ASK BEFORE YOU OPTIMIZE: DYNAMIC PRE-FORMULATION CLARIFICATION FOR INTERACTIVE OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) are increasingly used to formulate optimization models from natural-language problem descriptions, yet realistic operations research (OR) requests are often incomplete: missing objectives, constraints, or business rules can change the resulting mathematical program. Existing evaluations largely assume a complete specification and therefore overlook whether an agent knows when clarification is needed before modeling. We introduce OR-Clarify, a benchmark for pre-formulation clarification. Each task presents a partial public problem description, withholds structured hidden slots, and evaluates agents through bounded interaction with a simulated user. The benchmark supports both open-ended and choice-based clarification, and measures slot recovery, stopping behavior, silent assumptions, and interaction cost. We further propose Interactive Optimization (InterOPT), a two-stage framework that identifies unresolved formulation-critical gaps and uses them to guide whether to ask the next question or to stop. In our choice-based experiments, InterOPT substantially outperforms all baselines in exact slot recovery; in the open-ended setting, it remains competitive with strong prior methods. Together, OR-Clarify and InterOPT reframe OR assistance as a selective completeness decision: clarify when needed, stop when ready, and quantify what remains missing.

1 INTRODUCTION

The structure of an operations research (OR) model—including its objective function, constraints, decision variables, and feasible region—is determined by the business problem specification. With large language models (LLMs), non-experts can now describe problems in natural language and receive a candidate mathematical formulation (Ramamonjison et al., 2023; Shu et al., 2025), lowering the barrier to OR modeling. However, real-world business requests are rarely delivered as complete textbook-like specifications. A user may describe capacities and demands without specifying the objective, or state a routing rule without clarifying whether the time windows are hard or soft. These omissions are not superficial: they can fundamentally alter the mathematical structure of the problem.

The central failure mode in this setting is ***premature formulation***. Most evaluations of LLM-based optimization agents assume a sufficient specification and measure whether the agent can solve or express a given model (AhmadiTeshnizi et al., 2024; Huang et al., 2025), missing the earlier question of whether the available information supports a meaningful formulation. We find that strong LLM agents often declare readiness while core business facts remain unclarified, or silently fill missing facts with unsupported defaults. For example, in a vehicle-routing context, if a user omits whether a courier must return to the origin, an agent that silently assumes a closed tour changes the constraint structure without asking for confirmation.

ORPilot (Xie, 2026) uses interviews within an end-to-end modeling pipeline, while Drossman et al. (2026) study conversational optimization through iterative solution refinement toward stakeholder utility. These systems demonstrate the value of interaction but do not isolate pre-formulation clarification as an evaluation target: whether an agent recovers formulation-critical missing requirements and recognizes when enough information is available to proceed.

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.

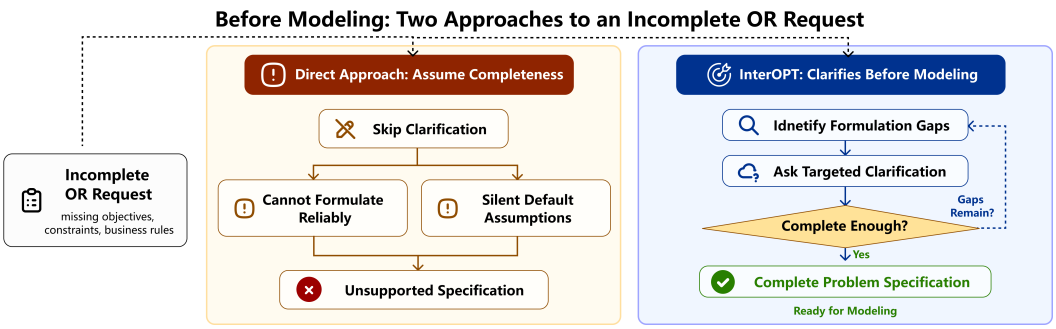

Figure 1: Motivation for pre-formulation clarification.

We study ***pre-formulation clarification***: determining whether a business request is ready for modeling and recovering missing facts that can change the formulation structure. We propose Interactive Optimization (**InterOPT**), which separates this task into two stages: *Dynamic Gap Search* tracks formulation-critical gaps across turns, and *Gap-Guided Action Search* selects the next question or stops.

We introduce ***OR-Clarify***, which evaluates requirement recovery and readiness through incomplete public briefs, fact-bounded simulated users, and hidden-slot rubrics under Open and Choice protocols. To our knowledge, it is the first OR-specific framework to jointly evaluate these capabilities before LLM-driven autoformulation.

In summary, this paper makes three contributions.

(1) The pre-formulation clarification task and the InterOPT framework. We formulate pre-formulation clarification as readiness assessment and recovery of missing formulation-critical facts, and propose InterOPT to separate persistent gap tracking from question selection and stopping.

(2) The OR-Clarify construction and evaluation framework. OR-Clarify converts fully specified optimization tasks into clarification instances by withholding critical facts. It uses the withheld facts to simulate user answers and evaluates whether the agent recovers them under free-form and Choice interaction protocols.

