ALGOWORLDS: BENCHMARKING TOOL USE FOR GLOBAL OPTIMIZATION IN ALGORITHMIC WORLDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-use benchmarks generally evaluate whether an agent completes a workflow using appropriate tools and valid arguments. However, feasibility alone is insufficient in real-world decision settings such as route planning and fleet dispatch. Individual choices interact through shared constraints and costs, so a feasible solution may still be substantially suboptimal. This raises a harder question: can an agent turn information gathered through tools into a globally optimal decision? In this paper, we introduce ALGOWORLDS, a benchmark that transforms formally specified combinatorial optimization problems into partially observed decision environments with global optima. Each environment contains a hidden optimization instance that the agent observes only through sequential calls to task-specific information tools. The agent then commits to one structured decision, which an independent checker evaluates for feasibility, objective value, and optimality. ALGOWORLDS contains 240 such environments, covering ten combinatorial optimization families and four workload levels. To make these environments both controllable and auditable, family-specific deterministic programs generate hidden instances, while exact algorithms certify their global optima and determine their workload levels. Each instance is exposed through two structurally different multi-tool interfaces, which preserve the same decision problem under different information presentation. We evaluate seven leading LLMs, including Claude Opus 4.8 and GPT-5.6 Sol, on ALGOWORLDS. Results show that achieving global optimality remains highly challenging: although leading models achieve feasibility in most cases, the best-performing model achieves exact optimality in only 38.61% of cases. Further analysis shows that even when the LLM agents collect sufficient information to reconstruct the hidden instance, most failures end in feasible but suboptimal decisions. The challenge therefore extends beyond information acquisition to information integration, global constraint reasoning, and decision verification.

1 INTRODUCTION

Tool-use benchmarks evaluate whether an agent selects appropriate tools and completes a prescribed workflow (Li et al., 2023; Yan et al., 2024; Lu et al., 2025b; Yao et al., 2025). These benchmarks generally assess correct tool use and task completion, but neglect whether the solution is globally optimal. In many real-world decision problems, such as route planning, fleet dispatch, and resource allocation, a feasible solution may still be far from optimal (Xie et al., 2024a; Zhang et al., 2026b; Qin et al., 2025). The difficulty is that individual decisions interact through shared constraints and aggregate costs. For example, in the fleet dispatch problem, a dispatcher must assign every job to a vehicle using job requirements, vehicle eligibility, capacities, route costs, and global activation costs distributed across separate information sources. Assigning each job to its cheapest eligible route can violate capacity constraints, while a capacity-feasible assignment can remain substantially suboptimal if it activates an unnecessarily costly set of vehicles. Solving such problems therefore requires multi-source information integration and joint reasoning rather than considering each tool use independently. We therefore ask: *Can an LLM agent make a globally optimal decision?*

We introduce ALGOWORLDS, a benchmark for evaluating whether LLM agents can turn information acquired through tools into globally optimal decisions. An *algorithmic world* combines a query, a hidden problem instance that determines feasible decisions and objective values, and task-specific tools that expose parts of that instance. The agent receives the query and tools, selects sequential

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.

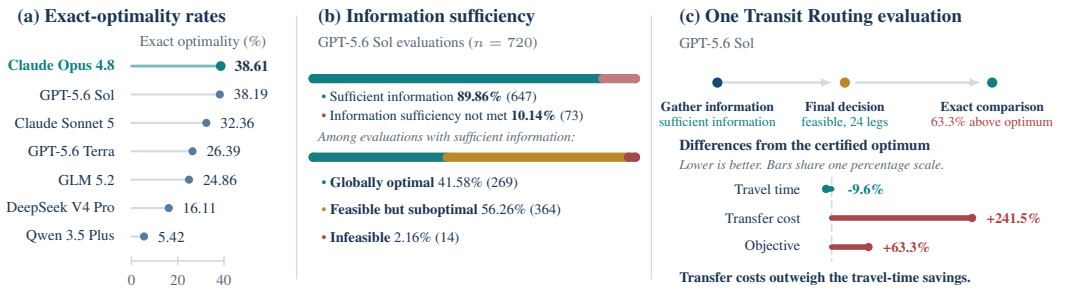


Figure 1: **Sufficient information does not guarantee global optimality.** **(a)** Exact-optimality rates for the seven evaluated LLMs. **(b)** In the evaluation of GPT-5.6 Sol, trajectories are initially categorized based on whether they gather sufficient information. Those that meet this criterion are further classified according to their final-decision results. **(c)** An example evaluation of GPT-5.6 Sol on Transit Routing. The trajectory gathers sufficient information and ends with a feasible, albeit non-optimal, route.

costed calls under a fixed access budget based on prior observations, and finally commits to one decision.

Building such worlds reliably requires hidden instances that can be scaled systematically and whose optima can be verified exactly, together with faithful informational tools. Following this recipe, we instantiate ALGOWORLDS from ten formally specified combinatorial optimization families, *e.g.*, routing, exact cover, facility selection and constrained sequencing. Within each family, a human-written deterministic generator creates hidden instances based on specified size and structure configurations, enabling us to systematically vary the problem's scale and the underlying reasoning structure of its decisions. A tailored exact algorithm then solves each instance to global optimality. We also count the primitive operations the algorithm actually executes to identify four workload levels. We next build the informational tools that expose different parts of an instance, such as the travel distance between two cities. Specifically, we present each hidden instance through two structurally different multi-tool interfaces that expose the same underlying decision information. This paired design lets us assess sensitivity to tool presentation while holding the same optimization problem. To further ensure data quality, we independently confirm each instance's optimal decision, objective value, and uniqueness, verify that the paired interfaces encode the same hidden instance, and confirm that sufficient information can be acquired within the total access budget.

To evaluate LLM agents in ALGOWORLDS, we report three final-decision metrics and two trajectory-oriented diagnostics. (a) *Final-decision metrics.* Exact optimality, our primary metric, records whether a feasible decision attains the global optimum. Feasibility records whether the decision satisfies the task constraints. Reference utility assigns a feasible decision a continuous score ranging from 0 to 1, with the global optimum at 1 and a predefined suboptimal reference decision at 0. (b) *Trajectory-oriented diagnostics.* Information sufficiency is satisfied when the trajectory collects enough tool-provided information to reconstruct the hidden instance and determine its global optimum. Discovery coverage measures how close the information gathered along the trajectory comes to meeting that threshold.

Using ALGOWORLDS, we evaluate seven leading LLMs: Claude Opus 4.8 (Anthropic, 2026a), Claude Sonnet 5 (Anthropic, 2026b), GPT-5.6 Sol, GPT-5.6 Terra (OpenAI, 2026), DeepSeek V4 Pro (DeepSeek-AI, 2026), Qwen 3.5 Plus (Qwen Team, 2026), and GLM 5.2 (Z.ai, 2026). As Figure 1(a) shows, Claude Opus 4.8 achieves the highest exact-optimality rate, at only 38.61%, followed closely by GPT-5.6 Sol at 38.19%. Notably, 89.86% of GPT-5.6 Sol's trajectories contain sufficient information, yet 58.42% of them still end without a globally optimal decision. Most of these failures end in feasible but suboptimal decisions, as Figure 1(b) shows. The Transit Routing example in Figure 1(c) illustrates this pattern. In this example, the trajectory gathers sufficient information, yet GPT-5.6 Sol constructs a feasible route that has lower travel time than the globally optimal route while incurring substantially higher transfer costs. As a result, its overall objective value is 63.3% higher. Together, these results expose a substantial gap between information availability and global optimization: current LLM agents often fail to reach the optimum even when they collect sufficient information.

Our main contributions are threefold. (1) We introduce ALGOWORLDS, a benchmark for evaluating whether LLM agents can turn information acquired through tools into globally optimal decisions. Its 240 algorithmic worlds span ten optimization families and four workload levels. **(2)** To build the benchmark, we provide a verified optimum and faithful information tools for every hidden instance. Paired tool interfaces further measure the sensitivity of information presentation while holding the same underlying decision problem. **(3)** Across seven leading LLMs, we identify a clear capability profile: agents are often effective at gathering information and producing feasible decisions, but remain much less reliable at finding the global optimum. This profile is relatively stable across paired interfaces but varies sharply across different optimization families in ways that do not follow classical asymptotic complexity alone.

2 RELATED WORK

Prior tool-use and interactive-agent benchmarks test API selection, valid arguments, stateful interaction, and workflow completion (Li et al., 2023; Yan et al., 2024; Lu et al., 2025b; Yao et al., 2025). Partially observed environments test information acquisition and synthesis across executable states and distributed sources (Zhou et al., 2024; Mialon et al., 2023). Planning benchmarks add feasibility and utility under constraints (Xie et al., 2024a; Zhang et al., 2026b; Qin et al., 2025). These settings emphasize task completion or valid plans rather than exact global optimality. Work on algorithmic and optimization reasoning typically assumes that the problem instance is available before optimization, or treats the generated solver as the target (Markeeva et al., 2024; Huang et al., 2024b; Sun et al., 2025). ALGOWORLDS instead asks an agent to acquire information about a hidden instance through task-specific tools and provide one structured decision scored for feasibility, objective value, and exact global optimality. Appendix A provides the full comparison.

3 ALGOWORLDS

3.1 FORMAL DEFINITION

Algorithmic Worlds. An *algorithmic world* is a partially observed decision environment containing a textual question (q), a hidden optimization instance ($\mathcal{I}$), and task-specific tools ($\mathcal{T}$). To simulate the limitations of information access in real-world scenarios, each tool $t_i \in \mathcal{T}$ incurs a specific cost c_i, and there is a predefined budget (b) to specify the upper limit of the total cost. Based on the above, we define an algorithmic world as:

$$w = \{q, \mathcal{I}, \mathcal{T}, f_c, b\} \tag{1}$$

where f_c is the cost function to define the cost of each tool, *i.e.*, $c_i = f_c(t_i)$. The textual question q describes the decision to be made, the constraints it must satisfy, and the objective to optimize. The instance $\mathcal{I}$ contains the case-specific data that determine the feasible decisions and their objective values. The task-specific tools $\mathcal{T}$ include various information tools $\mathcal{T}_{\text{info}}$ and one final-decision tool t_{dec} for LLM agents to obtain information about $\mathcal{I}$ and commit a final decision, respectively. The final-decision tool t_{dec} is zero-cost, and can be invoked only once at the end of one trajectory. Taking the fleet dispatch problem as an example, $\mathcal{I}$ records job loads, vehicle capacities, eligible vehicle–job pairs, route prices, and fixed activation costs. The question q asks the agent to assign every job while minimizing route and activation costs subject to eligibility and capacity constraints. Separate job, route, policy, and vehicle-contract tools in $\mathcal{T}_{\text{info}}$ reveal the information needed to make the optimal assignment. The final-decision tool t_{dec} accepts one complete job-to-vehicle mapping.

Agent trajectories. Let π denote an LLM agent. When addressing the task, the agent receives $w \setminus \mathcal{I} = \{q, \mathcal{T}, f_c, b\}$, and interacts with task-specific tools $\mathcal{T}$ to solve the task:

$$\pi(w \setminus \mathcal{I}) \to t_{s_1} \to o_{s_1} \to t_{s_2} \to o_{s_2} \to \cdots \to t_{s_k} \to o_{s_k} \to t_{\text{dec}}(y), \qquad \sum_{i=1}^{k} f_c(t_{s_i}) \le b. \tag{2}$$

where t_{s_i} and o_{s_i} denote the tool call(s) in the i-th step and the corresponding returned information (observation). y is the agent's final decision committed via t_{dec}. At each step, the agent interacts with tools based on the task and the trajectory history. Information tools can be repeatedly invoked as long as their cumulative cost does not exceed b. If a call is rejected due to insufficient remaining

budget, it may return an error response; however, it will not be executed, nor will it incur any costs. Appendix B presents an example drawn from a released Fleet Dispatch world, illustrating its task, tool interactions, and final decision.

