lu, Zhonglin Xie, Yaoyu Wu, Can Ren, Yuxuan Chen, and Zaiwen Wen. OptMATH: A Scalable Bidirectional Data Synthesis Framework for Optimization Modeling, 2025. URL https://arxiv.org/abs/2502.11102.

OpenAI. GPT-6 Astra: A new generation of intelligence, 2026a. URL https://openai.com/index/gpt-6-astra/.

OpenAI. Codex CLI, 2026b. URL https://github.com/openai/codex.

OpenAI. GPT-5.6 Luna, 2026c. URL https://developers.openai.com/api/docs/models/gpt-5.6-luna.

Rindranirina Ramamonjison, Timothy Yu, Raymond Li, et al. NL4Opt Competition: Formulating Optimization Problems Based on Their Natural Language Descriptions. In *Proceedings of the NeurIPS 2022 Competitions Track*, volume 220 of *Proceedings of Machine Learning Research*, pp. 189–203, 2022. URL https://proceedings.mlr.press/v220/ramamonjison23a.html.

Yifan Shi, Jiayi Wang, Minyi Wu, Ye Fan, Jialong Shi, and Jianyong Sun. MIRROR: A Multi-Agent Framework with Iterative Adaptive Revision and Hierarchical Retrieval for Optimization Modeling in Operations Research, 2026. URL https://arxiv.org/abs/2602.03318.

Manan Suri, Puneet Mathur, Nedim Lipka, Franck Dernoncourt, Ryan A. Rossi, and Dinesh Manocha. Structured Uncertainty guided Clarification for LLM Agents, 2026. URL https://arxiv.org/abs/2511.08798.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10014–10037, 2023. doi: 10.18653/v1/2023.acl-long.557. URL https://doi.org/10.18653/v1/2023.acl-long.557.

Ziyang Xiao, Dongxiang Zhang, Yangjun Wu, et al. Chain-of-Experts: When LLMs Meet Complex Operations Research Problems. In *International Conference on Learning Representations*, pp. 48519–48537, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/d45ee77826332c100a1e15f7765b99ff-Paper-Conference.pdf.

Guangrui Xie. ORPilot: A Production-Oriented Agentic LLM-for-OR Tool for Optimization Modeling, 2026. URL https://arxiv.org/abs/2605.02728.

Shunyu Yao, Jeffrey Zhao, Dian Yu, et al. ReAct: Synergizing Reasoning and Acting in Language Models, 2023. URL https://arxiv.org/abs/2210.03629.

Chenyu Zhou, Tianyi Xu, Jianghao Lin, and Dongdong Ge. StepORLM: A Self-Evolving Framework With Generative Process Supervision For Operations Research Language Models, 2025. URL https://arxiv.org/abs/2509.22558.

Chenyu Zhou, Xinyun Lu, Jiangyue Zhao, Jianghao Lin, Dongdong Ge, and Yinyu Ye. OR-Space: A Full-Lifecycle Workspace Benchmark for Industrial Optimization Agents, 2026. URL https://arxiv.org/abs/2605.28158.

A SearchWorthyOR Construction Examples and Evaluation Details

This appendix supplements Section 4.3 with field definitions, statistical denominators, and exception handling. The task-level metrics use the same 100 source tasks as the main text. Overall and condition-wise accuracies use denominators of 300 and 100, respectively. Conditional ratios use their explicitly defined denominators. Different conditions of the same problem belong to the same source task. Each metric is computed separately within each run, and the reported result is the arithmetic mean over three runs. The semantic audit uses the object-level cohorts defined in Appendix A.6.

A.1 Final Answers and Record Coverage

Field coverage and the answer scorable rate are

$$\begin{aligned} \text{Coverage} &= \frac{\#\{\text{instances with the relevant field observable}\}}{N}, \\ \text{ScorableRate} &= \frac{\#\{\text{instances with both actions and objective observable}\}}{N}. \end{aligned} \tag{A.1}$$

Here, $\#$ denotes the number of instances. Coverage is computed separately for each field. The completeness of one field cannot stand in for that of another.

Observable actions require unique identifiers whose set matches the public output requirements, with integer, non-Boolean values. An observable objective requires a minimization or maximization direction, a finite numerical value, and a string-valued unit. A readable unit is not necessarily a correct unit. Accuracy on observable instances is

$$\text{Acc}_{\text{obs}} = \frac{\#\{\text{instances observable and correct for this item}\}}{\#\{\text{instances observable for this item}\}}. \tag{A.2}$$

When a method does not provide an interface for an item, that item is marked as not evaluable, without inferring the method's capability from its absence. Instances for which an interface is supported but no valid field is returned remain in the denominator of full-set accuracy. If no instance is observable, only accuracy on observable instances is undefined. Failures still count in full-set scoring. Confirmed zero searches, empty modifications, and missing records are stored separately.

A.2 Search Decisions, Execution, and Retrieval Completion

Search-decision accuracy compares the trigger/no-trigger decision from the shared interface with the private reference annotation and divides the number correct by N. Decision coverage divides the number of readable decisions by N. Under the reference annotation, C1/C2 require external verification, whereas C3 requires no additional acquisition of the rule already supplied. This metric measures agreement with the annotation. Final-answer scoring does not require search.

The actual search rate is the number of instances with a confirmed positive search count divided by N. Search execution is not inferred from a trigger decision. Table 12 reports the mean actual search count. Count coverage denotes the fraction with an observable count. If some counts are unknown, the rate represents confirmed search instances and does not classify unknown instances as having no

Table 5: Answer metrics and record coverage.

Metric	Criterion and calculation
Action accuracy	The action identifiers and values are complete and belong to the set of all reference optimal actions. Divide the number correct by N.
Objective accuracy	The direction is correct, and the units and value agree with an accepted reference representation, with absolute numerical error at most 10^{-6}. Divide the number correct by N.
Joint final-answer accuracy	Number of instances with both correct actions and a correct objective divided by N. This is the primary evaluation metric.
Answer scorable rate	Number of instances with observable action and objective structures divided by N. Scorability does not require correctness.
Field coverage	Number of instances with an observable field divided by N, reported separately for actions, objective, status, applicability, modifications, joint modeling fields, and search decisions.
Accuracy on observable instances	The proportion correct among instances for which the relevant item is observable, reported as a supplementary description.

search. Page-open attempts, successfully read pages, and quotations passing source-text matching may be retained as auxiliary records. They are not interpreted as counts of valid evidence.

Search execution and process completion use different denominators:

$$\begin{aligned} \mathrm{SearchRate} &= \frac{\#\{\text{instances with an actual search count} > 0\}}{N}, \\ \mathrm{RetrievalPass} &= \frac{\#\{\text{instances recorded as triggered and retrieval-complete}\}}{\#\{\text{instances recorded as retrieval-triggered}\}}. \end{aligned} \tag{A.3}$$

Retrieval completion is evaluated only on instances with an observed retrieval trigger in the shared interface. The interface reads an explicit retrieval-decision field, or the trigger status if that field is missing. Neither is replaced by the actual search count. Instances with a trigger but no retrieval status, or without satisfaction of the completion criterion, remain in this denominator. The rate is undefined when no instance is triggered.