(3) A systematic empirical study of pre-formulation clarification. We compare InterOPT with strong LLM-based baselines and analyze exact requirement recovery, readiness decisions and silent assumptions. The results demonstrate strong recovery gains in the Choice setting and competitive performance in the free-form setting, while the ablations and behavioral diagnostics reveal the roles of gap tracking, question selection, and stopping.

2 Related Work

2.1 LLMs for Optimization Modeling

Recent work studies whether LLMs can translate natural-language problem descriptions into optimization models and solver-ready code. Benchmarks and systems such as NL4Opt, OptiMUS, LLMOPT, and ORLM evaluate model generation, solver integration, and domain adaptation (Ramamonjison et al., 2023; AhmadiTeshnizi et al., 2024; Shu et al., 2025; Huang et al., 2025), and production-oriented systems such as ORPilot organize modeling into structured pipelines spanning interview, data collection, code generation, and execution (Xie, 2026). Most prior benchmarks assume complete specifications. ORPilot supports clarification through interviews, but does not directly evaluate hidden-slot recovery or readiness.

Recent work extends OR modeling through process supervision (StepORLM), memory-enhanced agent coordination (Agora-Opt), and structured workflows (LEAN-LLM-OPT) (Zhou et al., 2025; Lin et al., 2026; Liang et al., 2026). OR-Space further evaluates model building, revision, and explanation in persistent workspaces (Zhou et al., 2026). Pre-formulation clarification remains un-

derexplored as a standalone target: identifying formulation-critical missing facts and deciding when the specification is ready for modeling.

2.2 Clarification, Elicitation, and Abstention

Clarifying-question research studies how agents ask questions when user intent is underspecified, with datasets and methods for conversational retrieval (Aliannejadi et al., 2019; 2020), ambiguity resolution in open-domain QA (Min et al., 2020), and selective clarification (Kuhn et al., 2022). Related settings include follow-up questions about missing rule conditions in conversational machine reading (Saeidi et al., 2018) and option-based clarification of alternative interpretations in CAmbigNQ (Lee et al., 2023).

Preference-elicitation work asks informative questions to improve downstream decisions (Li et al., 2025), teaches models to ask better clarifying questions (Andukuri et al., 2024), and optimizes multi-turn trajectories (Dou & Liu, 2026; Zhang et al., 2025). Recent work studies clarification value (IntentSim), missing-intent and false-premise settings (AskBench), and structured uncertainty for question selection (SAGE-Agent) (Zhang & Choi, 2025; Zhao et al., 2026; Suri et al., 2026). In the optimization context, Drossman et al. (2026) showed that conversational interaction improves solution quality over one-shot submission. These methods provide close points of comparison because they treat questioning as active information gathering. Still, their targets are general user intent or preferences, whereas a question here is correct only if it recovers a business fact that can change the optimization formulation.

Adjacent benchmarks evaluate dialogue through final database states and reliability (τ-bench) (Yao et al., 2025) or schema-level ambiguity resolution in NL2SQL (CLARITY) (Sarwar et al., 2026). In contrast, OR-Clarify evaluates recovery of formulation-critical business requirements, silent assumptions, and readiness before optimization modeling.

Work on abstention evaluates whether LLMs recognize insufficient information (Kirichenko et al., 2026; Kadavath et al., 2022). Pre-formulation clarification additionally requires identifying and eliciting missing business facts that can change the optimization formulation before declaring readiness.

3 Problem Formulation

We study ***pre-formulation clarification***: the task of (1) deciding whether a business request contains enough information to determine a meaningful optimization formulation, and (2) recovering formulation-critical information that remains unresolved before formulation begins.

Let b denote the initial public business brief, and let $\tau_t = \big((a_s, u_s)\big)_{s<t}$ denote the public interaction transcript before turn t, where s indexes preceding interaction turns, a_s is the agent's public clarification action, and u_s is the user's response. Let P_t denote the public problem statement induced by b and τ_t.

Let $\mathcal{B}(P_t)$ be the set of plausible business completions consistent with P_t. For $c \in \mathcal{B}(P_t)$, let $\phi(P_t, c)$ denote the formulation structure induced by completing P_t with c, including the objective, constraints, decision variables, and other formulation-level structures. We say that P_t is *formulation-complete* if and only if all plausible completions induce the same formulation structure:

$$|\{\phi(P_t, c) : c \in \mathcal{B}(P_t)\}| = 1. \tag{3.1}$$

Conversely, P_t is *formulation-incomplete* if plausible completions can induce more than one formulation structure, i.e., $|\{\phi(P_t, c) : c \in \mathcal{B}(P_t)\}| > 1$. Formulation incompleteness therefore refers specifically to unresolved business conditions whose alternative resolutions can change the induced optimization formulation.

At each turn, the agent either asks a clarification question or judges that the current information is sufficient for formulation. If the agent asks, the user's response is appended to the public transcript and induces an updated problem statement P_{t+1}. The interaction terminates when the agent emits `READY_TO_MODEL` or reaches the maximum turn limit $T_{\max}$.