3.2 BENCHMARK CONSTRUCTION

ALGOWORLDS is constructed from ten formally specified combinatorial optimization families. Each family specifies the structure of $\mathcal{I}$, the constraints that make a decision y feasible, and the objective used to evaluate y. Although the families share the interaction protocol in Section 3.1, they differ in how individual choices interact under global constraints and in the exact algorithms used to find the global optimum. Table 1 lists all ten families and summarizes their decision, instance data, exact solution method, and number of information tools $\mathcal{T}_{\text{info}}$. Formal problem definitions and complexity analyses are provided in Appendix C.

Table 1: The ten optimization families in ALGOWORLDS.

Task family	Decision y	Instance data in $\mathcal{I}$	Exact solution method	$\lvert\mathcal{T}_{\text{info}}\rvert$
Transit Routing	One route leg per stage	Stage-specific legs, travel times, line labels, directed transfer costs, and residues	Residue and last-line dynamic program	6
Basket Assembly	A ticket set forming an exact cover	Requirements, ticket–item incidence, and prices	First-uncovered-item bitmask dynamic program	5
Station Siting	One site per zone	Shared capacity, site costs, and bounded-span pairwise demand rewards	Resource-aware band-frontier dynamic program	4
Authorization Planning	A package sequence	Authorization relations, policy adjustments, handoffs, and clearance residue	Product-state dynamic program	6
Series Portfolio	One lot per title	Lot values, scopes, and overlapping signed factors	Bounded-span factor dynamic program	6
Machine Layout	A machine-to-slot permutation	Machine flows, a unit-spaced slot order, and placement adjustments	Cut-identity subset dynamic program	8
Sequential Matching	An ordered perfect matching	Compatibility, pair costs, and directed handoffs	Used-set and last-match dynamic program	6
Fleet Dispatch	A job-to-vehicle assignment	Job loads, eligible vehicle–job pairs, capacities, route costs, and activation costs	Mixed-radix capacity dynamic program	6
Evidence-Joined Routing	A layered source-to-target path	Layered graph with source and target, quantities, exceptions, tariffs, and residues	Expanded-state shortest path	6
Migration Portfolio	One package per component group and an activated domain set	Package costs, support domains, activation costs, and signed interactions	Activated-domain mask enumeration	8

For each family, benchmark construction proceeds in three stages: First, a deterministic generator produces a hidden instance ($\mathcal{I}$) based on a specified size and structure configuration. Second, a tailored exact algorithm solves the instance to find its optimal decision and determines its workload level. Finally, we construct a textual question (q) and expose the same instance through Direct and Mediated tool interfaces ($\mathcal{T}_{\text{d}}$ and $\mathcal{T}_{\text{m}}$) with the same call costs (f_c) and access budget (b), yielding a pair of algorithmic worlds, *i.e.*, $\{q, \mathcal{I}, \mathcal{T}_{\text{d}}, f_c, b\}$ and $\{q, \mathcal{I}, \mathcal{T}_{\text{m}}, f_c, b\}$.

Stage 1: deterministic instance generation. Each family uses a human-written deterministic generator to produce $\mathcal{I}$ based on a specified size and structure configuration. Among them, the size configuration dictates the scale of the problem, while the structure configuration governs how the decision y interacts within the global constraints and objectives. With a specific configuration, the generator produces a consistent instance, ensuring its reproducibility.

Stage 2: optimal decision and workload definition. For each family, a tailored exact algorithm is employed to solve every generated instance to global optimality, and the number of its executed primitive operations is used to define four workload levels. In this way, we ensure that instances (from different families) at the same workload level require a comparable amount of computational effort. Section 4.2 formalizes the primitive operations and workload levels.

Stage 3: tool-mediated world construction. For each family, we predefine a *reusable world-construction template* that specifies the textual question q, the mappings from instance fields to

tools $\mathcal{T}$, the tool-call costs f_c and access budget b. Instantiating the template with a hidden instance $\mathcal{I}$ produces two algorithmic worlds, one using the Direct interface $\mathcal{T}_{\mathrm{d}}$ and the other using the Mediated interface $\mathcal{T}_{\mathrm{m}}$. The paired worlds share the same $\mathcal{I}$, q, f_c and b; their difference lies in the relational structure used to present the instance information. For each of the ten families and four workload levels, we generate three hidden instances and present each through both interfaces, yielding $10 \times 4 \times 3 = 120$ hidden instances and $120 \times 2 = 240$ algorithmic worlds.

3.3 Data Quality

To ensure data quality, we verify that each hidden instance possesses a feasible decision and a unique global optimum. Additionally, we confirm that sufficient information can be obtained using tools within the access budget.

Verified optima. For each family, the construction algorithm is exact under the formal problem definition, but the optimal decision and objective value for a generated instance $\mathcal{I}$ still depend on the correctness of its implementation. We therefore solve $\mathcal{I}$ a second time with an independently implemented exact solver based on a different state direction or problem presentation. An instance is retained only if both implementations concur on its optimal decision, objective value, and identify the same unique optimum.

Information access. For every algorithmic world w, we validate two alternative ways of gathering information ($\mathcal{T}_{\mathrm{d}}$ and $\mathcal{T}_{\mathrm{m}}$). Each way specifies exactly which information tools are called. When these tools are invoked, the covered information must reconstruct $\mathcal{I}$ and recover its optimum, while their total cost remains within b. We also evaluate the cost of exhaustively calling all tools to disclose instance information. The cost exceeds b in all w, confirming that enough information can be acquired within the budget but exhaustive tool use cannot.

For more details of the validation, please refer to Appendix F.

3.4 Evaluation Metrics

Final-decision metrics. We use three metrics to evaluate the quality of the final decision y. If no final decision is submitted via t_{dec}, all three metrics are set to zero.

(1) *Feasibility.* For a world w, let $F_w(y) \in \{0, 1\}$ indicate whether the decision y is feasible. Specifically, $F_w(y) = 1$ if y satisfies all task constraints, and $F_w(y) = 0$ otherwise.

(2) *Exact optimality.* For a feasible decision y, let $v_w(y)$ denote its objective value and $v_w^\star$ the verified optimal value for w. We define its optimality gap as $\Delta_w(y) = |v_w(y) - v_w^\star|$. Exact optimality, our primary metric, is then defined as

$$E_w(y) = \begin{cases} 1, & F_w(y) = 1 \text{ and } \Delta_w(y) = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

(3) *Reference Utility.* To assess the nuanced degree of optimality, we also define a reference optimality gap as Δ_w^{ref}, which is calculated based on a well-defined suboptimal reference decision. In this way, the reference utility of y is defined as:

$$U_w(y) = \begin{cases} \max\left\{0, 1 - \Delta_w(y)/\Delta_w^{\mathrm{ref}}\right\}, & F_w(y) = 1, \\ 0, & F_w(y) = 0. \end{cases} \tag{4}$$

$U_w(y)$ equals 1 at the optimum, decreases to zero at the reference decision, and remains zero for feasible decisions no better than the reference or infeasible decisions.

Trajectory diagnostics. We employ two diagnostic measures to evaluate the decision information gathered from $\mathcal{T}_{\mathrm{info}}$ along a trajectory τ. As described in Section 3.3, for each world w, we define the necessary information required to reconstruct $\mathcal{I}$. Since this information can be acquired in various ways, a world may possess multiple sufficient information sets. (1) *Information sufficiency* records whether the gathered information in τ include a complete sufficient set, while (2) *discovery coverage* measures the largest proportion of any sufficient set present within the gathered information. Formally,

let $\mathcal{A}_w(\tau)$ denote the set of instance information contained in τ, and let $\mathcal{S}(w)$ denote the sufficient information sets described above. Information sufficiency and discovery coverage can be defined as:

$$C_w(\tau) = \mathbb{1}[D_w(\tau) = 1], \qquad D_w(\tau) = \max_{S \in \mathcal{S}(w)} \frac{|\mathcal{A}_w(\tau) \cap S|}{|S|}. \tag{5}$$

In this way, $C_w(\tau) = 1$ and $E_w(y) = 0$ indicate that the agent gathers enough information, but fail into a non-optimal decision. Because trajectory diagnostics are separate from final-decision quality, they can be utilized to assess whether and how LLM agents turn available information into globally optimal decisions. Further details on the sufficient information sets and diagnostic computation are provided in Appendix G.

4 Algorithmic Foundations

This section explains two algorithmic motivations that underlie the benchmark construction. First, within each task family, individual decisions interact through shared constraints or objective terms, and a family-specific exact algorithm solves the resulting combinatorial optimization problem to global optimality. Second, the primitive operations executed by these algorithms on each generated instance provide a reproducible construction scale for assigning instances to four workload levels.

4.1 Global Coupling and Exact Algorithms

In every family, actions are coupled through shared constraints or objectives. Routing and sequencing families propagate residues or transition costs across successive actions, whereas other families couple actions through shared capacities, activation costs, or exact-cover conditions. As a result, optimizing each action independently does not lead to a global optimum. The family-specific exact algorithms therefore keep, at every step, the information that later actions depend on. For example, in Fleet Dispatch, the dynamic program retains the vehicle-load vector while processing jobs: this vector records both how much room each vehicle still has and which activation costs have already been incurred, neither of which can be recovered from the total remaining capacity alone.

We run these exact algorithms only when constructing the benchmark, so that every instance carries a certified optimum and agent performance can be reported as a gap to it. These procedures are never shown to the evaluated LLM agents, which must uncover the coupling structure of each family on their own and turn information acquired through tools into globally optimal decisions.

4.2 Algorithm-Grounded Workload Levels

Size configuration in deterministic instance generation does not provide a meaningful scale across families because their exact algorithms operate over different decision spaces. For each generated instance, the number of five primitive operations (*i.e.*, visited states, examined transitions, constraint checks, value updates, and reconstructed decision components) is counted:

$$\mathrm{Work}(\mathcal{I}) = N_{\mathrm{state}} + N_{\mathrm{transition}} + N_{\mathrm{check}} + N_{\mathrm{update}} + N_{\mathrm{reconstruct}}. \tag{6}$$

Every instance uses the same primary operations and counting rules across different families. We assign family-specific configuration settings to four workload levels: Levels 1-4 use the inclusive bands $[2^{15}, 2^{17}]$, $[2^{19}, 2^{21}]$, $[2^{22}, 2^{24}]$, and $[2^{24}, 2^{26}]$, respectively. This procedure aligns instances $\mathcal{I}$ by the number of executed primitive operations of their exact algorithms without requiring their raw size configuration or decision-space sizes. The gaps between consecutive bands are deliberate: they keep adjacent levels at least a factor of two apart in $\mathrm{Work}(\mathcal{I})$, so the levels stay clearly separable. Levels 3 and 4 share the endpoint 2^{24} instead, and are separated by requiring $\mathrm{Work}(\mathcal{I})$ to increase strictly across levels within every family. For more details of the operation-counting rules and the workloads implemented at each level, please refer to Appendix H.