The completion flag requires a quotation matching the source text of a page actually opened in that round and a model judgment that the material is sufficient. Source-text matching normalizes whitespace only. It does not check whether all reference evidence has been acquired or establish that the quotation supports the final constraints. Methods without a shared decision or completion interface report only verifiable search-execution records and are not assigned the same process states.

A.3 Modeling Decisions and Modifications

Status accuracy, applicability accuracy, exact modification accuracy, and semantic modification accuracy each divide the number correct by N. The three reference statuses correspond to an inapplicable rule with the base model retained, an applicable rule with the model augmented, and a rule already supplied in the problem. Applicability concerns the target rule and is distinct from whether further search is needed.

Exact modification matching first normalizes operation serialization and then compares multisets of operations. Operation order therefore does not affect the comparison, whereas repeated operations do. If the reference modification is empty, an explicitly returned empty list can be correct. A missing field does not count as a correct empty modification. Let TP, FP, and FN denote the total numbers of matched, additional, and omitted operations, respectively, accumulated over all instances. Then

$$\begin{aligned} \mathrm{Precision}_{\Delta} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}, \\ \mathrm{Recall}_{\Delta} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}. \end{aligned} \tag{A.4}$$

Missing modifications provide no matching operations, and nonempty reference operations count as omissions. A metric with a zero denominator is undefined.

Semantic modification comparison applies the predicted modification to the instance's initial model and compares all feasible combinations over the public action variables, the objective direction, and the objective value for every feasible combination, using an absolute numerical tolerance of 10^{-6}. Equality of optimal values alone is insufficient. The presence of a modification field does not ensure that semantic comparison can be performed, so modification coverage and semantic evaluability are defined separately:

$$\text{SemanticCoverage} = \frac{\#\{\text{instances with a completed semantic modification comparison}\}}{N}. \quad \text{(A.5)}$$

The auxiliary modification-based joint diagnostic requires correct status, target-rule applicability, actions, and objective, together with either an exact modification match or semantic agreement. The number satisfying these requirements is divided by N. This diagnostic is distinct from full-model accuracy, the joint evidence and model rate in Section 4.3, and evidence-chain verification.

Recall for applicable rules, R_+, recall for inapplicable rules, R_-, and balanced accuracy are defined by

$$\begin{aligned} R_+ &= \frac{\text{correct applicability decisions on applicable instances}}{\text{number of applicable instances}}, \\ R_- &= \frac{\text{correct applicability decisions on inapplicable instances}}{\text{number of inapplicable instances}}, \\ \text{BalancedAcc} &= \frac{R_+ + R_-}{2}. \end{aligned} \quad \text{(A.6)}$$

Each recall uses all instances in its true class as its denominator, and missing decisions contribute no correct prediction. Within a condition containing only one class, recall for the absent class and balanced accuracy are undefined. Condition-wise aggregation keeps the many empty modifications in C1/C3 from obscuring modification errors in C2.

Full-model semantic verification. The semantic evaluator compares the reference and candidate in the public-action representation. For a public action x, feasibility means that the model admits a feasible completion of any auxiliary variables. Its objective value is the conditional optimum over those completions, in the model's optimization direction and the frozen canonical unit. Auxiliary formulations can therefore be compared without identical variable names or constraint syntax.

Three paths implement this comparison. For models containing only public integer actions, the evaluator enumerates a grid covering both reference and candidate bounds and checks bounds, constraints, and objective values. Covering both domains exposes actions incorrectly admitted outside the reference bounds. For models with auxiliary variables, it fixes each public assignment and uses Gurobi to check for a feasible completion and compute its conditional objective. A complete-program adapter also compares explicitly enumerated finite feasible sets with independent reference-model evaluation. It checks public-action mappings, source hashes, objective reconstruction, and source-level completeness. Generated programs can enumerate their full binary domain, retain feasible assignments, and select the optimum. Their full sets then undergo this independent comparison. Optimization-pruned candidate lists do not qualify as complete feasible sets.

The public grid is capped at 10^6 assignments. Nonfinite candidate bounds undergo global minimum and maximum bound checks before enumeration. Continuous public actions, unsupported model extensions, and checks without a certified result remain unresolved. Conditional audit solves use one thread, a 10-second limit, zero relative and absolute MIP gaps, and feasibility, integrality, and optimality tolerances of 10^{-9}. These evaluator settings are separate from the agent's solve budget. Units follow the frozen unit contract and explicit conversion factors. Feasibility disagreement, a direction mismatch, or an objective difference above 10^{-7} produces a model-error witness. Unresolved checks retain their flag and earn no model-correctness credit. The 10^{-6} tolerance above belongs to the separate semantic-modification diagnostic.

Why answer and model correctness differ. Let two binary actions maximize $2x_1 - x_2$. The reference imposes $x_1 + x_2 \leq 1$, while the candidate omits it. Both have the unique optimum $(1, 0)$ with value 2. The candidate also admits $(1, 1)$ with value 1, which the reference excludes. The answer receives credit, while model equivalence fails with $(1, 1)$ as a counterexample.

Table 6: Mutually exclusive run-outcome categories, applied in order.

Category	Criterion
Correct answer	The final answer is jointly correct.
Budget failure	No credit is earned and a budget failure is recorded.
Other non-OK failure	No credit is earned, no budget failure is recorded, and the outer status is non-OK.
OK without credit	No credit is earned, no budget failure is recorded, and the outer status is OK.

Accounting for every optimal action. Equation (4.3) concerns sets, not individual solver outputs. Maximizing $x_1 + x_2$ subject to $x_1 + x_2 \leq 1$ for binary actions gives optima $(1, 0)$ and $(0, 1)$. Adding $x_1 = 0$ retains $(0, 1)$, so the criterion fails even if separate solves return different actions. Adding $x_1 = x_2$ instead leaves only $(0, 0)$ and excludes every base optimum.

A.4 Run Status and Error Distributions

Run statuses and error records support auditing and do not replace answer scoring. When existing records are summarized, the OK rate and basic-check rate are

$$\begin{aligned} \mathrm{OKRate} &= \frac{\#\{\text{instances with outer status OK}\}}{N}, \\ \mathrm{CheckRate} &= \frac{\#\{\text{instances whose run records pass basic checks}\}}{N}. \end{aligned} \tag{A.7}$$

Basic checks examine correspondence between the public input and experiment configuration, consistency of calls and resource records, and method-specific retrieval budgets, evidence records, solver outputs, and failure records. Wrong answers, format errors, and budget failures can pass these checks when their records are complete. These checks differ from the reference-model quality checks in Section 4.1 and do not validate regulatory interpretation. If methods use different checking procedures, those procedures are described separately and their pass rates are not directly compared.

Run outcomes are assigned to the first applicable category below, in the stated order. The four counts sum to N.