4 OR-CLARIFY BENCHMARK AND EVALUATION FRAMEWORK

Each case i represents the setting as a public–private tuple $(b_i, \mathcal{F}_i, \mathcal{H}_i)$, where b_i is the public business brief, $\mathcal{F}_i$ is the source-grounded fact set withheld from the agent, and $\mathcal{H}_i = \{h_{i,j}\}_{j=1}^{m_i}$ is the set of m_i hidden slots that are grounded in $\mathcal{F}_i$ but absent from b_i.

During interaction, the agent receives b_i as its initial input, while the simulated user consults $\mathcal{F}_i$ to answer clarification questions. After interaction, the judge evaluates recovery by applying the hidden-slot annotations $\mathcal{H}_i$ to the public dialogue transcript. Specifically, $\mathcal{F}_i$ provides the underlying business facts, whereas $\mathcal{H}_i$ identifies the withheld requirements to be evaluated and specifies their recovery criteria. For example, if a private fact requires all vehicles to return to the depot, the corresponding hidden slot specifies how a targeted question or explicit assumption check about depot return is recognized as a recovery hit.

4.1 BENCHMARK CONSTRUCTION FRAMEWORK

Each case is built from a source-supported private case state, decomposed into individual facts describing the business setting, numerical inputs, objective, operational constraints, and modeling assumptions. The business setting and numerical inputs remain visible. Objective, constraint, and assumption facts are eligible for masking unless their mathematical meaning is already determined by the visible information. For instance, in a multi-period production-planning problem, 15,000 available production hours already implies a capacity upper bound, while a demand of 1,000 units does not determine whether it must be met exactly or may be backlogged.

A deterministic pseudorandom procedure with a fixed, case-specific seed masks roughly half of the eligible facts; the remaining facts form the public brief b_i. The masking rate and selection are frozen before method evaluation. Each masked fact becomes one hidden slot, annotated with supporting evidence, a fact-bounded simulated-user answer, acceptable-question examples, a semantic recovery rule, and a severity label.

Severity is assigned after masking and does not influence selection: a $\mathrm{P0}$ slot is a blocking condition whose omission can change the problem itself (e.g., whether a route is open or closed); a $\mathrm{P1}$ slot is a substantive modeling condition whose omission can leave the formulation incomplete or materially incorrect (e.g., whether unmet demand is penalized); and a $\mathrm{P2}$ slot is a secondary boundary or interpretive condition that affects modeling fidelity but is less central to the core evaluation (e.g., whether vehicles may be scheduled across day boundaries).

OR-Clarify consists of 100 clarification cases containing 178 hidden slots, with 1–5 slots per case (mean 1.78): 75 $\mathrm{P0}$, 83 $\mathrm{P1}$, and 20 $\mathrm{P2}$ slots. Human auditing verifies slot boundaries, severity labels, answer support, and rubric consistency.

4.2 CONTROLLED INFORMATION BOUNDARY

OR-Clarify enforces a strict information boundary: the agent sees only the public brief and public transcript; the simulated user is instructed to answer only within the private case state and not to volunteer hidden slots; the judge evaluates only the final transcript using $\mathcal{H}_i$ and frozen rubrics; and the protocol monitor records action validity without providing recovery labels. This separation ensures that hidden slots and rubrics are used only for evaluation, not as inputs to the agent.

A slot receives exact-recovery credit only if, before `READY_TO_MODEL`, an agent question or an explicit assumption check semantically identifies that requirement. Facts volunteered without being requested, assumptions introduced only in the final model, vague catch-all questions, and partial matches receive no exact credit. For every slot, the judge records the supporting transcript location along with a `yes`, `partial`, or `no` label. OR-Clarify supports an open/free-form setting, where the agent asks natural-language questions, and a Choice setting, where it poses questions with candidate options.

4.3 EVALUATION PROTOCOL AND METRICS

The evaluation protocol runs N cases with K repeated runs per case and at most $T_{\max}$ turns per run; all metrics are computed per run, averaged within each case, and then averaged across cases. Let

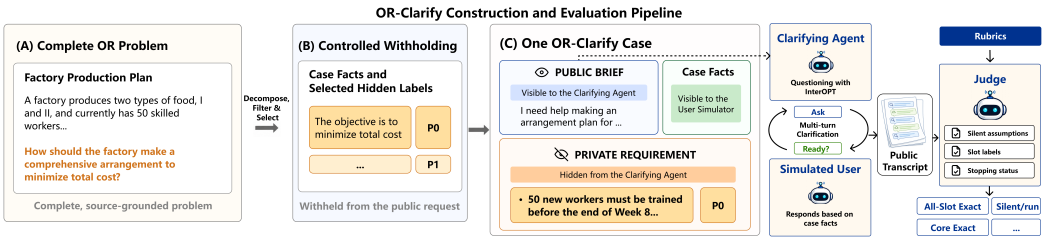

Figure 2: Evaluation workflow and information flow. Hidden slots never enter the evaluated agent. The simulator supplies controlled answers, the passive monitor records protocol behavior, and the post-hoc judge scores the completed public transcript against frozen hidden-slot rubrics.