5 Experiments and Analysis

5.1 Experimental Setup

We evaluate seven leading LLMs: GPT-5.6 Terra and GPT-5.6 Sol (OpenAI, 2026), Qwen 3.5 Plus (Qwen Team, 2026), DeepSeek V4 Pro (DeepSeek-AI, 2026), Claude Opus 4.8 and Claude Sonnet

5 (Anthropic, 2026a;b), and GLM 5.2 (Z.ai, 2026). We enable the reasoning mode for all LLM agents and limit the number of information-tool calls to a maximum of 255. For each agent, we perform three trials and report the mean and sample standard deviation of the results. Detailed hyperparameters used during model inference, such as temperature and reasoning effort, along with additional evaluation details, are provided in Appendix I.

5.2 Main Results

Optimality vs. Feasibility. Table 2 presents the overall performance of all LLM agents. Notably, none of the agents achieve an exact-optimality score exceeding 40%. The best LLM, Claude Opus 4.8, achieves 38.61%, followed closely by GPT-5.6 Sol at 38.19%. In terms of feasibility, the scores are substantially higher: Claude Opus 4.8 and GPT-5.6 Sol achieve 96.25% and 97.50%, respectively. Consequently, in over half of the cases, these two leading LLMs produce decisions that are feasible but not optimal, demonstrating that the most advanced models are significantly more adept at meeting task constraints than at identifying the global optimum. This result also highlights the challenges within our AlgoWorlds, suggesting that making a globally optimal decision remains an open question for LLM agents.

Table 2: Performance on all 240 algorithmic worlds in AlgoWorlds. Entries are the mean ± sample standard deviation over three trials. **Bold** marks the best performance.

	Final-decision metrics			**Trajectory diagnostics**	
Model	Exact optimality	Feasibility	Reference utility	Information sufficiency	Discovery coverage
Qwen 3.5 Plus	5.42 ± 0.42	48.89 ± 1.73	15.82 ± 0.87	37.92 ± 1.25	81.55 ± 0.42
DeepSeek V4 Pro	16.11 ± 1.05	83.89 ± 1.34	28.49 ± 0.91	79.58 ± 1.67	97.74 ± 0.11
GLM 5.2	24.86 ± 0.64	64.58 ± 3.15	35.48 ± 2.67	83.33 ± 3.82	93.91 ± 1.49
GPT-5.6 Terra	26.39 ± 1.34	87.78 ± 0.87	54.10 ± 1.95	79.72 ± 3.13	96.65 ± 0.25
GPT-5.6 Sol	38.19 ± 1.34	**97.50 ± 0.00**	**74.98 ± 1.77**	**89.86 ± 0.64**	**99.10 ± 0.13**
Claude Sonnet 5	32.36 ± 3.85	92.08 ± 0.83	67.20 ± 9.53	86.39 ± 7.36	98.59 ± 1.08
Claude Opus 4.8	**38.61 ± 4.65**	96.25 ± 2.89	73.14 ± 11.81	81.53 ± 8.74	97.33 ± 1.45

Informative but Suboptimal. The trajectory diagnostics further show that the gap between optimality and feasibility *cannot* be explained by information acquisition alone. The information-sufficiency score exceeds the exact-optimality score for every model, ranging from 37.92% to 89.86%. Discovery coverage is higher still: five of the seven models exceed 96%, with GPT-5.6 Sol reaching 99.10%. The results indicate that agents generally gather enough information to determine the optimum but still fail to turn that information into a globally optimal decision.

Table 3: Information sufficiency and the final results in informative trajectories.

	Information sufficiency	**Three final results in informative trajectories**		
Model		Globally optimal	Feasible but suboptimal	Other failure
Qwen 3.5 Plus	37.9	3.3	49.9	46.8
DeepSeek V4 Pro	79.6	17.8	69.4	12.8
GLM 5.2	83.3	26.0	40.5	33.5
GPT-5.6 Terra	79.7	29.2	59.8	11.0
GPT-5.6 Sol	89.9	41.6	56.3	2.1
Claude Sonnet 5	86.4	33.9	58.0	8.1
Claude Opus 4.8	81.5	36.3	59.5	4.2

To further investigate this gap, we categorize trajectories with sufficient information (denoted as *informative trajectories*) into three distinct final results: globally optimal, feasible but suboptimal, and other failures (*e.g.*, infeasible decisions or absent final decisions). As shown in Table 3, only 29.2% of GPT-5.6 Terra's informative trajectories achieve a globally optimal outcome, while the remaining 70.8% fall short, with 59.8% resulting in feasible but suboptimal decisions. This pattern is consistent across all seven models, with "feasible but suboptimal" representing the largest proportion

among the three results. Even for GPT-5.6 Sol, 56.3% of informative trajectories result in feasible but suboptimal decisions, while only 41.6% reach the optimum. Information sufficiency measures the gathered information in trajectories, rather than the agent's ability to recognize or utilize it correctly. This finding thus underscores the gap between the ability to gather information and the capacity to reason towards an optimal decision in ALGOWORLDS. Detailed conditional outcome counts and structured trajectory examples appear in Table 20 and Appendix J.7.

5.3 PERFORMANCE ACROSS DIFFERENT PERSPECTIVES

We analyze the model performance specifically from the three perspectives of task families, workload levels, and tool interfaces:

Task families. Table 4 illustrates the significant variation in exact-optimality scores across different task families. For instance, the scores for Station Siting, Fleet Dispatch, and Authorization Planning range from 0.0% to 98.6%, 6.9% to 94.4%, and 0.0% to 72.2% across various LLMs, respectively. Notably, Basket Assembly achieves the highest scores, with four LLMs reaching 100% optimality. In stark contrast, no model surpasses 10% in tasks such as Transit Routing, Series Portfolio, Machine Layout, Sequential Matching, and Evidence-Joined Routing. The disparity between Transit Routing and Basket Assembly is particularly revealing. Although the exact algorithm for Transit Routing has a worst-case bound that is polynomial in its listed parameters, the bound for Basket Assembly includes an exponential factor, as detailed in Appendix C. Despite this, LLMs perform significantly better on Basket Assembly. This suggests that the asymptotic complexity of the benchmark's exact algorithms alone does not account for the observed differences in performance across task families.

Table 4: Performance across task families (in terms of exact optimality). The pale teal color represents the magnitude of the scores, and the **bold** indicates the row-best value.

Task family	Terra	Sol	Qwen	DeepSeek	Opus	Sonnet	GLM
Transit Routing	0.0	**1.4**	0.0	0.0	0.0	0.0	0.0
Basket Assembly	**100.0**	**100.0**	47.2	88.9	**100.0**	**100.0**	97.2
Station Siting	83.3	**98.6**	0.0	8.3	97.2	90.3	58.3
Authorization Planning	18.1	**72.2**	0.0	4.2	62.5	59.7	19.4
Series Portfolio	0.0	0.0	0.0	0.0	**9.7**	1.4	0.0
Machine Layout	1.4	1.4	0.0	0.0	**4.2**	0.0	0.0
Sequential Matching	1.4	**6.9**	0.0	0.0	0.0	2.8	1.4
Fleet Dispatch	59.7	**94.4**	6.9	59.7	87.5	50.0	72.2
Evidence-Joined Routing	0.0	0.0	0.0	0.0	2.8	**4.2**	0.0
Migration Portfolio	0.0	6.9	0.0	0.0	**22.2**	15.3	0.0

Workload levels. Table 5 shows declining performance with workload. From L1 to L4, reference utility falls by about 8˜29% for every model and exact optimality by 1.1˜7.7%. Because workload is assigned from exact-algorithm operations independently of model trajectories, this trend validates it as an algorithm-grounded challenge scale.

Table 5: Performance across workload levels (in terms of exact optimality / reference utility).

	Workload level			
Model	L1	L2	L3	L4
GPT-5.6 Terra	27.2 / 68.4	30.6 / 61.0	21.7 / 47.2	26.1 / 39.8
GPT-5.6 Sol	42.2 / 87.8	38.9 / 81.3	35.0 / 69.2	36.7 / 61.7
Qwen 3.5 Plus	4.4 / 20.3	10.0 / 17.1	4.4 / 15.3	2.8 / 10.6
DeepSeek V4 Pro	19.4 / 35.9	18.3 / 27.6	11.7 / 22.3	15.0 / 28.1
Claude Opus 4.8	44.4 / 84.0	41.1 / 76.7	32.2 / 69.8	36.7 / 62.1
Claude Sonnet 5	36.1 / 77.1	36.1 / 73.9	27.2 / 61.3	30.0 / 56.5
GLM 5.2	26.7 / 40.5	26.7 / 37.2	20.6 / 32.2	25.6 / 32.1

Reference utility also decreases at every successive level for six of the seven models.

Tool Interfaces. Table 6 compares exact-optimality scores under the Direct and Mediated interfaces. The Direct-minus-Mediated difference, denoted as $\Delta_{\mathrm{D-M}}$, is positive for all seven models, ranging from +0.6% to +6.4%. Despite these aggregate differences, 315–343 of the 360 paired evaluations for each model have the same exact-optimality status under both interfaces. The 95% confidence interval for $\Delta_{\mathrm{D-M}}$ excludes zero only for GPT-5.6 Sol and Claude Sonnet 5. Overall, exact-optimality outcomes are broadly stable across the paired interfaces, with a distinguishable Direct advantage for these two models. Additional experimental results are provided in Appendix J.

Table 6: Matched Direct–Mediated exact-optimality results. Direct and Mediated are mean rates (%) across three trials, and $\Delta_{\mathrm{D-M}}$ is their difference in percentage points. Direct wins, Mediated wins, and ties count the 360 matched instance–trial pairs for each model. The intervals are 95% paired bootstrap confidence intervals for $\Delta_{\mathrm{D-M}}$; complete statistical details appear in Appendix J.4.

	Exact-optimality comparison				Matched outcomes		
Model	**Direct**	**Mediated**	$\Delta_{\mathrm{D-M}}$	**95% paired bootstrap interval**	**Direct wins**	**Mediated wins**	**Ties**
GPT-5.6 Terra	27.2	25.6	+1.7	$[-1.9, +5.3]$	19	13	328
GPT-5.6 Sol	40.6	35.8	+4.7	$[+1.4, +8.3]$	25	8	327
Qwen 3.5 Plus	6.1	4.7	+1.4	$[-0.8, +3.9]$	11	6	343
DeepSeek V4 Pro	16.4	15.8	+0.6	$[-1.9, +3.1]$	15	13	332
Claude Opus 4.8	39.2	38.1	+1.1	$[-1.7, +3.9]$	17	13	330
Claude Sonnet 5	35.6	29.2	+6.4	$[+2.2, +11.1]$	34	11	315
GLM 5.2	26.1	23.6	+2.5	$[-0.8, +6.1]$	20	11	329

6 LIMITATIONS

While we show the challenges lie in ALGOWORLDS, there are some limitations worth noting: (1) Our experiments are based on seven leading LLMs, and the results may reflect inherent biases present in these models. Besides, the results only characterize LLMs with information tools, and providing them with more external tools (*e.g.*, a general-purpose code execution environment or an external optimization solver) might enhance their performance in ALGOWORLDS. (2) All worlds are formally generated and have unique global optima. This approach allows for precise verification; however, it fails to account for the noise, ambiguity, and temporal fluctuations inherent in real-world information sources, as well as decision problems that may have multiple globally optimal solutions. (3) The trajectory diagnostics assess the gathered information against sufficient information sets. However, these sets may not encompass all possible methods for determining the optimum. As a result, a trajectory that does not fully include any of these sets might still possess enough information to facilitate a globally optimal decision.