Instances earning no credit are further divided by answer structure into four categories: unscorable, scorable with only the action wrong, scorable with only the objective wrong, and scorable with both wrong. These counts sum to the number of instances earning no credit. The corresponding rates are

$$\begin{aligned} \mathrm{Rate}_{\mathrm{run},b} &= \frac{\text{number of instances in run category } b}{N}, \\ \mathrm{Rate}_{\mathrm{error},b} &= \frac{\text{number of instances in answer-error category } b}{\text{number of instances earning no credit}}. \end{aligned} \tag{A.8}$$

Here, b indexes the listed categories. The latter rate is undefined if every instance earns credit. Raw failure flags are additionally summarized as the number of instances carrying a flag divided by N, including parsing, output-format requirements, solving, retrieval, service interfaces, runtime execution, and configuration deviations. An instance may carry several flags, so their counts cannot be added to obtain the number of failed instances. Statuses and flags describe records. Specific causes require trajectory inspection.

A.5 Resource Accounting

Resource fields distinguish input, output, and total token counts, searches, page opens, model calls, upstream requests, and elapsed time. Table 12 reports logical search usage. The following aggregation contract defines observed and missing counts, means, and totals for additional resource records. Logical model calls and upstream requests including retries are distinct quantities. Observability of one field does not establish observability of another.

For any resource field c, only recorded numerical values are used:

$$\begin{aligned} \mathrm{Total}_c &= \sum_{i:\, c_i \text{ observed}} c_i, \\ \overline{c} &= \frac{\mathrm{Total}_c}{N_{\mathrm{obs}}}. \end{aligned} \tag{A.9}$$

Here, N_{obs} is the observed count for that field, and the missing count $N - N_{\mathrm{obs}}$ is also recorded. Missing usage is not treated as zero, and missing costs are not diluted by dividing by the total number of instances. With no observations, both the total and mean are undefined. Elapsed-time quantiles are

$$\begin{aligned} \mathrm{P50} &= Q_{0.50}(\{t_i\}), \\ \mathrm{P90} &= Q_{0.90}(\{t_i\}). \end{aligned} \tag{A.10}$$

Here, t_i is an observed per-instance elapsed time, and Q_p is the $100p$-th percentile, computed by linear interpolation between adjacent sorted values. With one observation, both quantiles equal that value. With none, they are undefined. If resumed execution leaves end-to-end elapsed time unknown, it remains unknown rather than being replaced by the duration of a partial call.

Evaluation uses three fixed runs, with each instance contributing one result per run. Historical resource accounting sums recorded execution attempts within each fixed run. The observed historical-token total is defined separately:

$$\mathrm{HistoricalTokens}_{\mathrm{obs}} = \sum_{\substack{r:\, \text{deduplicated historical request} \\ \text{with observed usage}}} \mathrm{tokens}_r. \tag{A.11}$$

Historical requests r include failed attempts and retries within the fixed runs and exclude duplicate saved copies and diagnostic runs outside the statistical scope. The corresponding missing count records instances with any unknown historical usage. With partially missing historical usage, the total is an observed lower bound. If all usage is unknown, it is reported as unavailable rather than as a zero-cost sum over empty records.

A.6 Automatic Cross-Review of Evidence and Model Trajectories

Before task evaluation, reference review completes the evidence packages of all 100 fixed source tasks. It checks the supplied interpretation against official source passages, scope, versions, and exceptions. Supporting records are completed before final cross-review, with the 100 task statements, reference models, and answers fixed. The trajectory audit is a separate SWAgent diagnostic on 30 sources across the ten scenario categories. It covers C1/C2 in each of three task runs, yielding $30 \times 2 \times 3 = 180$ trajectories. Main task-level results use all 100 sources. For each frozen reference package or trajectory, GPT-6 Astra and DeepSeek Flash each perform three separate assessment rounds. These rounds are distinct from the three task runs. Trajectory review uses the sources visible to the agent at each model-update request. Subsequently acquired text cannot establish earlier evidence use.

Each assessment labels five dimensions as pass, fail, or unresolved: source support, instance applicability, rule-to-model binding, conflict resolution, and final-model follow-through. The last dimension checks whether the supported interpretation is carried into the final formulation. Exact model correctness remains the separate mathematical test in Appendix A.3. All substantive model changes and retention decisions are within scope. A source-support pass requires a supporting quote from an available source body. A URL or the reference annotation alone earns no support credit. A deterministic span check verifies that the quoted text appears in the supplied body, while semantic relevance is judged by the two models.

A model's label requires at least two matching determinate assessments across its three rounds. The cross-review accepts that label only when both models agree. Disagreements, unavailable source text, and insufficient evidence remain unresolved. Labels are aggregated per dimension before taking their conjunction. The complete-chain rate requires passes on all five aggregated dimensions. Figure 4 reports the fixed-denominator trajectory label distributions. Appendix D.8 provides finalized reference coverage and trajectory diagnostics. Correct answers, successful retrieval calls, and strict URL or quotation matching do not substitute for these semantic checks.

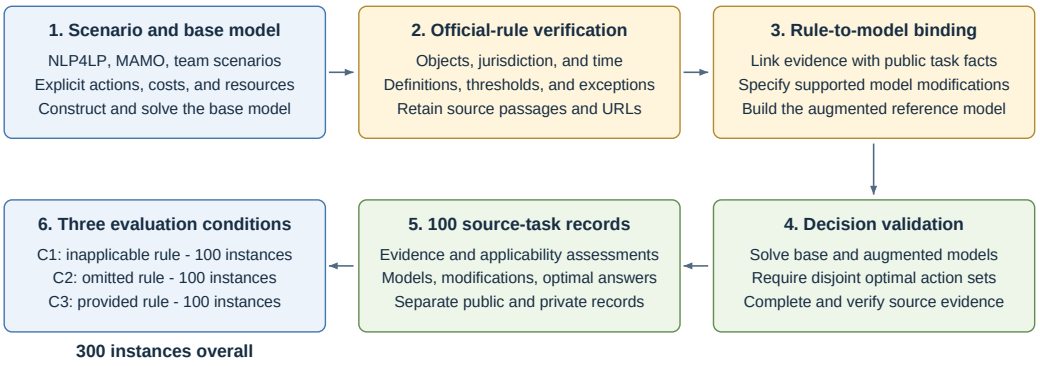

Figure 5: Construction of 100 source tasks and 300 evaluation instances. External-rule verification and decision validation connect each scenario to its reference records before C1–C3 are derived.

A.7 Source-Task Construction

SWOR is constructed as a collection of 100 source tasks spanning 10 optimization scenario categories. Scenario backgrounds and modeling structures draw on NLP4LP, MAMO-ComplexLP, and team-constructed scenarios. The business parameters, reference models, and answers are constructed anew. Figure 5 details how scenario construction, source verification, and decision validation produce the source tasks and their three evaluation conditions.