$\mathcal{H}_i^{\text{core}} \subseteq \mathcal{H}_i$ contain all hidden slots with severity P0 or P1, and let $\mathcal{I}_{\text{core}} = \{i : |\mathcal{H}_i^{\text{core}}| > 0\}$; cases without core slots are excluded from core-based evaluation but remain in the full-slot analysis. For run k of case i, let $z_{i,k,j} = 1$ if hidden slot $h_{i,j}$ is exactly recovered in the final public transcript and 0 otherwise (partial recovery counts as 0). Core Exact counts a run as successful only when all P0/P1 slots of a core-eligible case are exactly recovered:

$$\text{CoreExact} = \frac{1}{|\mathcal{I}_{\text{core}}|} \sum_{i \in \mathcal{I}_{\text{core}}} \frac{1}{K} \sum_{k=1}^{K} \mathbf{1}[\forall h_{i,j} \in \mathcal{H}_i^{\text{core}},\ z_{i,k,j} = 1]\,. \tag{4.1}$$

All-Slot Exact is defined analogously over all N cases, requiring every slot in $\mathcal{H}_i$, including P2, to be exactly recovered. Silent/run counts unconfirmed hidden requirements for which the agent treats a particular value as established in a clarification turn, readiness summary, or final answer. The judge also labels each run's stopping behavior as premature, appropriate, over-questioning, or no-stop.

5 METHOD

We introduce Interactive Optimization (InterOPT), a two-stage framework for formulation-gap-guided clarification prior to optimization. Consider two earlier examples: in production planning, a demand of 1,000 units does not specify whether unmet demand may be backlogged; in routing, the brief may omit whether vehicles must return to the depot. Each omission selects between distinct constraint structures, yet the request can appear model-ready, and silently adopting a default alters the model without user confirmation. We call such a condition a *formulation gap*: an unresolved business condition whose alternative plausible values induce different optimization formulations, such as an objective convention, a feasibility rule, a decision boundary, or a hard-versus-soft policy. Following the monitoring–control view of problem solving (Nelson, 1990), InterOPT separates diagnosing remaining gaps (Stage 1) from deciding whether and what to ask next (Stage 2).

InterOPT grounds this clarification process in optimization modeling through three design choices: (i) *what counts as a pending item*—a business condition searched across optimization components (Section 5.1); (ii) *how questions relate to pending items*—each question must bind to a registered gap when an open entry is available, enforced by runner-level validation instead of relying solely on prompt instructions (Section 5.2); and (iii) *how stopping is guided*—the agent is instructed to stop only when residual uncertainty is confined to notation, raw data, or solver details without affecting formulation structure (Section 5.2). Both stages use only public information (Section 4.2).

5.1 STAGE 1: DYNAMIC GAP SEARCH

At turn t, Stage 1 updates a persistent gap memory $\mathcal{M}_t = \Phi(b, \tau_t, \mathcal{M}_{t-1})$ via a structured LLM call with deterministic validation. Given the public brief, transcript, and previous ledger, Φ searches six categories:
(i) **Objectives and trade-offs**: objective terms, weights, and penalties;
(ii) **Decision scope**: decisions represented in the model;
(iii) **Operational constraints**: the constraint set and feasible region;
(iv) **Time boundaries**: planning horizon and cross-period structure;

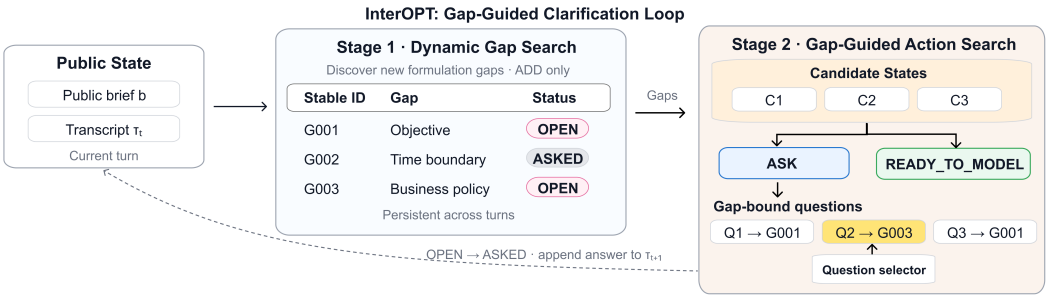

Figure 3: InterOPT gap-guided clarification loop. Stage 1 maintains a public-evidence gap ledger; Stage 2 selects a gap-bound question or READY_TO_MODEL.

(v) **Entity and decision relationships**: coupling between entities or decisions;
(vi) **Hard-versus-soft policies**: mandatory requirements versus penalized violations.

For example, whether unmet demand is penalized changes the objective; whether vehicles must return to the depot changes the constraint set; whether vehicles may be scheduled across day boundaries changes the time structure; and whether demand must be satisfied within each period or may be backlogged changes the demand-satisfaction constraints and cross-period coupling. Searching along these formulation components directs attention toward conditions that alter the mathematical program, filtering out superficial queries about notation or raw data.