7 CONCLUSION

We introduce ALGOWORLDS, a benchmark for evaluating whether LLM agents can turn information acquired through tools into globally optimal decisions. It comprises 120 hidden optimization instances spanning ten formally specified combinatorial optimization families and four workload levels. Each instance has an independently verified unique global optimum and is presented through paired Direct and Mediated interfaces, yielding 240 algorithmic worlds. Our extensive experiments with seven leading LLMs reveal that the top-performing model, Claude Opus 4.8, achieves an exact-optimality score of only 38.61%, despite demonstrating significantly higher feasibility. This suggests that even when LLMs gather sufficient information to identify the global optimum, most still fail to achieve it. These findings highlight that acquiring adequate information and adhering to task constraints do not necessarily guarantee consistent global optimization. The challenge of making globally optimal decisions remains unresolved for LLM agents.

AI USE STATEMENT

Generative AI tools were used only for editing and improving software code and for improving the language and readability of the manuscript. They were not used to design the research methodology or experiments, implement the proposed methods, formulate or prove mathematical claims, generate benchmark instances or datasets, interpret experimental results, or develop the scientific conclusions. All AI-assisted code and text were reviewed and verified by the authors, who take full responsibility for the final content.

REPRODUCIBILITY STATEMENT

ALGOWORLDS is designed to provide exceptionally strong reproducibility and auditability. Benchmark instances are generated deterministically from fixed configurations, while the benchmark construction, optimal solutions, and evaluation records are systematically verified through independent and consistency checks. The evaluation protocol, model configurations, metrics, and aggregation procedures are fixed and clearly specified. As a result, the complete benchmark construction and evaluation pipeline is highly repeatable, independently verifiable, and reproducible under the same experimental settings.

REFERENCES

Anthropic. Introducing Claude Opus 4.8, May 2026a. URL https://www.anthropic.com/news/claude-opus-4-8.

Anthropic. Introducing Claude Sonnet 5, June 2026b. URL https://www.anthropic.com/news/claude-sonnet-5.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025. URL https://arxiv.org/abs/2506.07982.

Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Yuefeng Huang, Wulong Liu, Xinzhi Wang, Defu Lian, Baoqun Yin, Yasheng Wang, and Wu Liu. ACEBench: Who wins the match point in tool usage? *arXiv preprint arXiv:2501.12851*, 2025a. URL https://arxiv.org/abs/2501.12851.

Hongzheng Chen, Yingheng Wang, Yaohui Cai, Hins Hu, Jiajie Li, Shirley Huang, Chenhui Deng, Rongjian Liang, Shufeng Kong, Haoxing Ren, Samitha Samaranayake, Carla P. Gomes, and Zhiru Zhang. HeuriGym: An agentic benchmark for LLM-crafted heuristics in combinatorial optimization. *arXiv preprint arXiv:2506.07972*, 2025b. URL https://arxiv.org/abs/2506.07972.

DeepSeek-AI. DeepSeek V4 model card, April 2026. URL https://fe-static.deepseek.com/chat/transparency/deepseek-V4-model-card-EN.pdf.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks? *arXiv preprint arXiv:2403.07718*, 2024. URL https://arxiv.org/abs/2403.07718.

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. *arXiv preprint arXiv:2405.17743*, 2024a. URL https://arxiv.org/abs/2405.17743.

Xuhan Huang, Qingning Shen, Yan Hu, Anningzhe Gao, and Benyou Wang. LLMs for mathematical modeling: Towards bridging the gap between natural and mathematical languages. *arXiv preprint arXiv:2405.13144*, 2024b. URL https://arxiv.org/abs/2405.13144.

Harsha Kokel, Michael Katz, Kavitha Srinivas, and Shirin Sohrabi. ACPBench: Reasoning about action, change, and planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39

(25):26559–26568, 2025. doi: 10.1609/aaai.v39i25.34857. URL https://ojs.aaai.org/index.php/AAAI/article/view/34857.

Junlong Li, Wenshuo Zhao, Jian Zhao, Weihao Zeng, Haoze Wu, Xiaochen Wang, Rui Ge, Yuxuan Cao, Yuzhen Huang, Wei Liu, Junteng Liu, Zhaochen Su, Yiyang Guo, Fan Zhou, Lueyang Zhang, Juan Michelini, Xingyao Wang, Xiang Yue, Shuyan Zhou, Graham Neubig, and Junxian He. The tool decathlon: Benchmarking language agents for diverse, realistic, and long-horizon task execution. *arXiv preprint arXiv:2510.25726*, 2025. URL https://arxiv.org/abs/2510.25726.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://arxiv.org/abs/2304.08244.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. *arXiv preprint arXiv:2308.03688*, 2023. URL https://arxiv.org/abs/2308.03688.

Hongliang Lu, Zhonglin Xie, Yaoyu Wu, Can Ren, Yuxuan Chen, and Zaiwen Wen. OptMATH: A scalable bidirectional data synthesis framework for optimization modeling. *arXiv preprint arXiv:2502.11102*, 2025a. URL https://arxiv.org/abs/2502.11102.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, 2025b. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. AgentBoard: An analytical evaluation board of multi-turn LLM agents. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=4S8agvKjle.

Larisa Markeeva, Sean McLeish, Borja Ibarz, Wilfried Bounsi, Olga Kozlova, Alex Vitvitskyi, Charles Blundell, Tom Goldstein, Avi Schwarzschild, and Petar Veličković. The CLRS-text algorithmic reasoning language benchmark. *arXiv preprint arXiv:2406.04229*, 2024. URL https://arxiv.org/abs/2406.04229.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. *arXiv preprint arXiv:2311.12983*, 2023. URL https://arxiv.org/abs/2311.12983.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition, July 2026. URL https://openai.com/index/gpt-5-6/.

Peter Pirolli and Stuart Card. Information foraging. *Psychological Review*, 106(4):643–675, 1999. doi: 10.1037/0033-295X.106.4.643.

Hongjin Qian and Zheng Liu. Scent of knowledge: Optimizing search-enhanced reasoning with information foraging. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2505.09316.

Tian Qin, Felix Bai, Ting-Yao Hu, Raviteja Vemulapalli, Hema Swetha Koppula, Zhiyang Xu, Bowen Jin, Mert Cemri, Jiarui Lu, Zirui Wang, and Meng Cao. COMPASS: Benchmarking constrained optimization in LLM agents. *arXiv preprint arXiv:2510.07043*, 2025. URL https://arxiv.org/abs/2510.07043.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2307.16789.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Rindranirina Ramamonjison, Timothy T. Yu, Raymond Li, Haley Li, Giuseppe Carenini, Bissan Ghaddar, Shiqi He, Mahdi Mostajabdaveh, Amin Banitalebi-Dehkordi, Zirui Zhou, and Yong Zhang. NL4Opt competition: Formulating optimization problems based on their natural language descriptions. *arXiv preprint arXiv:2303.08233*, 2023. URL https://arxiv.org/abs/2303.08233.

Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. TaskBench: Benchmarking large language models for task automation. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=bAxUA5r3Ss.

Haoyu Sun, Wenxuan Wang, Mingyang Song, Jujie He, Weinan Zhang, Yang Liu, Yang Yang, and Yu Cheng. Agent planning benchmark: A diagnostic framework for planning capabilities in LLM agents. *arXiv preprint arXiv:2606.04874*, 2026. URL https://arxiv.org/abs/2606.04874.

Weiwei Sun, Shengyu Feng, Shanda Li, and Yiming Yang. CO-Bench: Benchmarking language model agents in algorithm search for combinatorial optimization. *arXiv preprint arXiv:2504.04310*, 2025. URL https://arxiv.org/abs/2504.04310.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://arxiv.org/abs/2407.18901.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. PlanBench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2023. URL https://arxiv.org/abs/2206.10498.

Yanbo Wang, Zixiang Xu, Yue Huang, Xiangqi Wang, Zirui Song, Lang Gao, Chenxi Wang, Robert Tang, Yue Zhao, Arman Cohan, Xiangliang Zhang, and Xiuying Chen. DyFlow: Dynamic workflow framework for agentic reasoning. In *Advances in Neural Information Processing Systems*, volume 38, 2025a. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/fe9910d2b03324faeb5371a9658277bb-Abstract-Conference.html.

Yunhan Wang, Jiaan Wang, Lianzhe Huang, Xianfeng Zeng, and Fandong Meng. Evobrowsecomp: Benchmarking search agents on evolving knowledge. *arXiv preprint arXiv:2606.13120*, 2026.

Zhenting Wang, Qi Chang, Hemani Patel, Shashank Biju, Cheng-En Wu, Quan Liu, Aolin Ding, Alireza Rezazadeh, Ankit Shah, Yujia Bao, and Eugene Siow. MCP-Bench: Benchmarking tool-using LLM agents with complex real-world tasks via MCP servers. *arXiv preprint arXiv:2508.20453*, 2025b. URL https://arxiv.org/abs/2508.20453.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. BrowseComp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025. URL https://arxiv.org/abs/2504.12516.

Ryan Z. Y. Wong, Jiawei Wang, Junjie Zhao, Li Chen, Yan Gao, Long Zhang, Xuan Zhou, Zuo Wang, Kai Xiang, Ge Zhang, Wenhao Huang, Yang Wang, and Ke Wang. WideSearch: Benchmarking agentic broad info-seeking. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust

(eds.), *International Conference on Learning Representations*, volume 2026, pp. 10012–10086, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/1119587863e78451f080da2a768c4935-Paper-Conference.pdf.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. TravelPlanner: A benchmark for real-world planning with language agents. In *International Conference on Machine Learning*, 2024a. URL https://openreview.net/forum?id=l5XQzNkAOe.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Jing Hua Toh, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*, 2024b. URL https://arxiv.org/abs/2404.07972.

Zixiang Xu, Yanbo Wang, Yue Huang, Haomin Zhuang, Yujun Zhou, Jiayi Ye, Sixian Li, Zirui Song, Lang Gao, Chenxi Wang, Zhaorun Chen, Wang Pan, Yue Zhao, Jieyu Zhao, Xiangliang Zhang, and Xiuying Chen. SocialMaze: A benchmark for evaluating and enhancing social reasoning in large language models in complex social environments. *arXiv preprint arXiv:2505.23713*, 2025. doi: 10.48550/arXiv.2505.23713. URL https://arxiv.org/abs/2505.23713.

Zixiang Xu, Yanbo Wang, Chenxi Wang, Lang Gao, Zirui Song, Yue Huang, Zhaorun Chen, Xiangliang Zhang, and Xiuying Chen. GTA: Graph theory agent and benchmark for algorithmic graph reasoning with LLMs. OpenReview preprint, 2026. URL https://openreview.net/forum?id=bcGClKY3gQ.

Fanjia Yan, Huanzhi Mao, Charlie Cheng-Jie Ji, Tianjun Zhang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Berkeley function-calling leaderboard. UC Berkeley Sky Computing Lab, 2024. URL https://gorilla.cs.berkeley.edu/leaderboard.

Zhicheng Yang, Yiwei Wang, Yinya Huang, Zhijiang Guo, Wei Shi, Xiongwei Han, Liang Feng, Linqi Song, Xiaodan Liang, and Jing Tang. OptiBench meets ReSocratic: Measure and improve LLMs for optimization modeling. *arXiv preprint arXiv:2407.09887*, 2024. URL https://arxiv.org/abs/2407.09887.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2406.12045.

Ori Yoran, Samuel Joseph Amouyal, Chaitanya Malaviya, Ben Bogin, Ofir Press, and Jonathan Berant. AssistantBench: Can web agents solve realistic and time-consuming tasks? *arXiv preprint arXiv:2407.15711*, 2024. URL https://arxiv.org/abs/2407.15711.