Scenario and base-model construction. Each task begins with an independently solvable optimization problem. The public description specifies the available actions, costs, benefits, resources, and capabilities needed to formulate the base model M_s^0. Objects, jurisdiction, time, and business activities establish the scope for external-rule discovery. An LLM assists with organizing the scenario and drafting the model. Nonpublic business facts remain explicit in the task rather than being supplied by external search.

External-rule discovery and applicability. Queries use those scope facts to locate official sources, including eCFR, EUR-Lex, and regulatory agencies. Verification follows the source text, definitions, annexes, and cross-references to resolve the relevant objects, effective dates, thresholds, obligations, and exceptions. It distinguishes, for example, a per-transaction threshold from an annual cumulative threshold and a strict inequality from an inclusive boundary. Search snippets locate sources. The reference evidence retains official URLs, authorities, supporting passages, applicability conditions, and available access times. Initial construction may clarify the same scenario and revise its model until decision validation is complete. After finalization, source recovery completes the passages and applicability arguments before the final cross-review. During this final evidence-completion pass, the 100 source identifiers, public problems, reference models, and answers remain fixed.

Rule-to-model binding and decision validation. A supported rule is translated into a modification Δ_s linked to the source passage and the task facts. The rule supplies the external requirement, while the problem supplies the actions, capabilities, and business costs through which that requirement affects the model. Applying Δ_s produces M_s^+. The base and augmented models are solved, and the construction criterion requires their complete optimal-action sets to be disjoint, as in Equation (4.3). Initial scenario design sets business parameters while preserving the meaning of the rule. Programmatic checks verify consistency among citation records, model modifications, and solver results. The LLM assists with source interpretation and candidate modifications.

Three conditions and reference records. Each source task first produces C2, whose public problem contains the applicable task facts but omits the rule. C1 changes the relevant applicability facts while preserving the base optimization structure, so the target rule does not apply. C3 retains the C2 facts and adds a source-based rule summary specifying scope, timing, thresholds, obligations, and exceptions, without revealing the model modification or answer. The 100 source tasks therefore

yield 100 instances per condition and 300 instances overall. Public records contain the problem and output requirements. Private records retain the supporting evidence, applicability assessment, reference models and modifications, all optimal actions and values, and the condition pairings. Construction checks their separation and the consistency of the three conditions.

Coverage beyond business categories. Figure 6 separates business scenarios from the operative requirements and evidence structures they instantiate. Each business category contributes 9–12 source tasks. Grouping by the main operative requirement yields six descriptive rule families: eligibility or authorization (25%), quantitative limits (22%), safety or technical procedures (19%), time or scheduling (15%), prohibitions or restrictions (12%), and reporting or recordkeeping (7%). Source portals and specific applicability roles are multi-label dimensions. Their percentages measure source-task incidence. The recorded single-/multi-hop modes describe evidence reasoning, separately from the number of distinct legal rules.

A.8 An Import-Batch Selection Example

The problem requires selecting exactly three batches from A through F. In this example, the selected batches constitute the complete annual total, with no other existing batches. Their weights are 20, 18, 17, 16, 15, and 14 tonnes, and their values are 24, 23, 22, 21, 20, and 19 points, respectively. Option P provides declaration and certificate-processing capabilities at a cost of 8 points. Let m_j and v_j denote the weight and value of batch j, and let the binary variables x_j and x_P denote selection of that batch and activation of P. The base model from these business facts alone is

$$\begin{aligned} \max \ & \sum_{j=1}^{6} v_j x_j - 8x_P, \\ & \sum_{j=1}^{6} x_j = 3, \\ & x_j, x_P \in \{0,1\}. \end{aligned} \tag{A.12}$$

Within the reference commodity scope and time period, external evidence adds an obligation when the annual cumulative total strictly exceeds 50 tonnes. Combined with P's capabilities stated in the problem, this yields

$$\sum_{j=1}^{6} m_j x_j \le 50 + 5x_P. \tag{A.13}$$

The coefficient 5 follows from the maximum weight of 55 tonnes when exactly three batches are selected. The external rule supplies the 50-tonne boundary, while the problem supplies P's capabilities and cost. The base optimum selects ABC without P, with objective value 69. The augmented optimum selects BCE without P, with weight 50 tonnes and objective value 65. Selecting ABC now requires P and reduces its objective value to 61, so it is no longer optimal. The example shows how an external threshold, task-specified capabilities, and cost jointly determine a model modification and the resulting decision.

For the same source, C1 changes the facts so that the target rule is inapplicable and the reference modification is empty. C2 requires external acquisition of the rule and application of the modification above. C3 supplies the rule summary in the public input, so its initial reference model already contains the constraint and its reference modification is also empty.