Φ proposes at most three additions, each with a category, description, and supporting public evidence; exact duplicates after text normalization are dropped, and the remaining additions receive stable identifiers and the status OPEN. By maintaining an explicit ledger across turns, registered gaps remain directly visible to Stage 2 instead of being lost in the expanding transcript. This persistence is intended to reduce omissions and unsupported default assumptions, while stable identifiers connect subsequent questions to specific gaps. Stage 1 records gaps without ranking them or asking the user. The bindable gaps are $\mathcal{O}_t = \{\ell \in \mathcal{M}_t : \text{status}(\ell) = \text{OPEN}\}$, i.e., registered gaps not yet asked; an empty $\mathcal{O}_t$ does not establish formulation completeness.

5.2 Stage 2: Gap-Guided Action Search

At each turn, the action generator Ψ, instantiated by the tested model, reads b, τ_t, and $\mathcal{O}_t$ to produce a stopping decision d_t and three candidate clarification states $\mathcal{C}_t$, each recording confirmed elements and unresolved business assumptions. The stopping decision rests with Ψ, not the ledger: it is instructed to favor ASK when an unresolved assumption could change the objective, decision scope, constraints, entities, or time structure, and to allow READY_TO_MODEL when remaining uncertainty concerns only notation, raw data, or solver details.

If d_t is ASK, Ψ pairs each state with one candidate question, yielding $\mathcal{Q}_t$. When $\mathcal{O}_t \neq \emptyset$, every candidate names the identifier of its targeted open entry; otherwise, candidates may address another clarification need without a binding. The runner validates the output schema and required binding, regenerating invalid output up to three times.

An LLM selector then picks $a_t = \text{Select}(\tau_t, \mathcal{C}_t, \mathcal{Q}_t, \mathcal{O}_t)$, prioritizing questions by their structural impact on the formulation: objectives and trade-offs (what is optimized), business constraints and rules (the feasible region), decision scope (which decisions are modeled), and silent-assumption risk (conditions vulnerable to unsupported defaults), while downweighting mathematical details. The entry bound to a_t moves from OPEN to ASKED once selected, before the user answers; this records question selection, not resolution. There is no answer-conditioned resolution step or reopening of ASKED entries, although Stage 1 may add new entries from later transcript evidence. Algorithm 1 summarizes the general procedure.

Algorithm 1 InterOPT clarification framework

Require: Public brief b, interaction protocol π, maximum turns $T_{\max}$
$\tau_1 \leftarrow ()$, $\mathcal{M}_0 \leftarrow \emptyset$
for $t = 1$ to $T_{\max}$ **do**
  $\mathcal{M}_t \leftarrow \Phi(b, \tau_t, \mathcal{M}_{t-1})$; $\mathcal{O}_t \leftarrow \{\ell \in \mathcal{M}_t : \text{status}(\ell) = \textsc{Open}\}$
  $(d_t, \mathcal{C}_t, \mathcal{Q}_t) \leftarrow \Psi(b, \tau_t, \mathcal{O}_t, \pi)$
  if $d_t =$ `READY_TO_MODEL` **then return** τ_t
  $a_t \leftarrow \text{Select}(\tau_t, \mathcal{C}_t, \mathcal{Q}_t, \mathcal{O}_t)$; mark its bound gap (if any) as ASKED
  Execute a_t under protocol π, receive answer u_t, and append (a_t, u_t) to τ_t to obtain τ_{t+1}
end for
return $\tau_{T_{\max}+1}$ (turn-limit termination)

6 Experiments and Results

6.1 Experimental Setup

All main experiments use the 100 cases in OR-Clarify with five independent runs per case unless otherwise stated; Core Exact is evaluated on the 94 cases containing core slots, while All-Slot Exact and Silent/run use all 100 cases. Runs that fail the post-hoc protocol audit remain in the headline metric denominators and are retained for diagnostic analysis. Unless otherwise specified, method comparisons and ablations use DeepSeek V4 Pro as the tested model, $T_{\max} = 20$, agent temperature 0.2, and simulator/judge/selector temperature 0.0. For methods with internal planning roles, all agent-facing decisions—gap discovery, ask/ready decision, candidate generation, and candidate selection—are instantiated with the tested model; the simulated user, protocol monitor, and post-hoc judge are fixed evaluation components (Section 4.2).

6.2 Clarification Protocols

Choice-based setting. **MC** asks questions with three generated options A–C; when none matches, the simulated user must choose the closest option. **MC-D** uses the same agent but adds a fixed option D for free-form correction, and serves as the base protocol for the Choice-based variants. **ReadyGate** augments MC-D with an independent reviewer that accepts or rejects `READY_TO_MODEL`. **InterOPT** operates under MC-D. The simulator's match-status label (exact, acceptable, no-match, or undetermined) is recorded only for diagnostics and is hidden from the agent and judge; the protocol monitor is passive and neither rejects multi-question turns nor triggers regeneration. All Choice methods share the same cases, repetitions, model, simulator, judge, and monitor.