Z.ai. GLM-5.2: Built for long-horizon tasks, June 2026. URL https://z.ai/blog/glm-5.2.

Xinyu Zhang, Boxuan Zhang, Yuchen Wan, Lingling Zhang, YiXing Yao, Bifan Wei, Yaqiang Wu, and Jun Liu. OptiVerse: A comprehensive benchmark towards optimization problem solving. *arXiv preprint arXiv:2604.21510*, 2026a. URL https://arxiv.org/abs/2604.21510.

Yinger Zhang, Shutong Jiang, Renhao Li, Jianhong Tu, Yang Su, Lianghao Deng, Xudong Guo, Chenxu Lv, and Junyang Lin. DeepPlanning: Benchmarking long-horizon agentic planning with verifiable constraints. *arXiv preprint arXiv:2601.18137*, 2026b. URL https://arxiv.org/abs/2601.18137.

Zhenliang Zhang, Wenqing Wang, Yong Hu, Yaming Yang, Jiaheng Gao, Chen Shen, and Xiaojun Wan. SCOUT: Active information foraging for long-text understanding with decoupled epistemic states. In *International Conference on Machine Learning*, 2026c. URL https://xavierzhang2002.github.io/scout-page/.

Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, and Denny Zhou. NATURAL PLAN: Benchmarking LLMs on natural language planning. *arXiv preprint arXiv:2406.04520*, 2024. URL https://arxiv.org/abs/2406.04520.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2307.13854.

A Extended Related Work

Tool use and interactive agents. Tool-use benchmarks evaluate whether an LLM agent can select an appropriate tool, supply valid arguments, maintain state across turns, and complete a task using external systems. API-Bank (Li et al., 2023) and ToolLLM (Qin et al., 2024) provide broad evaluations of API selection and argument construction. BFCL (Yan et al., 2024) focuses on function-calling accuracy, while ACEBench (Chen et al., 2025a) extends evaluation across tools, agents, and domains. Interactive benchmarks extend evaluation from isolated calls to sustained interaction. ToolSandbox (Lu et al., 2025b) records stateful execution and intermediate milestones. τ-bench (Yao et al., 2025) and τ^2-bench (Barres et al., 2025) evaluate policy-constrained agent–user interaction. The Tool Decathlon (Li et al., 2025; Wang et al., 2025a) covers diverse long-horizon workflows, while TaskBench (Shen et al., 2024) and MCP-Bench (Wang et al., 2025b) study task decomposition and coordination across tools or servers. These benchmarks together test whether agents can operate tools and complete workflows. AlgoWorlds builds on these capabilities by asking whether an agent can use information acquired through tools to identify a globally optimal decision.

Partially observed environments and information seeking. This line of work studies reasoning and action when task information is not fully available at the beginning (Xu et al., 2025). AgentBench (Liu et al., 2023) provides interactive environments for evaluating general agents, and AgentBoard (Ma et al., 2024) provides trajectory-level analysis of their progress. WebArena (Zhou et al., 2024), WorkArena (Drouin et al., 2024), OSWorld (Xie et al., 2024b), and AppWorld (Trivedi et al., 2024) ground interaction in executable web, desktop, or application states. GAIA (Mialon et al., 2023), AssistantBench (Yoran et al., 2024), BrowseComp (Wei et al., 2025), WideSearch (Wong et al., 2026), and EvoBrowseComp (Wang et al., 2026) emphasize retrieval and synthesis across multiple sources. Information foraging theory (Pirolli & Card, 1999) provides a complementary account of how information-seeking behavior responds to the value and cost of information. InForage (Qian & Liu, 2025) and SCOUT (Zhang et al., 2026c) bring this perspective to search-enhanced reasoning and long-context information acquisition. In AlgoWorlds, task-specific information tools expose well-defined facts from a hidden optimization instance. For each algorithmic world, executable tool-use plans yield fact sets sufficient to reconstruct that instance. Trajectory diagnostics measure coverage of these sets, while the final decision is evaluated against a formal objective and an exact global optimum rather than a textual answer or workflow state.

Planning with consequential decisions. Planning benchmarks evaluate whether an agent can produce actions or decisions that remain coherent under constraints and have consequences beyond an individual step. PlanBench (Valmeekam et al., 2023) and ACPBench (Kokel et al., 2025) express symbolic planning problems in controlled language. NATURAL PLAN (Zheng et al., 2024) evaluates planning over records provided in context. TravelPlanner (Xie et al., 2024a) requires agents to gather information and construct feasible itineraries under real-world constraints. DeepPlanning (Zhang et al., 2026b) combines proactive information acquisition with long, constrained workflows, while the Agent Planning Benchmark (Sun et al., 2026) tests planning in environments with irrelevant or malfunctioning tools. COMPASS (Qin et al., 2025) evaluates both feasibility and utility after conversational or tool-mediated information gathering. These works show that successful planning requires more than valid individual actions. AlgoWorlds focuses specifically on globally optimal decision making: after gathering information through tools, the agent provides a structured decision whose feasibility, objective value, and exact global optimality can be evaluated.

Algorithmic and optimization reasoning. Work on algorithmic and optimization reasoning studies whether language models can execute algorithms, formulate optimization problems, or construct procedures for solving them (Xu et al., 2026). CLRS-Text (Markeeva et al., 2024) evaluates textual execution of classical algorithms under controlled input scaling. NL4Opt (Ramamonjison et al., 2023) and Mamo (Huang et al., 2024b) study the translation of natural-language descriptions into mathematical optimization models. OptiBench (Yang et al., 2024) and ORLM (Huang et al., 2024a) evaluate automated optimization modeling, while OptMATH (Lu et al., 2025a) and OptiVerse (Zhang et al., 2026a) broaden the coverage of optimization problems and solution processes. CO-Bench (Sun et al., 2025) asks agents to generate algorithms for combinatorial optimization, and HeuriGym (Chen et al., 2025b) evaluates executable heuristic construction. In these settings, the problem instance is typically available before optimization, or the generated solver is itself the target. AlgoWorlds

instead asks an agent to acquire information about a hidden instance through task-specific information tools and then provide one structured decision. Its exact algorithms support benchmark construction and certify the global optima used for scoring. They are not tools available to the evaluated agent.

B A WORKED FLEET DISPATCH PAIR

This section gives a concrete view of one released Fleet Dispatch pair at workload level 1 and instance index 1. It focuses on what the agent receives, how the paired tools present the hidden information, and how the agent submits its final decision. We denote its Direct and Mediated worlds by $w^{(\mathrm{D})}$ and $w^{(\mathrm{M})}$. The shared instance contains 14 jobs and five vehicles. For readability, we replace its opaque public identifiers with $J_1, \ldots, J_{14}$ and $V_1, \ldots, V_5$; all numerical values and relations are unchanged. Repeated records are omitted.

During evaluation, the agent receives the textual question, tool schemas and costs, and access budget, but not the hidden instance or validation results. The example distinguishes this model-visible content from the offline records used for construction and validation.

World components and visibility. The shared hidden instance $\mathcal{I}$ contains the job loads, eligible vehicle–job pairs, vehicle capacities, route costs, and fixed activation costs. This instance has 14 positive job loads, five vehicles with capacity 4, five activation costs, and a 5×14 route-cost matrix. The eligibility relation determines which matrix entries correspond to admissible assignments. For example, J_1 has load 1, the fixed activation cost of V_1 is 122,070,312,500, and the route cost from V_1 to J_1 is 6,103,515,625. These values belong to the hidden instance and become visible only when the corresponding information-tool calls return them. Interface-specific identifiers and serialized response records are not part of $\mathcal{I}$.

The textual question q lists the public job and vehicle identifiers. It asks the agent to assign every job to an eligible vehicle without exceeding vehicle capacity and to minimize the Fleet Dispatch objective $g_{\mathcal{I}}$ defined in Appendix C. This objective is the sum of route costs, activation costs, and the decision-independent offset $\gamma_{\mathcal{I}}$. The offset changes the reported objective value but not the minimizing mapping. The agent also receives the tool schemas in $\mathcal{T}$, their per-call costs under f_c, and the access budget $b = 97$. The final-decision tool requires a 14-entry job-to-vehicle mapping.

Table 7 summarizes the available tools and their costs in the paired worlds.

Tool	Arguments	Returned information or effect	$f_c(t)$
`read_fleet_policy`	None	Vehicle capacities and the complete vehicle and job rosters	2
`read_job`	One job	Its load and eligible vehicles, without route costs	1
`read_vehicle_contract`	One vehicle	Its fixed activation cost, without capacity or assigned jobs	4
`read_route_row`	One vehicle	Interface-specific route records associated with that vehicle	3
`read_dispatch_window`	Start job and length 2 or 3	Loads, eligible vehicles, and interface-specific route records for consecutive jobs	14
`evaluate_assignment`	One complete mapping	Feasibility and, when feasible, its objective value	5
`submit_assignment`	One complete mapping	Records the binding final decision and ends the trajectory	0

Table 7: The available tools in the worked Fleet Dispatch pair. The first five reveal instance information. `evaluate_assignment` provides nonterminal feedback and may be used once, whereas `submit_assignment` records the binding final decision.

Representative Direct and Mediated responses. Calling `read_job` on J_1 returns its load and the five eligible vehicles. In the Direct world, a route-row call on V_1 returns a record that includes the corresponding route cost:

```
read_job({job: J1})
  -> {job_id: J1, load: 1, eligible_vehicles: [V1, ..., V5]}
```

```
Direct route record:
  {vehicle_id: V1, job_id: J1, route_cost: 6103515625}
```

The Mediated world has the same textual question, tool names, argument schemas, costs, and budget. Its route handler exposes the same route fact through an intermediate shift:

```
Mediated route records:
  {vehicle_id: V1, shift_id: S1}
  {shift_id: S1, job_id: J1, route_cost: 6103515625}
```

Joining the two Mediated records on S_1 and then removing that interface-specific identifier yields the same route-cost fact as the Direct record. Both presentations therefore encode $(V_1, J_1, 6{,}103{,}515{,}625)$. The identifier S_1 appears in the Mediated response but is not part of $\mathcal{I}$. The general response-normalization procedure and the pair-level equivalence check are defined in Appendices F.2 and F.3.

Candidate evaluation and the binding decision. The `evaluate_assignment` tool accepts one complete mapping, costs five units, and may be called at most once. It reports feasibility and, for a feasible mapping, its objective value. The call is nonterminal and does not commit a final decision. Because it evaluates an agent-proposed mapping rather than revealing fields of the hidden instance, its feedback is not counted as instance information by the trajectory diagnostics.

Only `submit_assignment` is the final-decision tool t_{dec}. It accepts one 14-entry object mapping every J_i to a V_j, costs zero, and terminates the trajectory. The family checker verifies that every job appears exactly once, every selected vehicle–job pair is eligible, and no vehicle capacity is exceeded. For a feasible mapping, it recomputes the objective $g_{\mathcal{I}}$. The submitted mapping is the final decision y scored by the metrics in Section 3.4. A mapping previously passed to the candidate-evaluation tool is not binding unless it is later submitted through t_{dec}.

C Formal Problems and Exact Algorithms

This section formalizes the ten optimization families and describes the exact algorithms used during benchmark construction. For each family, we give the decision, feasibility conditions, objective, source of global coupling, algorithm, correctness argument, and complexity. The independent verification algorithms and the remaining validation procedures are described in Appendix F.