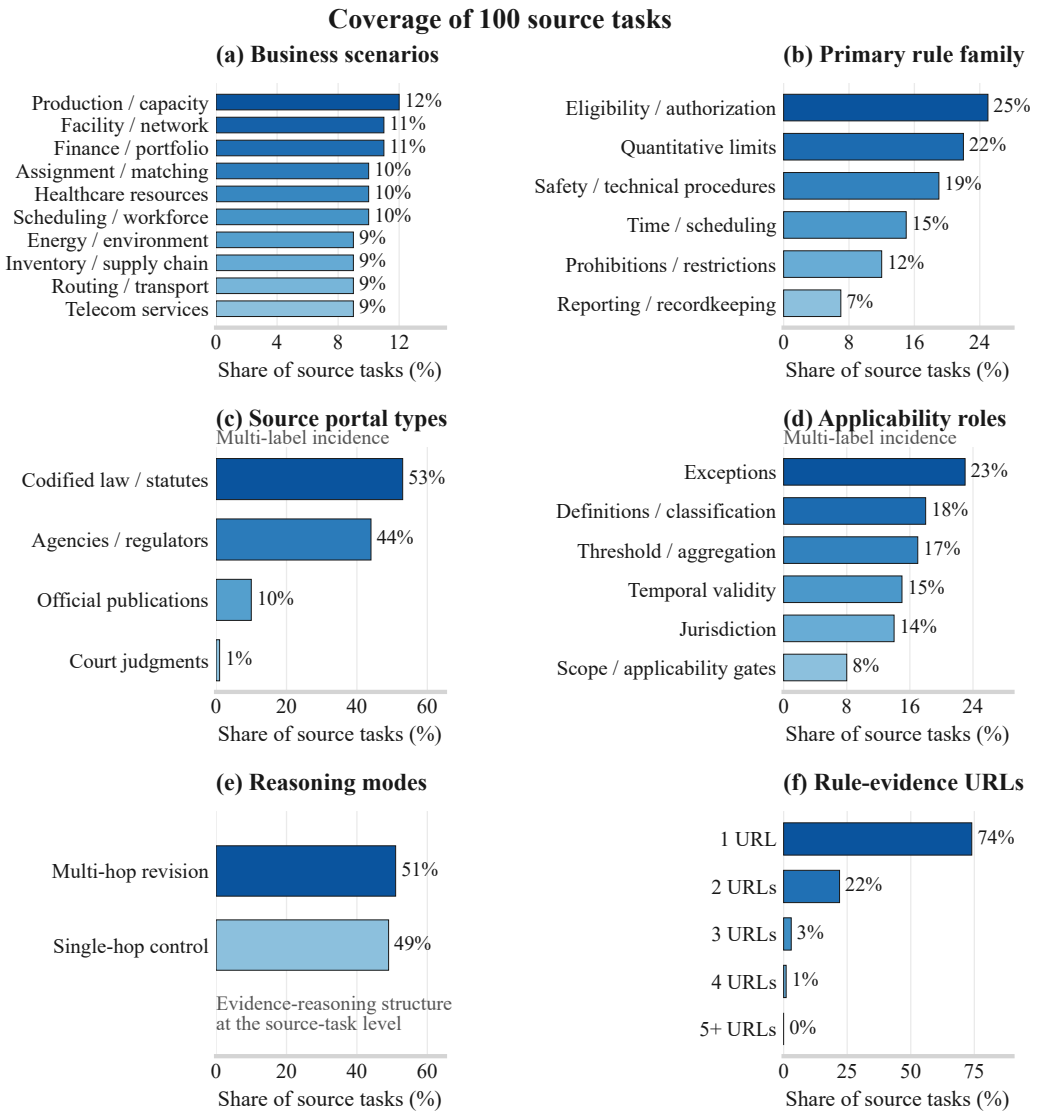

Figure 6: Coverage of the 100 source tasks. Rule families group each task's main operative requirement. Source portals and specific applicability roles are multi-label, so their shares can exceed 100% in total. Untagged roles do not establish absence. Reasoning modes and evidence-URL counts describe recorded evidence structure, rather than a count of independently identified rules.

B SWAGENT IMPLEMENTATION DETAILS

This appendix details the model interface, compilation of binary implications, source acquisition, and condition records used in Section 5. The procedure follows the main text: evidence and candidate modifications are maintained across rounds, and executable model revision is performed after operational closure.

B.1 STRUCTURED MODELS AND THE MODIFICATION INTERFACE

The agent defines decision variables using the action identifiers in the public output contract. Variables may be binary, integer, or continuous and have finite lower and upper bounds. Binary variables retain lower and upper bounds of 0 and 1. Restrictions such as disabling an action are expressed through constraints. The objective is linear, may include a constant term, has a minimization or maximization direction, and uses the canonical unit specified in the public contract.

Supported modifications include adding, replacing, and removing constraints, and replacing the objective. The program assigns an identifier to each added constraint. Replacement and removal target an existing identifier, and removal also requires supporting citations. The variable set and its initial declarations remain fixed in this interface. A modification list may contain several operations in sequence or be empty. During information acquisition, these operations are maintained as candidates in the table. Accepted operations are merged and applied after closure, as in Section 5.3. A conditional linear relation is written as $x_j = \beta \Rightarrow \boldsymbol{\alpha}^\mathsf{T} x \bowtie r$, where the trigger variable x_j is binary, $\beta \in \{0, 1\}$ and $\bowtie \in \{\leq, \geq, =\}$, explicitly specifying the trigger and its mathematical requirement.

Every constraint and objective has a nonempty set of citations. Available identifiers refer to public problem clauses, the output contract, or external source-text windows already presented to the model. The program checks citation identifiers, public variables, finite numerical values, and supported constraint relations. The objective direction must be in the `min/max` enumeration, and its unit must match the public canonical unit. The LLM determines the optimization direction from the problem and explains how the cited text supports the mathematical relation.

B.2 DETERMINISTIC COMPILATION OF BINARY IMPLICATIONS

For $x_j = \beta \Rightarrow \boldsymbol{\alpha}^\mathsf{T} x \bowtie r$, let ℓ_i, u_i be the finite lower and upper bounds of variable i. A term with a nonnegative coefficient reaches its maximum at the upper bound, and a term with a negative coefficient reaches its maximum at the lower bound. The choices are reversed for the minimum. Thus, the bounds of the linear expression over the variable-bound box are

$$\begin{aligned} s_{\max} &= \sum_{i:\alpha_i \geq 0} \alpha_i u_i + \sum_{i:\alpha_i < 0} \alpha_i \ell_i, \\ s_{\min} &= \sum_{i:\alpha_i \geq 0} \alpha_i \ell_i + \sum_{i:\alpha_i < 0} \alpha_i u_i. \end{aligned} \tag{B.1}$$

Let $\kappa_+ = \max(0, s_{\max} - r)$ and $\kappa_- = \max(0, r - s_{\min})$, and define a binary quantity equal to 1 when the trigger is false:

$$h_\beta(x_j) = \begin{cases} 1 - x_j, & \beta = 1, \\ x_j, & \beta = 0. \end{cases} \tag{B.2}$$

The two implication types are compiled as

$$\begin{aligned} x_j = \beta \Rightarrow \boldsymbol{\alpha}^\mathsf{T} x \leq r &\iff \boldsymbol{\alpha}^\mathsf{T} x \leq r + \kappa_+ h_\beta(x_j), \\ x_j = \beta \Rightarrow \boldsymbol{\alpha}^\mathsf{T} x \geq r &\iff \boldsymbol{\alpha}^\mathsf{T} x \geq r - \kappa_- h_\beta(x_j). \end{aligned} \tag{B.3}$$

When the trigger holds, $h_\beta = 0$ preserves the original requirement. Otherwise, $h_\beta = 1$ makes the relaxed inequality hold at every point in the bound box. An equality consequent is split into two inequalities. Compilation uses bounds determined solely by the variable bounds, so the required constants can be computed directly. The compiled model is passed to the fixed Gurobi interface.

For example, the requirement that selecting binary action x_2 also requires x_1 is submitted as $x_2 = 1 \Rightarrow x_1 \geq 1$. From $x_1 \in [0, 1]$, we obtain $s_{\min} = 0$ and $\kappa_- = 1$, giving $x_1 \geq 1 - (1 - x_2) = x_2$, the relation used in Section 5.3. If the problem also imposes $x_1 + x_2 = 1$, the binary domains imply $x_2 = 0$.

B.3 Source Acquisition and Citation Availability

Queries and candidates for the same question. Every acquisition request is associated with an explicit question, either through an existing question identifier or by obtaining or creating a record from explicit question text. The target question's answer is set to unresolved. For a given question, untried candidates are read first. A new search is considered only when no candidate is available for that question. New queries are deduplicated against the entire query history after trimming surrounding whitespace and ignoring case.

The prompt guides the LLM to produce compact English domain queries based on the question to be verified, with a suggested length of at most 200 characters. Programmatic format checks require a nonempty string of at most 320 characters and exclude Chinese characters and specified benchmark- or answer-related forbidden patterns. Every acquisition request must include a valid query, including requests that directly read existing candidates.

Reading attempts candidates in order. After a READ response has been processed, acquisition returns to the update loop as soon as at least one source text is stored. Links actually observed in the source text may be added as candidates for the same question. Ordinary reading failures are recorded and the next candidate is tried. Budget exceptions follow Section B.5.