Open-ended setting. All methods share the FreeQA interface, with each action a natural-language question or `READY_TO_MODEL`, and all transcripts are scored by the same judge. **FreeQA** is the base open protocol without an external planner or stopping reviewer; **ReadyGate** adds a readiness reviewer that checks unresolved assumptions when the agent attempts to stop; **InterOPT** is our two-stage formulation-gap-guided framework. GATE (Li et al., 2025), ORPilot (Xie, 2026), and Let's T-P-P (Drossman et al., 2026) are adaptations of prior systems to this pre-formulation clarification interface.

6.3 Performance across Tested Models

Table 1: Baseline model comparison in OR-Clarify.

	Open / FreeQA			Choice / MC-D		
Tested model	All-Slot Exact ↑	Core Exact ↑	Silent/run ↓	All-Slot Exact ↑	Core Exact ↑	Silent/run ↓
DeepSeek V4 Pro	0.426	0.472	0.682	0.474	0.506	0.692
GLM-5.1	0.454	0.491	0.624	0.400	0.434	0.712
GPT-5.5	0.430	0.498	0.632	0.414	0.449	0.660
Opus-4.8	**0.548**	**0.583**	**0.424**	**0.542**	**0.583**	**0.456**

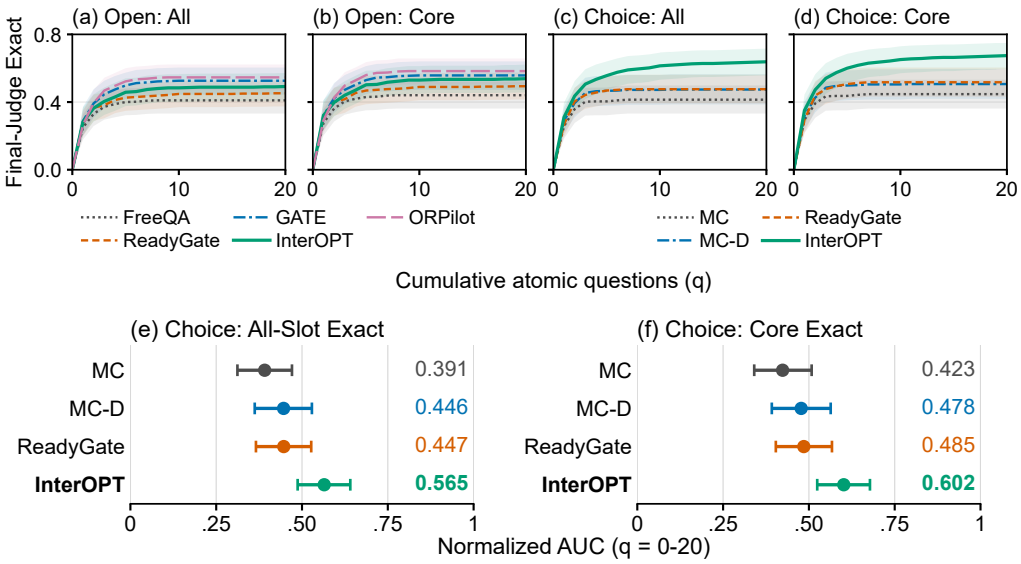

Figure 4: Exact restoration over atomic-question budgets. (a–d) Within-protocol recovery curves. (e–f) Choice AUC, computed by trapezoidal integration over $q = 0$–20 and divided by 20 (higher is better). Choice uses 100 All-Slot and 94 Core-eligible cases, with $K = 5$. Bands and bars show 95% intervals from 10,000 case-bootstrap resamples. Recovery timing is retrospectively localized from final-Judge evidence; curves remain flat after stopping.

Across off-the-shelf LLMs (Table 1), Opus-4.8 is strongest under both protocols, yet no model exceeds 60% Core Exact and all retain substantial silent assumptions, confirming that OR-Clarify remains challenging.

6.4 Method Effectiveness and Recovery over Questions

Table 2: Training-free method comparison in OR-Clarify.

Setting	Method	All-Slot Exact ↑	Core Exact ↑	Silent/run ↓
Open	FreeQA	0.426	0.472	0.682
Open	ReadyGate	0.452	0.494	0.418
Open	ORPilot	**0.546**	**0.583**	**0.268**
Open	Let's T-P-P	0.458	0.489	0.640
Open	GATE	0.528	0.560	0.412
Open	InterOPT	0.492	0.538	0.560
Choice	MC	0.414	0.447	0.774
Choice	MC-D	0.474	0.506	0.692
Choice	ReadyGate	0.476	0.517	0.614
Choice	InterOPT	**0.638**	**0.675**	**0.366**

Under the common 20-turn limit (Table 2), InterOPT achieves the highest Choice recovery and the fewest silent assumptions, while ORPilot leads under Open.