C.1 Transit Routing

Problem and global coupling. The instance has a positive residue modulus M and a target residue $r_\star \in \{0, \dots, M-1\}$. It contains d ordered stages, each with a set E_i of candidate legs. Let $\mathcal{L} = \{\ell(e) : e \in \bigcup_i E_i\}$ be the set of lines. A leg e has travel time $\tau(e)$, line $\ell(e) \in \mathcal{L}$, and residue $\rho(e) \in \{0, \dots, M-1\}$. A route $x = (e_1, \dots, e_d)$ selects one leg from each stage and is feasible when

$$\sum_{i=1}^{d} \rho(e_i) \equiv r_\star \pmod{M}, \qquad e_i \in E_i. \tag{7}$$

Its objective is

$$\min_x \sum_{i=1}^{d} \tau(e_i) + \sum_{i=2}^{d} C(\ell(e_{i-1}), \ell(e_i)), \tag{8}$$

where $C(\lambda, \lambda')$ is the directed transfer cost for an ordered pair of lines and need not be symmetric. The modular condition couples the residues of all selected legs, while each transfer cost couples two consecutive stages. A route prefix must therefore retain both its accumulated residue and its final line.

Exact algorithm and correctness. Let $D_i(\lambda, r)$ be the minimum cost of a prefix through stage i whose final line is λ and whose accumulated residue is r modulo M. Initialize every first-layer value to $+\infty$ and, for each $e \in E_1$, set

$$D_1(\ell(e), \rho(e)) \leftarrow \min\{D_1(\ell(e), \rho(e)), \tau(e)\}. \tag{9}$$

For $i \geq 2$, each reachable state (λ, r) at stage $i-1$ and each leg $e \in E_i$ induce the update

$$D_i(\ell(e), (r+\rho(e)) \bmod M) \leftarrow \min\Big\{ D_i(\ell(e), (r+\rho(e)) \bmod M)\,, \\ D_{i-1}(\lambda, r) + \tau(e) + C(\lambda, \ell(e)) \Big\}. \tag{10}$$

The optimum is $\min_{\lambda \in \mathcal{L}} D_d(\lambda, r_\star)$; stored predecessor states and legs reconstruct the route.

The state is sufficient because an earlier prefix can affect a continuation only through its final line and accumulated residue. Induction over the stages therefore shows that retaining the cheapest prefix for each state preserves an optimal complete route.

The construction algorithm evaluates the same recurrence using ordered predecessor scans. The ordering changes how candidate transitions are examined but not the recurrence. In the benchmark construction, $|\mathcal{L}| = p = k+2$ and $|E_i| = k$ for every stage. Including the $O(p^2 \log p)$ ordering step, the algorithm runs in $O(p^2 \log p + dk^2 M)$ time.

C.2 Basket Assembly

Problem and global coupling. Let A, B, and C be pairwise disjoint item groups with $|A| = |B| = |C| = n$, and let $U = A \cup B \cup C$. A finite catalog $\mathcal{E}$ contains tickets e with prices p_e; each ticket is either a singleton or a triple containing one item from each group. A decision $X \subseteq \mathcal{E}$ is feasible when it forms an exact cover:

$$\sum_{e \in X} \mathbb{1}[v \in e] = 1 \qquad \text{for every } v \in U. \tag{11}$$

The objective is to minimize $\sum_{e \in X} p_e$. Tickets interact through the exact-cover constraints: selecting one ticket simultaneously covers all of its items and excludes every other ticket containing any of them.

Exact algorithm and correctness. Fix an order on U, and represent each subset by a bitmask. For $S \subseteq U$ representing the items already covered exactly once, let $F(S)$ be the minimum additional price of tickets that are disjoint from S and cover every item in $U \setminus S$ exactly once. Set $F(U) = 0$. For $S \subsetneq U$, let $u(S)$ be the least-indexed uncovered item. The recurrence is

$$F(S) = \min_{\substack{e \in \mathcal{E},\ u(S) \in e \\ e \cap S = \varnothing}} \{ p_e + F(S \cup e) \}\,, \tag{12}$$

where a state with no admissible ticket has value $+\infty$. Every exact-cover completion of S contains exactly one admissible ticket covering $u(S)$. Conditioning on that ticket gives Eq. (12); induction on $|U \setminus S|$ therefore establishes that $F(S)$ is the minimum remaining cost. Stored minimizing tickets reconstruct the selected ticket set.

Let Δ be the maximum number of candidate tickets examined at one reached state. The algorithm runs in $O(2^{3n}\Delta)$ time.

C.3 Station Siting

Problem and global coupling. There are n ordered zones and two site options, labeled 0 and 1, per zone. Option a at zone i has cost $\kappa_i(a)$. A decision $x \in \{0,1\}^n$ must satisfy the shared capacity constraint $\sum_i x_i \leq H$. Let $\mathcal{D} \subseteq \{(i,j) : 1 \leq i < j \leq n,\ j - i \leq \omega\}$ be the set of demand relations. Each $(i,j) \in \mathcal{D}$ contributes reward r_{ij} exactly when its endpoint choices equal a required pair $(\alpha_{ij}, \beta_{ij}) \in \{0,1\}^2$. The objective is

$$\max_{x \in \{0,1\}^n} \left\{ \sum_{(i,j) \in \mathcal{D}} r_{ij} \mathbb{1}[(x_i, x_j) = (\alpha_{ij}, \beta_{ij})] - \sum_{i=1}^{n} \kappa_i(x_i) \right\} \quad \text{s.t.} \quad \sum_{i=1}^{n} x_i \leq H. \tag{13}$$

All option-one choices consume the same capacity, and each demand reward depends jointly on two zone choices. These shared terms prevent the zones from being optimized independently.

Exact algorithm and correctness. Every optimal decision in the generated instances uses exactly H option-one sites. This construction property is checked by the independent verification algorithm, which retains the original at-most-H capacity constraint (Table 10). Writing $\bar{H} = n - H$, the construction algorithm can therefore search decisions with exactly $\bar{H}$ option-zero choices without changing the optimal value. The algorithm processes zones from left to right and records the number of option-zero choices used together with the most recent ω binary choices. The empty prefix has value zero and uses no option-zero choices. Appending a choice subtracts its site cost, increments the counter when that choice is zero, and closes every demand term whose right endpoint is the new zone. States that cannot be extended to exactly $\bar{H}$ option-zero choices are discarded, and the terminal state must have counter $\bar{H}$.

Choices older than the frontier cannot occur in a future demand term because of the span bound. The counter retains the past information needed by the shared capacity constraint under this construction property. Each site cost is subtracted when its zone is processed, and each demand reward is added once when its right endpoint enters the frontier. Induction over the zones shows that the best value for each state preserves an optimal completion; backpointers reconstruct the site choices.

The algorithm runs in $O(n(\bar{H}+1)2^{\omega}(\omega+1))$ time and $O((\bar{H}+1)2^{\omega})$ value-only memory.

C.4 Authorization Planning

Problem and global coupling. The instance has a positive residue modulus M, a target residue $R \in \{0, \dots, M-1\}$, d ordered stages, a package set $\mathcal{V}_i$ at each stage, and authorized transitions $\mathcal{E}_i \subseteq \mathcal{V}_i \times \mathcal{V}_{i+1}$ between consecutive stages. A decision is a sequence $x = (v_1, \dots, v_d)$ with $v_i \in \mathcal{V}_i$. It is feasible when $(v_i, v_{i+1}) \in \mathcal{E}_i$ for every $i < d$ and

$$\sum_{i=1}^{d} z(v_i) \equiv R \pmod{M}, \tag{14}$$

where $z(v) \in \{0, \dots, M-1\}$ is the clearance increment of package v. Each package also has a base price $p(v)$ and a policy code $\mathrm{code}(v)$, whose adjustment is $\eta[\mathrm{code}(v)]$. Let $h(u, v)$ be the handoff cost of an authorized transition. The objective is

$$\min_{x} \sum_{i=1}^{d} (p(v_i) + \eta[\mathrm{code}(v_i)]) + \sum_{i=1}^{d-1} h(v_i, v_{i+1}). \tag{15}$$

The preceding package determines the feasibility and handoff cost of the next selection, while the modular condition couples the clearance increments across the complete sequence.

Exact algorithm and correctness. Let $D_i(v, r)$ be the minimum cost of a feasible prefix through stage i that ends at package $v \in \mathcal{V}_i$ and has accumulated clearance residue r modulo M. Initialize all values to $+\infty$ and, for every $v \in \mathcal{V}_1$, set

$$D_1(v, z(v)) = p(v) + \eta[\mathrm{code}(v)]. \tag{16}$$

For each $(u, v) \in \mathcal{E}_i$ and residue r, the transition is

$$\begin{aligned} D_{i+1}(v, (r + z(v)) \bmod M) \leftarrow \min\Big\{ & D_{i+1}(v, (r + z(v)) \bmod M), \\ & D_i(u, r) + p(v) + \eta[\mathrm{code}(v)] + h(u, v) \Big\}. \end{aligned} \tag{17}$$

The optimum is $\min_{v \in \mathcal{V}_d} D_d(v, R)$; stored predecessors reconstruct the package sequence.

The state is sufficient because the final package determines the authorization and handoff effects on the next stage. The accumulated residue is the only other prefix information needed to enforce the terminal residue condition. The transition enumerates every authorized extension and adds each package and handoff term exactly once. Induction over the stages therefore establishes that the minimum terminal value is the global optimum. Let $V = \sum_{i=1}^{d} |\mathcal{V}_i|$ and $E = \sum_{i=1}^{d-1} |\mathcal{E}_i|$. The algorithm runs in $O(M(V+E))$ time.

C.5 SERIES PORTFOLIO

Problem and global coupling. Each of n titles has two lots, represented by a binary choice $x_i \in \{0,1\}$, where lot z of title i has value $v_{i,z}$. Let $\mathcal{F}$ be a finite set of signed factors. Each $s \in \mathcal{F}$ has a nonempty scope $P_s \subseteq \{1,\dots,n\}$ satisfying $\max P_s - \min P_s + 1 \le \omega$ and a complete table $A_s : \{0,1\}^{|P_s|} \to \mathbb{Z}$. The objective is

$$\max_{x\in\{0,1\}^n} \left\{ \sum_{i=1}^{n} v_{i,x_i} - C_{\text{title}} + \sum_{s\in\mathcal{F}} A_s(x|_{P_s}) \right\}, \qquad C_{\text{title}} = \sum_{i=1}^{n} \min_{z\in\{0,1\}} v_{i,z}. \tag{18}$$

The offset C_{title} is independent of x and therefore does not change the maximizing portfolio. Overlapping factor scopes make the value of one title depend on choices at several nearby titles.

Exact algorithm and correctness. The construction algorithm uses suffix states. Let $B_i(\sigma)$ be the maximum sum of the first i lot values and all factor values whose scopes end at or before title i, among prefixes whose latest $\min\{i, \omega-1\}$ choices form suffix σ. Initialize $B_0(\varnothing) = 0$ and all other values to $-\infty$. To advance from title $i-1$ to title i, the algorithm appends a choice $z \in \{0,1\}$, adds $v_{i,z}$ and every factor $s \in \mathcal{F}$ with $\max P_s = i$, and maximizes into the state retaining the latest $\omega - 1$ choices. Each factor is thus added exactly once, when its complete scope is known. The optimum is $\max_\sigma B_n(\sigma) - C_{\text{title}}$.

Because every scope has span at most ω, no future factor can involve a choice older than the retained suffix. Induction over title positions therefore establishes that the best value for each suffix state preserves a globally optimal completion; stored predecessors reconstruct the binary portfolio. Let $f_\omega = \max_i |\{s \in \mathcal{F} : \max P_s = i\}|$. The algorithm runs in $O(n2^\omega(1+f_\omega))$ time. Rolling value computation uses $O(2^\omega)$ memory, while retaining reconstruction information uses $O(n2^\omega)$ memory.