Storage, windows, and semantic interpretation. Search snippets locate sources. External-rule citations come from the text obtained and presented by READ. The reading layer obtains extracted text from web pages or PDFs through HTTP and saves provenance and execution records. Returned text may be subject to extraction-length limits. Initial source filtering excludes navigation homepages. Accepted text then undergoes applicability review in UPDATE.

Evidence storage is separated from model-visible windows. Short texts are presented directly. For longer texts, verbatim windows are selected according to the question and query, covering obligations, exceptions, and source context near the beginning. Window citations comprise a document identifier and a character interval in the stored extracted text and point to text already presented to the model. Exposed identifiers remain in the available citation set. The evidence state E_t stores external materials and provenance used for condition assessment. UPDATE combines these with the problem to interpret the correspondence of objects, jurisdiction, time, triggers, and exceptions.

Acquisition maintains a complete record H_t and constructs from it a visible execution context $\overline{H}_t$ for UPDATE, containing query history, candidate-source queues, and submission feedback. The controller uses attempted URLs and READ-failure traces in the complete record for deduplication, repeated-state detection, and later review. External text enters the model context through the presented windows.

B.4 Candidate Modifications and Table Maintenance

Joint condition and modification updates. The LLM updates a condition's assessment, supporting citations, and modeling effect together, as in Equation (5.2). The modeling effect is a candidate modification, a supported reason to retain the model, or an unresolved marker. The program checks available citation identifiers, public variable identifiers, finite values, and supported mathematical relations before storing the record. Citation availability establishes which source text was presented. The LLM assesses its applicability and support using the problem facts.

Candidate modifications remain associated with their conditions across rounds. New evidence may revise an assessment, reveal a dependency, or add a condition. Table updates cross-check objects, units, and prerequisites and preserve supporting records for both modifications and retention. A condition stays unresolved if its answer or modeling effect is not determined. Conflicting sources are checked for differences in version, time, and scope. Lack of evidence alone does not support retention.

Closure and executable revision. Each round reviews the scope of the identified conditions, including selected and unselected actions, definitions, exceptions, and dependencies raised by evidence. The operational criterion in Equation (5.3) combines assessed information consistency, resolution of identified conditions and their modeling effects, and the absence of newly identified issues awaiting processing.

After closure, the LLM merges currently valid candidate modifications, combines duplicate relations, and addresses their dependencies. The integration program receives the initial model, the merged modification list, and the evidence supporting each modification. It applies the list to the initial model, checks and compiles the complete formulation, and supplies the full variables, constraints, and objective to Gurobi. If integration identifies a new semantic gap, the gap is registered in the table and the closure check is revisited. Model infeasibility alone does not invalidate operational closure.

Structured responses and records. The interface supports bounded deterministic recovery of JSON punctuation and at most one additional model-based repair for recoverable field errors. Repair uses shared model-call and time budgets and preserves fields outside its authorized scope. Rejection feedback, unresolved records, and solver outputs remain available for review. These interface checks support the structured workflow without replacing the condition and applicability assessments.

B.5 Resource Limits and Final Records

Acquisition under a finite budget. Resource availability is checked before further acquisition. Once the search budget is exhausted, existing candidates may still be read if the relevant reading, model-call, and time budgets permit. A failed reading attempt is recorded and does not become supporting evidence. Unavailable sources, missing nonpublic facts, and exhausted budgets leave the corresponding conditions unresolved. Ending acquisition because of a resource limit is distinct from satisfying the operational closure criterion.

Resource configuration. The model, search, and read limits are 12, 3, and 6, respectively, with a total limit of 1,200 seconds. Each solver call has a limit of 90 seconds and is further constrained by remaining execution time. Read usage counts physical HTTP-request attempts, including retries, redirects, and full-text navigation. Field-level model-based repairs consume the shared model-call budget, whereas deterministic punctuation recovery is local.

Output records. The final record retains the model, solver status, condition assessments, source citations, unresolved issues, and resource information. An available solver solution supplies the full action vector and objective value. Operational closure is the assessment defined by Equation (5.3). Stopping reasons and pending feedback are retained separately to describe the state at delivery.

B.6 Condition Records for a Binary Prerequisite

Consider the Section 5.3 example in which selecting action 2 requires action 1. The condition record links the applicability assessment, the supporting source and problem facts, and the candidate relation $x_2 \leq x_1$. The relation is maintained as a candidate while related conditions are reviewed and is included in the merged modification list after closure. Section B.2 gives its deterministic compilation from a binary implication.

If the rule is inapplicable and has not been imposed, or is already correctly represented, the record instead contains an empty modification together with the supporting assessment and reason. If a required premise remains unknown, the condition and its modeling effect remain unresolved. Rules supplied in C3 are part of the public input and are incorporated during initial modeling.

B.7 Complete Execution Procedure

Algorithm B.1 summarizes the method in Sections 5.1–5.3. The information–evidence table is updated during acquisition. Executable modifications are merged and applied after closure. If integration reveals a new condition, it returns to the same table and closure loop.

Algorithm B.1 SWAgent procedure with candidate modifications and operational closure

Require: Public input p_u and available acquisition and solving resources
Ensure: Model and solver record, condition assessments, citations, and unresolved issues

Generate and compile $\widehat{M}_0$ and obtain $y_0 = \mathrm{Solve}(\widehat{M}_0)$.
Initialize P, $E_0 = \varnothing$, identified conditions, and table $\mathcal{I}_0$.
while the workflow continues within available resources **do**
    Review task scope, current conditions, and issues raised by new evidence.
    Assess consistency and resolution using P, E_t, and $\mathcal{I}_t$.
    if Equation (5.3) establishes operational closure **then**
        $\widehat{\Delta}_T \leftarrow \mathrm{Merge}\{d_T(c) : c \in \widehat{\mathcal{C}}_T\}$.
        $\widehat{M}_T \leftarrow \mathrm{Apply}(\widehat{M}_0, \widehat{\Delta}_T)$, then check and compile.
        $y_T \leftarrow \mathrm{Solve}(\widehat{M}_T)$ and retain the model and solver record.
        If integration reveals a new semantic gap, register it and revisit closure. Otherwise, finish.
    else
        Select an unresolved condition and assess whether current materials suffice.
        If needed and publicly verifiable, read untried relevant sources or generate a query and search.
        Update E_{t+1} with any acquired source text and provenance, or retain E_t.
        Check applicability, versions, exceptions, units, and dependencies against problem facts.
        Update assessments, citations, and candidate modifications in $\mathcal{I}_{t+1}$ using Equation (5.2).
        Register new conditions and preserve unresolved assessments when support is insufficient.
    end if
end while
Return the available solver output, condition and source records, unresolved issues, and stopping state.