Figure 4(a–d) compares recovery at common atomic-question budgets within each protocol. Under Choice, InterOPT also achieves the highest normalized AUC for All-Slot Exact (0.565) and Core Exact (0.602; panels e–f). These AUCs assess the observed questioning and stopping policies jointly.

6.5 Contributions of the Two Stages

We evaluate two system-level ablations under both protocols: *w/o Stage 1* and *w/o Stage 2*. *w/o Stage 1* disables gap search and ledger population while retaining Stage 2's candidate states, three

candidate questions, and selector; *w/o Stage 2* retains gap search and the ledger but uses a direct gap-to-question policy, with Stage 1 supplying the public question and readiness decision. Both variants use the same cases, repetitions, temperatures, simulator, and judge as full InterOPT.

Table 3: System-level ablation of InterOPT under open/FreeQA and Choice (MC-D) clarification.

Clarification	Variant	All-Slot Exact ↑	Core Exact ↑	Silent/run ↓
Open / FreeQA	None	0.426	0.472	0.682
Open / FreeQA	w/o Stage 1	0.432	0.460	0.668
Open / FreeQA	w/o Stage 2	0.470	0.506	**0.254**
Open / FreeQA	InterOPT	**0.492**	**0.538**	0.560
Choice (MC-D)	None	0.474	0.506	0.692
Choice (MC-D)	w/o Stage 1	0.520	0.564	0.612
Choice (MC-D)	w/o Stage 2	0.572	0.615	0.462
Choice (MC-D)	InterOPT	**0.638**	**0.675**	**0.366**

Removing Stage 1 yields a larger Core Exact drop than removing Stage 2 under each protocol (Open: 7.8 vs. 3.2 percentage points; Choice: 11.1 vs. 6.0; Table 3). Both stages contribute to recovery in these system-level ablations.

6.6 Protocol-Specific Diagnostics

Choice diagnostics. In MC-D, all 132 no-match events among 1,129 audited choice events triggered option D and a public free-form correction; this observation does not isolate D's causal effect. Across 890 judged slots per method ($K = 5$), InterOPT receives 617 `yes`, 20 `partial`, and 253 `no` labels, versus 461, 16, and 413 for MC-D. Partial matches count as failures, so the exact-recovery advantage reflects more fully recovered slots, not partial credit.

Open diagnostics. Open-protocol InterOPT declares readiness in 98.8% of runs, yet the stopping audit labels 40.6% (203 of 500) as premature, and 0.560 silent assumptions per run remain. One possible contributor to premature stopping is the protocol itself: free-form answers can carry context beyond the immediate question, making a specification appear complete before every formulation-critical condition is settled. However, premature stopping alone does not explain how InterOPT compares with ORPilot and GATE.

An exploratory, model-assisted analysis of one run per case for each method (300 dialogues) identifies a possible pathway through which ORPilot may benefit from the Open setting. In some dialogues, users supplied facts beyond the immediate question, which the agent subsequently incorporated into confirmation requests. This distinction matters under our evaluation rules: volunteered facts receive no exact credit until a later question or explicit assumption check establishes the required evidence. ORPilot can therefore obtain credit by following up on user-supplied information, without independently identifying each missing condition beforehand. This broader confirmation style appears in the turn-level analysis: ORPilot asks about or confirms two or more hidden slots in 12.3% of question turns, compared with 1.8% for InterOPT and GATE.

GATE's margin over InterOPT is small (10 of 470 core-eligible runs) and inconsistent across repetitions: in the first-run subset, InterOPT's All-Slot Exact is higher (54% vs. 48%). We therefore do not attribute this margin to a specific mechanism. The same richness of free-form answers may explain why direct questioning remains competitive in Open, leaving less room for explicit gap tracking to add value.

7 Conclusion

We introduced OR-Clarify for evaluating pre-formulation requirement recovery and InterOPT, a two-stage framework for gap discovery, question selection, and stopping. Under Choice, InterOPT achieves the highest exact recovery among evaluated methods and fewer silent assumptions than MC-D and ReadyGate. Ablations support both stages, while Open performance and stopping calibration remain areas for improvement.

Ethics Statement

This work adheres to the ICLR Code of Ethics. Our study evaluates the clarification of operations-research problem descriptions using benchmark cases and simulated users; it does not involve human participants, personally identifiable data, or deployment with real stakeholders.

Reproducibility Statement

Sections 3–6 specify the task, benchmark construction, evaluation metrics, method, interaction protocols, model roles, and experimental settings. The anonymized supplementary package provides sanitized pipeline code, prompt and configuration templates, a case schema, aggregate results, and illustrative interaction excerpts to support inspection of the evaluation procedure.

AI Use Statement

Generative AI tools were used to assist with research methodology and experimental design, method implementation and code development, literature exploration, interpretation of experimental results, and manuscript preparation. All AI-assisted work was reviewed by the authors. AI-assisted code was tested, relevant references were verified against their original sources, and AI-assisted analyses and manuscript text were reviewed and revised by the authors. The authors take full responsibility for the final content of this work.