C.6 MACHINE LAYOUT

Problem and global coupling. A feasible layout is a bijection π from n machines to n linearly ordered, unit-spaced slots, with $d_{st} = |s-t|$. Let $F_{ab} = F_{ba}$ be the throughput between machines a and b, and let $A_{a,s}$ be the adjustment for placing machine a in slot s. The objective is

$$\min_{\pi} \sum_{a<b} F_{ab} d_{\pi(a),\pi(b)} + \sum_{a} A_{a,\pi(a)}. \tag{19}$$

The cost of placing one machine depends on its distance from every other machine with which it exchanges flow. The placement decisions are therefore coupled through the complete permutation rather than through independent machine–slot scores.

Exact algorithm and correctness. For a subset S of machines, define

$$\text{cut}(S) = \sum_{a\in S} \sum_{b\notin S} F_{ab}. \tag{20}$$

For a left-to-right order with prefix sets S_t, the pairwise flow–distance cost is $\sum_{t=1}^{n-1} \text{cut}(S_t)$. Because $d_{st} = |s-t|$, a machine pair appears in one prefix cut for each unit slot boundary between its two positions. Summing the cuts therefore reproduces the complete flow–distance cost. With $\text{dp}[\varnothing] = 0$, the recurrence

$$\text{dp}[S] = \min_{a\in S} \big\{ \text{dp}[S\setminus\{a\}] + A_{a,|S|} + \mathbb{1}[|S| < n]\,\text{cut}(S) \big\} \tag{21}$$

conditions on the machine a placed in slot $|S|$. The machines in $S\setminus\{a\}$ must optimally fill the preceding slots. Each prefix cut and placement adjustment is added exactly once, so induction on $|S|$ proves the recurrence. The optimal value is $\text{dp}[\{1,\dots,n\}]$, and stored minimizing machines reconstruct the layout in reverse order. All cut values can be precomputed by extending subsets one machine at a time. This takes $O(n2^n)$ time and $O(2^n)$ space, after which each candidate in Eq. (21) takes constant time. The algorithm therefore runs in $O(n2^n)$ time and uses $O(2^n + n^2)$ memory.

C.7 Sequential Matching

Problem and global coupling. Let $L = (1, \dots, n)$ be the fixed order of the left participants, let R be a set of n right participants, and let $E \subseteq L \times R$ contain the compatible pairs. A feasible decision is a permutation $\pi = (\pi_1, \dots, \pi_n)$ of R satisfying $(i, \pi_i) \in E$ for every position i. Each compatible pair $(i, j) \in E$ has unary match cost $c(i, j)$, and every ordered pair of distinct right participants (j, t) has directed handoff cost $h(j, t)$. The objective is

$$\min_{\pi} \sum_{i=1}^{n} c(i, \pi_i) + \sum_{i=2}^{n} h(\pi_{i-1}, \pi_i). \tag{22}$$

The bijection constraint couples assignments through the right participants already used, while each handoff cost couples two consecutive matches.

Exact algorithm and correctness. For $S \subseteq R$ and $j \in S$, let $D[S, j]$ be the minimum cost of matching the first $|S|$ left participants compatibly to exactly the right participants in S, with j assigned at position $|S|$. The initial states are

$$D[\{j\}, j] = c(1, j) \qquad \text{for every } (1, j) \in E, \tag{23}$$

and all other states have value $+\infty$. For $t \in R \setminus S$ satisfying $(|S| + 1, t) \in E$,

$$D[S \cup \{t\}, t] = c(|S| + 1, t) + \min_{j \in S} \{D[S, j] + h(j, t)\}. \tag{24}$$

The optimal value is $\min_{j \in R} D[R, j]$. Every finite state represents a compatible prefix with a specified set of right participants and a specified final participant. The recurrence considers every compatible extension of that prefix. Induction on $|S|$ therefore proves that each state retains the cheapest compatible prefix with the same assigned set and final participant. Stored predecessors reconstruct the matching.

There are at most $n2^{n-1}$ states (S, j), and the running time is $O(n^2 2^n)$.

C.8 Fleet Dispatch

Problem and global coupling. Let $J = \{1, \dots, n\}$ be the jobs, let $V = \{1, \dots, m\}$ be the vehicles, and let $E \subseteq V \times J$ contain the eligible vehicle–job pairs. Job j has positive integer load λ_j. Vehicle v has integer capacity $C_v \geq 0$, fixed activation cost f_v, and route cost r_{vj} for each $(v, j) \in E$. Binary variable x_{vj} records an assignment. Vehicle activation is determined by whether it receives at least one job. The objective includes a decision-independent instance offset $\gamma_{\mathcal{I}}$:

$$g_{\mathcal{I}}(x) = \gamma_{\mathcal{I}} + \sum_{(v,j) \in E} r_{vj} x_{vj} + \sum_{v \in V} f_v \mathbb{1}\left[\sum_{j:(v,j) \in E} x_{vj} > 0\right]. \tag{25}$$

The problem minimizes $g_{\mathcal{I}}(x)$ subject to

$$\begin{aligned} \sum_{v:(v,j) \in E} x_{vj} &= 1 && \forall j \in J, \\ \sum_{j:(v,j) \in E} \lambda_j x_{vj} &\leq C_v && \forall v \in V, \\ x_{vj} &\in \{0, 1\} && \forall (v, j) \in E. \end{aligned} \tag{26}$$

Jobs assigned to the same vehicle consume a shared capacity and share one activation cost. Assigning one job therefore changes both the remaining capacity and the marginal cost of later assignments to that vehicle.

Exact algorithm and correctness. Process jobs in their fixed order. Let $\boldsymbol{\ell} = (\ell_1, \dots, \ell_m)$ be the assigned vehicle loads and let $D_i(\boldsymbol{\ell})$ be the minimum cost of assigning the first i jobs with that load vector, excluding the decision-independent offset $\gamma_{\mathcal{I}}$. The vector is encoded as one mixed-radix index

with radices $C_v + 1$. Initialize $D_0(\mathbf{0}) = 0$ and every other state to $+\infty$. For every reached state $\boldsymbol{\ell}$ and every $(v, i+1) \in E$ satisfying $\ell_v + \lambda_{i+1} \le C_v$, update

$$D_{i+1}(\boldsymbol{\ell} + \lambda_{i+1}\mathbf{e}_v) \leftarrow \min\Big\{ D_{i+1}(\boldsymbol{\ell} + \lambda_{i+1}\mathbf{e}_v)\,, \\ D_i(\boldsymbol{\ell}) + r_{v,i+1} + f_v \mathbb{1}[\ell_v = 0] \Big\}. \tag{27}$$

Because loads are positive, $\ell_v = 0$ holds exactly when v has not yet been activated. Given the next job index, the load vector determines all remaining capacities and which activation costs have already been incurred. No other property of the prefix affects a continuation. Every feasible assignment of the first $i+1$ jobs assigns job $i+1$ to exactly one vehicle, so the recurrence considers every extension at this step. Induction over the job order proves the recurrence. The optimal objective value is $\gamma_{\mathcal{I}} + \min_{\boldsymbol{\ell}} D_n(\boldsymbol{\ell})$, and stored predecessors reconstruct the job-to-vehicle mapping.

Writing $P = \prod_{v=1}^{m}(C_v + 1)$, the algorithm runs in $O(nmP)$ time and uses $O(nP + nm)$ memory when storing reconstruction information.

C.9 Evidence-Joined Routing

Problem and global coupling. The instance has a positive residue modulus M, a target residue $R \in \{0, \dots, M-1\}$, and a layered directed acyclic graph $G = (V, E)$ with source s in the first layer and target t in the last. Every edge points from an earlier layer to a later layer. Each edge e has quantity Q_e, exception class χ_e, residue $\rho_e \in \{0, \dots, M-1\}$, and tariff $T(\chi_e)$. The quantity and exception-class tariff define the nonnegative edge cost $\kappa_e = Q_e T(\chi_e)$. A feasible decision is an s–t path P whose accumulated residue satisfies

$$\sum_{e \in P} \rho_e \equiv R \pmod{M}, \tag{28}$$

and the objective is to minimize $\sum_{e \in P} \kappa_e$. Edge costs combine quantity, exception-class, and tariff data, while path feasibility depends on the residue accumulated across all selected edges.

Exact algorithm and correctness. Let $D(v, r)$ be the minimum cost of a path from s to v with residue r. Initialize $D(s, 0) = 0$ and all other states to $+\infty$. Traversing an edge $e = (u, v)$ induces the expanded-state relaxation

$$D(v, (r + \rho_e) \bmod M) \leftarrow \min\Big\{ D(v, (r + \rho_e) \bmod M)\,, \\ D(u, r) + \kappa_e \Big\}. \tag{29}$$

Processing the layered graph in topological order therefore computes $D(t, R)$, and stored predecessors reconstruct the path. The state records exactly the endpoint and accumulated residue on which a continuation depends, so induction over the graph layers establishes the recurrence's correctness.

Writing $n_V = |V|$ and $n_E = |E|$ for the numbers of vertices and edges in the graph, the algorithm runs in $O(M(n_V + n_E))$ time.

C.10 Migration Portfolio

Problem and global coupling. Let $\mathcal{D} = \{1, \dots, n_D\}$ be the support domains, and let $\mathcal{P}_i$ be the package catalog for component group $i \in \{1, \dots, G\}$. Package $p \in \mathcal{P}_i$ has cost c_p and requirement set $R(p) \subseteq \mathcal{D}$. A decision $y = (S, p_1, \dots, p_G)$ activates a domain set $S \subseteq \mathcal{D}$ and selects one package $p_i \in \mathcal{P}_i$ per group. It is feasible when $R(p_i) \subseteq S$ for every group. Activating domain j incurs activation cost F_j.

The instance also has a finite set $\mathcal{H}$ of signed interaction factors. Each $a \in \mathcal{H}$ has a nonempty domain scope $P_a \subseteq \mathcal{D}$ and a table $\phi_a : \{0,1\}^{|P_a|} \to \mathbb{Z}$. Define

$$\Phi(S) = \sum_{a \in \mathcal{H}} \phi_a(\mathbb{1}_S|_{P_a})\,, \tag{30}$$

and let L be a fixed instance-specific constant. The term $-L+1$ is independent of the decision and does not change the minimizing decision. The objective of a feasible decision is

$$\min_{y} \left\{ \sum_{j \in S} F_j + \sum_{i=1}^{G} c_{p_i} + \Phi(S) - L + 1 \right\}. \tag{31}$$

Package choices are coupled because they can require the same activated domains, while the signed factors depend jointly on several domain statuses.

Exact algorithm and correctness. For $S \subseteq \mathcal{D}$, let

$$f_i(S) = \min_{\substack{p \in \mathcal{P}_i \\ R(p) \subseteq S}} c_p, \tag{32}$$

with $f_i(S) = +\infty$ when no package in group i is compatible with S. The construction algorithm evaluates every domain mask using

$$V(S) = \sum_{j \in S} F_j + \sum_{i=1}^{G} f_i(S) + \Phi(S) - L + 1. \tag{33}$$

For any fixed S, the package choices separate across groups, so the stored minimizer for each finite $f_i(S)$ yields a feasible decision whose objective is exactly $V(S)$. Conversely, every feasible decision with activated set S has package cost at least $\sum_i f_i(S)$. Minimizing $V(S)$ over all masks therefore gives the global optimum; the stored groupwise minimizers reconstruct the complete decision.