C Evaluation Scope and Result Summary

C.1 Evaluation Scope

The evaluation covers all 100 source tasks in SWOR, each with one instance under C1–C3, giving 300 instances overall, 100 per condition, and 200 under C1/C2 in each run. Each configuration is evaluated in three repeated runs. Metrics are computed within each run using its stated denominator, then reported as arithmetic means over the three runs. Counts are likewise mean counts. Failed deliveries remain in the answer denominator. Tables 2–4 report answer, model, search, and ablation results for this scope. Appendix D provides supplementary process diagnostics and search usage.

C.2 Source-Task Results

Averaging SWAgent's source-task histograms over the three runs yields 37 tasks with all three conditions answered correctly and 63 with two conditions answered correctly.

On SWOR, SWAgent has a mean of 237 correct answers per 300-instance run, with mean C1/C2/C3 counts of 93/45/99 and answer accuracy of 79.00%. These results and the main-text comparisons use the same 100 source tasks.

C.3 Scoring and Solver Protocol

Initial and final answers are evaluated against the same frozen action and objective contract. Standard uses Gurobi with one thread, a 90-second limit, and zero relative optimality gap. Answer correctness and full-model correctness are evaluated separately, and paired process diagnostics use only instances for which both initial and final solutions are available. This pairing requirement does not change the denominator of the main answer metric.

D Supplementary Process Metrics and Search Usage

D.1 Metric Definitions

Sections D.1–D.7 evaluate answer correctness, search behavior, full-model correctness, and evidence coverage on the 100-source SWOR benchmark: 300 instances overall, 100 per condition, and

Table 7: Definitions and denominators of the supplementary metrics.

Metric	Criterion and denominator
Answer accuracy	Both the action and the objective satisfy the frozen answer contract. The denominator is 300 overall and 100 per condition. Failed deliveries do not contribute correct answers.
Search-behavior accuracy	Actual search behavior follows the reference requirement: search under C1/C2 and no search under C3. The denominator is 300. Failed searches still count as search execution, not successful evidence acquisition. Technical retries do not increase the binary count.
Modeling accuracy	The complete public-action feasible set, objective direction, and objective value for every feasible action are correct, with absolute tolerance 10^{-7}, and the unit contract is satisfied. Unresolved and known-incorrect models are recorded separately.
Modeling-error rate among correct answers	The number of instances with a correct answer and a model adjudicated incorrect, divided by the number of correct answers. Unresolved models among correct answers are listed separately and are not counted as errors.
Evidence-match rate	For C1/C2, the material visible to the model in the run covers all reference quotations for that instance under the same official URL. Normalization removes differences in layout, whitespace, and punctuation glyphs while preserving numbers and comparison symbols. The denominator is 200. No search or incomplete matching fails this criterion.
Joint evidence and model rate	The number of evidence-matched instances whose final mathematical model is also confirmed correct, divided by 200. Jointly unresolved cases form a separate status and earn no correctness credit. This combines textual evidence coverage with final-model correctness. Interpretation and applicability are assessed separately.

Table 8: Search behavior by condition on the 100-source SWOR benchmark (%).

Method	C1 correct search	C2 correct search	C3 correct non-search
CoE	0.00	0.00	100.00
OptiMUS	0.00	0.00	100.00
OptiMiner training-free	4.00	4.00	100.00
Codex-Native-Search	34.00	45.00	98.00
SWAgent	92.00	97.00	37.00
Direct	34.00	34.00	99.00
SearchFirst	39.00	51.00	99.00
Prompt-as-ClosedJudge	81.00	82.00	37.00

200 under C1/C2. Each configuration is repeated three times. Rates and counts are computed within each run and averaged arithmetically, without pooling instances across runs. Table 7 defines these task-level metrics. Section D.8 uses the object-level audit cohorts defined in Appendix A.6.

C3 supplies rules in the problem statement, so external-evidence metrics cover C1/C2. Evidence matching is a strict textual-coverage test: it preserves numbers and comparison symbols while normalizing layout and typography. Equivalent official sources and alternative wording may provide valid support without matching all reference quotations. The joint evidence and model rate combines this coverage criterion with confirmed final-model correctness. Semantic assessment of source interpretation and instance applicability remains a separate criterion.

D.2 Condition-Level Diagnostics

SWAgent has mean correct-search counts of 92 under C1 and 97 under C2 and a mean correct-non-search count of 37 under C3, yielding a mean search-behavior accuracy of $226/300 = 75.33\%$. This binary behavior metric records whether search occurs. Evidence coverage evaluates the material subsequently obtained.

Table 9 reports model correctness for the 100 C3 instances per method, where the rules are already provided.

Table 9: C3 modeling accuracy on SWOR, where rules are supplied in the problem statement (%).

Method	C3 modeling accuracy
SWAgent	77.00
Direct	82.00
SearchFirst	84.00
Prompt-as-ClosedJudge	60.00

Table 10: Strategy variants and their configurations.

Ablation	Strategy label	Configuration
w/o Information–Evidence Table	Direct	Retains initial modeling, optional retrieval, updates, and re-solving. Removes persistent records linking conditions, evidence, and effects across rounds.
w/o Base Modeling	SearchFirst	Acquires information before constructing the model, removing the pre-search formulation that can guide acquisition.
w/o Closure Criterion	Prompt-as-ClosedJudge	Retains the table, initial modeling, updates, re-solving, and budget. Replaces the stopping prompt with a simple sufficiency judgment in the existing UPDATE call.

Direct has a mean unresolved-model count of one, and its mean confirmed model-correctness proportion is $173/300 = 57.67\%$. Unresolved outputs earn no full-model or joint correctness credit. All final models from the four base LLMs are decidable. The error intersection for answer and model evaluations in Table 2 is evaluated instance by instance.

D.3 Strategy Variants

The SWAgent configuration represented in Tables 3 and 4 performs the initial solve before acquisition, maintains subsequent modifications as candidates during UPDATE, and merges, applies, and solves them after operational closure. Integration gaps return to the same table and closure loop, following Appendix B.

Workflow implementation details. CoE uses a coordinator to organize modeling and programming experts in generating solver code. OptiMUS extracts parameters, the objective, and constraints before generating a mathematical formulation and solver code. OptiMiner retrieves arXiv papers, reads documents, and generates solver code without additional training. Its search scope is restricted to arXiv. The Standard workflow (formerly Direct) first formulates and solves the problem, decides whether to search, then updates the model and solves it again using retrieved information. These descriptions refer to the evaluated configurations. The remaining search-enabled workflows use web retrieval.

The prompt variant uses GPT-5.6-luna/xhigh with `temperature=1`, model/search/read budgets of 12/3/6, a 90-second solver limit, and a 1,200-second per-instance limit. Table 10 maps the main-text ablation names to the executed configurations. These comparisons characterize workflow behavior. Actual search usage appears in Table 12.

D.4 Verification Triggers

On SWOR, SWAgent's mean verification recall, precision, and C3 extra-search rate are 94.50%, 75.00%, and 63.00%, respectively. These statistics describe actual search execution, evaluated separately from an LLM's explicit search judgment.