References

Ali AhmadiTeshnizi, Wenzhi Gao, and Madeleine Udell. Optimus: Scalable optimization modeling with (mi) lp solvers and large language models. *arXiv preprint arXiv:2402.10172*, 2024.

Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W Bruce Croft. Asking clarifying questions in open-domain information-seeking conversations. In *Proceedings of the 42nd international acm sigir conference on research and development in information retrieval*, pp. 475–484, 2019.

Mohammad Aliannejadi, Julia Kiseleva, Aleksandr Chuklin, Jeff Dalton, and Mikhail Burtsev. Convai3: Generating clarifying questions for open-domain dialogue systems (clariq). *arXiv preprint arXiv:2009.11352*, 2020.

Chinmaya Andukuri, Jan-Philipp Fränken, Tobias Gerstenberg, and Noah D Goodman. Star-gate: Teaching language models to ask clarifying questions. *arXiv preprint arXiv:2403.19154*, 2024.

Yulin Dou and Jiangming Liu. To-gate: Clarifying questions and summarizing responses with trajectory optimization for eliciting human preference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 30548–30556, 2026.

Joshua Drossman, Alexandre Jacquillat, and Sébastien Martin. Let's have a conversation: Designing and evaluating llm agents for interactive optimization. *arXiv preprint arXiv:2604.02666*, 2026.

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

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Polina Kirichenko, Mark Ibrahim, Kamalika Chaudhuri, and Samuel J Bell. Abstentionbench: Reasoning llms fail on unanswerable questions. *Advances in Neural Information Processing Systems*, 38, 2026.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Clam: Selective clarification for ambiguous questions with generative language models. *arXiv preprint arXiv:2212.07769*, 2022.

Dongryeol Lee, Segwang Kim, Minwoo Lee, Hwanhee Lee, Joonsuk Park, Sang-Woo Lee, and Kyomin Jung. Asking clarification questions to handle ambiguity in open-domain qa. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 11526–11544, 2023.

Belinda Li, Alex Tamkin, Noah Goodman, and Jacob Andreas. Eliciting human preferences with language models. In *International Conference on Learning Representations*, volume 2025, pp. 80984–81013, 2025.

Kuo Liang, Yuhang Lu, Jianming Mao, Shuyi Sun, Chunwei Yang, Congcong Zeng, Xiao Jin, Hanzhang Qin, Ruihao Zhu, and Chung-Piaw Teo. 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, Tianyi Xu, Ruoqing Jiang, Zizhuo Wang, and Dongdong Ge. From soliloquy to agora: Memory-enhanced LLM agents with decentralized debate for optimization modeling, 2026. URL https://arxiv.org/abs/2604.25847.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. Ambigqa: Answering ambiguous open-domain questions. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 5783–5797, 2020.

Thomas O Nelson. Metamemory: A theoretical framework and new findings. In *Psychology of learning and motivation*, volume 26, pp. 125–173. Elsevier, 1990.

Rindranirina Ramamonjison, Timothy Yu, Raymond Li, Haley Li, Giuseppe Carenini, Bissan Ghaddar, Shiqi He, Mahdi Mostajabdaveh, Amin Banitalebi-Dehkordi, Zirui Zhou, et al. Nl4opt competition: Formulating optimization problems based on their natural language descriptions. In *NeurIPS 2022 competition track*, pp. 189–203. PMLR, 2023.

Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. Interpretation of natural language rules in conversational machine reading. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2087–2097, 2018.

Tabinda Sarwar, Farhad Moghimifar, Cong Duy Vu Hoang, Xiaoxiao Ma, Shawn Chang Xu, Fahimeh Saleh, Poorya Zaremoodi, Avirup Sil, and Katrin Kirchhoff. Clarity: A framework and benchmark for conversational language ambiguity and unanswerability in interactive nl2sql systems. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL 2026)*, pp. 1218–1250, 2026.

Xiang Shu, Hong Qian, Xingyu Lu, JUN ZHOU, Aimin Zhou, Yang Yu, et al. Llmopt: Learning to define and solve general optimization problems from scratch. In *International Conference on Learning Representations*, volume 2025, pp. 101580–101606, 2025.

Manan Suri, Puneet Mathur, Nedim Lipka, Franck Dernoncourt, Ryan A Rossi, and Dinesh Manocha. Structured uncertainty guided clarification for llm agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 40811–40838, 2026.

Guangrui Xie. Orpilot: A production-oriented agentic llm-for-or tool for optimization modeling, 2026. URL https://arxiv.org/abs/2605.02728.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=roNSXZpUDN.

Michael Zhang, W Bradley Knox, and Eunsol Choi. Modeling future conversation turns to teach llms to ask clarifying questions. In *International Conference on Learning Representations*, volume 2025, pp. 60722–60742, 2025.

Michael JQ Zhang and Eunsol Choi. Clarify when necessary: Resolving ambiguity through interaction with lms. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 5541–5558, 2025.

Jiale Zhao, Ke Fang, and Lu Cheng. When and what to ask: Askbench and rubric-guided rlvr for llm clarification. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17120–17140, 2026.

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.