D.5 Initial-to-Final Answer Changes

Both endpoints use the frozen action and objective contract. Paired counts describe records with both initial and final solves. Main answer accuracies use all 300 instances. On SWOR, SWAgent yields mean counts of 47 repairs and 4 degradations, for a mean net gain of 43 correct answers.

Table 11: Paired initial-to-final answer changes on the 100-source SWOR benchmark, reported as mean counts over three runs. Each run includes instances with both an initial and a final evaluation.

Scope	Method	Paired	Initially correct	Finally correct	Fixed	Broken	Net gain
All 100 sources	SWAgent	299	194	237	47	4	43
All 100 sources	Direct	299	196	187	14	23	-9
All 100 sources	Prompt-as-ClosedJudge	263	170	193	31	8	23

Table 12: Average logical searches per instance on the 100-source SWOR benchmark, averaged over three runs. A dash denotes disabled search.

Method	Average searches
Standard / Direct	0.610
CoE	—
OptiMUS	—
OptiMiner training-free	0.040
Codex-Native-Search	0.523
SWAgent	1.353
SearchFirst	0.820
Prompt-as-ClosedJudge	1.030

Under C2, SWAgent yields a mean of 43 repairs among 99 pairable instances. All repaired instances involve actual search and external-reference markers in accepted nonempty model updates. Full-model correctness provides an additional check beyond successful answer repair.

Across C1 and C3, SWAgent has mean degradation counts of 3 and 1 on SWOR, respectively.

D.6 Search Usage

Table 12 reports the search averages already presented in Tables 3 and 4 for the same 100 source tasks. Within each run, logical retrieval operations are summed over all 300 instances and divided by 300. Transport retries are not counted again. Reported values are arithmetic means of the three per-run averages.

Token usage is defined as the per-instance sum of returned input and output usage, averaged within each run over valid deliveries with available usage records and then across the three runs. Missing usage is not imputed. Codex-Native-Search uses cumulative Codex-session usage, whose accounting scope differs from API usage.

D.7 Repeated-Run Variation

Table 13 separates the three run-level answer accuracies from their arithmetic mean. For percentage-valued accuracies a_1, a_2, a_3, the sample variance is $s_a^2 = \sum_{r=1}^{3}(a_r - \bar{a})^2/(3-1)$, and the sample standard deviation is $s_a = \sqrt{s_a^2}$. SD is measured in percentage points and variance in squared percentage points. Summaries are computed before rounding. Condition-wise means and other metrics retain the main-text definitions and denominators. Repetitions measure run-to-run variation. They do not create additional source tasks.

D.8 Semantic Audit Diagnostics

Final reference evidence and evaluation scope. Before evaluation, evidence completion and final cross-review cover all 100 source tasks. Every finalized package contains the source bodies for all listed reference URLs, supporting passages, and an applicability argument addressing scope, versions, and exceptions. Task statements, reference models, and answers remain fixed throughout

Table 13: Run-level answer accuracy, mean, sample standard deviation, and sample variance. Accuracies are percentages.

Method	Run 1	Run 2	Run 3	Mean	SD	Variance
GPT-5.6-luna	61.33	63.33	63.33	62.67	1.15	1.33
GPT-6 Astra	76.33	77.67	78.00	77.33	0.88	0.78
Claude Opus 5	80.33	81.33	81.33	81.00	0.58	0.33
DeepSeek Flash	67.00	69.00	69.00	68.33	1.15	1.33
Standard	61.33	62.67	63.00	62.33	0.88	0.78
CoE	63.33	64.33	65.33	64.33	1.00	1.00
OptiMUS	63.33	64.33	64.33	64.00	0.58	0.33
OptiMiner	64.33	64.67	65.00	64.67	0.33	0.11
Codex-Native-Search	65.67	67.67	69.67	67.67	2.00	4.00
SWAgent	78.33	79.00	79.67	79.00	0.67	0.44
SearchFirst	67.00	68.33	68.67	68.00	0.88	0.78
Prompt-as-ClosedJudge	63.00	64.67	65.33	64.33	1.20	1.44

this verification. Main results use 300 instances overall, 100 per condition, and 200 under C1/C2. The trajectory diagnostic separately covers 30 sources across ten categories, with C1/C2 in three task runs yielding 180 trajectories. Each model's three assessments examine the same frozen object and are distinct from the three task runs.

Dimension-level outcomes. For the finalized references, pass/fail/unresolved counts are 100/0/0 for source support, applicability, binding, conflict handling, and model follow-through. Their conjunction is also 100/0/0. These are aggregated terminal labels, obtained after evidence completion and dimension-wise cross-review. For the 180 trajectories, the corresponding counts are 151/10/19, 142/13/25, 134/18/28, 144/12/24, and 145/13/22. Complete-chain outcomes are 60/10/20 for C1 and 44/18/28 for C2, giving 104/28/48 overall in Figure 4(a,b). A failed dimension makes the chain fail. Otherwise, any unresolved dimension makes it unresolved. All trajectory outcomes contribute to the fixed diagnostic denominator.

Cross-review and evidence timing. The deterministic span check verifies that a claimed quotation occurs in the supplied source body. GPT-6 Astra and DeepSeek Flash then assess its meaning, applicability, and mathematical effect. Each model's dimension label requires at least two matching determinate assessments, followed by agreement between the two model majorities. The reported complete-chain label is the conjunction of these five aggregated dimension labels. Reference evidence is completed before evaluation and frozen for the final assessments. Trajectory labels use only evidence visible to the agent at the corresponding update. Missing support, unstable labels, and model disagreement remain unresolved in the trajectory diagnostic. Later source recovery supplies no retroactive credit for an earlier update.

Semantic and mathematical endpoints. Reference verification establishes the semantic support used to evaluate the complete 100-source benchmark. Trajectory verification examines whether an agent carries that support into its own decisions. Model follow-through checks whether the supported interpretation reaches the final formulation. Full feasible-set and objective equivalence are evaluated separately by the mathematical checks in Appendix A.3. Likewise, the joint evidence and model rate in Table 4 combines strict quotation coverage with exact model correctness over all 200 C1/C2 instances per run. These textual, semantic, and mathematical endpoints answer distinct questions. Their separate denominators connect reference validity, agent evidence use, and final optimization correctness without conflating the 30-source diagnostic with the main evaluation.

E AI USE STATEMENT

This work uses large language models, including GPT and DeepSeek models, for synthesizing and refining optimization scenarios, retrieving and interpreting public rules, constructing and revising optimization models, and cross-reviewing reference rule packages and modeling trajectories. The corresponding procedures and configurations are described in Sections 4–6 and Appendices A–D.

During research and manuscript preparation, we use generative AI to assist with conceptual and mathematical exposition, methodological and experimental-design feedback, implementation and analysis code, result interpretation, scientific-figure preparation, language editing, and consistency checks. The authors are responsible for reviewing AI-assisted content and deciding whether to incorporate it, and take responsibility for the final text, technical claims, experimental conclusions, and accompanying materials.