The construction algorithm uses the least-significant set bit to update the activation-cost and interaction contributions incrementally. It scans the package catalogs to obtain the groupwise minima. Define the per-mask work bound

$$I = n_D + G + \sum_{i=1}^{G} \sum_{p \in \mathcal{P}_i} \big(1 + |R(p)|\big) + \sum_{a \in \mathcal{H}} \big(1 + |P_a|\big). \tag{34}$$

The algorithm runs in $O(I 2^{n_D})$ time and uses $O(2^{n_D} + G)$ additional working memory.

D Deterministic Instance Generation

Section 3.2 describes instance generation using a size configuration, a structure configuration, and a deterministic generator. For each task family and workload level, we fix one size configuration and one structure configuration. These configurations, together with an instance index $j \in \{1, 2, 3\}$, determine the input to the family's human-written generator and thus one hidden instance $\mathcal{I}$. Repeating the construction with the same family, workload level, and instance index reproduces the same entities, relations, coefficients, and constraints.

Size and structure controls. The size configuration governs scale-bearing quantities such as the numbers of entities, choices, stages, or relations. The structure configuration governs the relations through which decisions interact in the constraints and objective. A single quantity can affect both problem scale and the pattern of dependencies among decisions, so the two roles need not correspond to separate generator arguments. Table 8 summarizes the principal quantities and coupling mechanisms for each family. The table is not a complete list of generator arguments.

Task family	Principal scale-bearing quantities	Structure producing global coupling
Transit Routing	Numbers of stages, candidate legs, and lines	Residue condition and directed transfer costs between consecutive lines
Basket Assembly	Item-group and ticket-catalog sizes	Ticket–item incidence and exact-cover overlap
Station Siting	Numbers of zones and demand relations	Shared site-count constraint and bounded-span pairwise demand terms
Authorization Planning	Numbers of stages, packages, and authorized transitions	Authorized transitions, handoffs, and the global clearance residue
Series Portfolio	Numbers of titles, lots, and factors	Overlapping bounded factor scopes
Machine Layout	Numbers of machines and slots	Permutation feasibility and pairwise flow terms
Sequential Matching	Participant and compatible-pair counts	Bijective matching and handoff costs between consecutive matches
Fleet Dispatch	Numbers of jobs and vehicles	Eligibility, competition for vehicle capacity, and shared activation costs
Evidence-Joined Routing	Layer, vertex, and edge counts	Joined edge attributes, path continuity, and the global residue condition
Migration Portfolio	Numbers of component groups, packages, and support domains	Shared domain activation and signed cross-domain interactions

Table 8: Scale-bearing quantities and global-coupling structures represented by the ten formal problem families. The entries summarize semantic roles in the formal instances rather than a complete list of generator arguments; the exact algorithms and their complexity parameters are specified in Appendices C and H.

Fixed membership and reproducibility. The family, workload level, and instance index are fixed before model evaluation. We validate the instance at each fixed family–level–index coordinate rather than selecting a replacement based on validation results or model outcomes. The released benchmark contains three fixed instances for each family–level combination, and all of them pass the required checks. The serialized hidden instances record this benchmark membership. The manuscript describes the construction contract, formal problems, and validation checks used for these instances.

E TOOL-MEDIATED WORLD CONSTRUCTION

Reusable family templates. World construction turns a generated hidden instance $\mathcal{I}$ into an environment that an agent can access through tools. For each family, a reusable template instantiates the textual question, maps fields of $\mathcal{I}$ to interface records, defines the tool schemas, assigns tool-call costs and an access budget, and specifies the final-decision format. Table 9 distinguishes the components shown to the agent from the hidden instance and private records used to produce tool responses.

Construction artifact	Template action	Paired-world relation and agent visibility
Hidden instance $\mathcal{I}$	Supplies the constraints, objective terms, and case-specific values	Shared by the paired worlds and not provided directly to the agent.
Textual question q	Instantiates the family task statement and the public identifiers needed to describe and submit a decision	Identical in the paired worlds and provided to the agent.
Private interface records	Maps the fields of $\mathcal{I}$ into the relational records queried by an interface's tools	Organized differently in the Direct and Mediated worlds; selected records appear only through executed tool responses.
Information tools $\mathcal{T}_{\text{info}}$	Defines the callable operations, argument schemas, and record queries	The agent receives the tool schemas and the responses to executed calls, but not the complete private record collection.
Cost function f_c and budget b	Assigns per-call costs and the total amount available for one trajectory	Identical in the paired worlds and provided to the agent.
Final-decision tool t_{dec}	Defines the family's structured decision format and records the submitted decision	Identical in the paired worlds; its schema is provided to the agent, and an executed call commits the final decision.

Table 9: Components used or instantiated by a family-specific world-construction template. Paired worlds share the decision problem and agent-facing task conditions while using different private relational organizations for the instance information.

Paired relational records. Applying the two interface mappings to the same $\mathcal{I}$ produces private Direct and Mediated record collections, denoted $R_{\mathcal{I}}^{(\text{D})}$ and $R_{\mathcal{I}}^{(\text{M})}$, respectively. These collections encode the same instance rather than adding new optimization data. A collection may introduce interface-specific identifiers or distribute one decision-relevant relation across several linked records. The information tools return only the records selected by the calls made along a trajectory. The paired worlds retain the same $\mathcal{I}$, q, f_c, b, and final-decision format. When a candidate-evaluation tool is included, its schema, cost, and checker behavior are also shared by the pair. The checks in Appendix F.3 confirm that both private record collections recover the same hidden instance, although their raw records and tool responses need not match.

Candidate evaluation. Some worlds additionally include a candidate-evaluation tool $t_{\text{eval}} \in \mathcal{T}_{\text{info}}$. It accepts one complete candidate decision and reports whether that decision is feasible and, when it is feasible, its objective value. The tool has positive cost, is nonterminal, and may be invoked at most once. It neither commits the candidate as the final decision nor directly returns fields of $\mathcal{I}$; only a later call to t_{dec} submits a binding final decision. This tool-specific one-call limit applies even though the other information tools may be invoked repeatedly within the trajectory limits.

Its feedback appears in the recorded trajectory and contributes to tool expenditure, but it does not add instance information to the trajectory diagnostics. The corresponding accounting appears in Appendices F.2 and G.

Released specifications. For each world, the instantiated specification fixes the textual question, tool schemas, mappings between calls and private records, tool-call costs, access budget, and final-decision format. The offline validation records also contain the two information-gathering call lists described in Section 3.3. Each list specifies the ordered information-tool calls and their arguments, and its total cost follows from f_c.

Appendix F.2 defines the request form, cost, response normalization, sufficiency criterion, and budget conditions for these lists. Appendix F.5 presents both lists and their costs for one released Fleet Dispatch world. The 480 instantiated lists are retained in the released validation records rather than printed in the manuscript, and they are not provided to the evaluated agents.

F Offline Validation

Before model evaluation, we validate the recorded optima, final-decision scoring, information access, and paired-interface construction described in Sections 3.2 and 3.3. We also verify the integrity of the released world bundles. Only worlds that pass all applicable checks are included in the benchmark.

Because the checks use different validation units, the summary at the end of the section reports each unit separately.

F.1 Instance and Scoring Checks

Why independent verification is required. The recurrences and correctness arguments in Appendix C establish that the construction algorithms are exact under the formal problem definitions. The recorded optimum, however, is produced by a concrete implementation of the state encoding, objective accounting, and decision reconstruction. We therefore solve every hidden instance again with an independently implemented exact algorithm that uses a different state direction or problem representation. Agreement between the two algorithms provides an independent check of the value and decision recorded during construction.

Independent optima. Table 10 summarizes how the independent verification algorithm differs from the construction algorithm described for each family in Appendix C. A unique optimum means that exactly one feasible structured decision attains the optimal objective value. A fixed instance passes this check only when the two implementations agree on that decision and value and both confirm uniqueness. All fixed instances pass.

Task family	Independent verification algorithm
Transit Routing	Represents the computation as exhaustive relaxation over the dense stage–line–residue graph rather than as the construction run's ordered predecessor scan.
Basket Assembly	Uses a forward-mask recurrence over the tripartite coverage state rather than the first-uncovered-item recurrence.
Station Siting	Uses a top-down recurrence with the generic at-most-H capacity semantics rather than the construction run's saturation reduction.
Authorization Planning	Uses a reverse suffix recurrence rather than the forward prefix recurrence.
Series Portfolio	Scans from right to left and closes each factor at its leftmost position rather than scanning left to right and closing it at its rightmost position.
Machine Layout	Builds the layout from the first slot by choosing the next machine rather than choosing the final machine of each prefix.
Sequential Matching	Processes suffixes in reverse and conditions on their first right participant rather than processing prefixes and retaining their last match.
Fleet Dispatch	Uses remaining capacities in a top-down recurrence rather than assigned loads in a forward recurrence.
Evidence-Joined Routing	Represents the computation as a general nonnegative shortest-path problem on the expanded state graph and runs Dijkstra's algorithm rather than the construction run's topological dynamic program.
Migration Portfolio	Represents consecutive domain masks in Gray-code order, updates the single changed domain, and independently recomputes compatible groupwise package minima.

Table 10: Independent verification algorithms used to validate the optima produced by the ten construction algorithms.

The work record comes only from the construction algorithm, as described in Appendix H; the independent verification algorithm serves only to validate its output. All 120 hidden instances pass the agreement and uniqueness checks.

Decision checking. For each of the 240 worlds, the family-specific decision checker determines whether a structured decision y satisfies the family's feasibility conditions. When it does, the checker recomputes the objective value directly from $\mathcal{I}$. As a consistency check, it must accept the independently verified optimal decision and reproduce the verified value $v_w^\star$.

Reference decisions. Let $y_{\mathrm{ref}}(w)$ denote the suboptimal reference decision used to set the reference-utility scale for world w. Before model evaluation, the checker accepts this decision as feasible and verifies

$$F_w(y_{\mathrm{ref}}(w)) = 1, \qquad \Delta_w^{\mathrm{ref}} = \Delta_w(y_{\mathrm{ref}}(w)) > 0. \tag{35}$$

Thus the denominator in Eq. (4) is positive. Because the objectives are integer-valued, exact optimality is checked by equality without a floating-point tolerance. All 240 worlds pass the optimal-decision

and reference-decision checks. The fixed scoring record for a world stores the verified optimal decision and value together with the validated reference decision and its recomputed gap.

F.2 Acquisition and Information-Separation Checks

From information tools to instance facts. The main text uses a *sufficient information set* to mean instance information that is enough to reconstruct $\mathcal{I}$ and determine its verified optimum. For formal comparison across interfaces, we represent this information as *normalized instance facts*: canonical facts about $\mathcal{I}$ obtained after combining and interpreting the relevant tool responses.

For a world w, let $\mathcal{T}_{\text{info}}(w)$ denote its information-tool suite. We call a tool *fact-revealing* when its responses expose normalized instance facts, and denote the set of these tools by $\mathcal{T}_{\text{fact}}(w) \subseteq \mathcal{T}_{\text{info}}(w)$. The candidate-evaluation tool defined in Appendix E belongs to $\mathcal{T}_{\text{info}}(w) \setminus \mathcal{T}_{\text{fact}}(w)$ because it returns feedback on a proposed decision rather than fields of $\mathcal{I}$. The final-decision tool t_{dec} is likewise not a fact-revealing tool.

Acquisition plans. Section 3.3 describes two alternative call lists. Each specifies the fact-revealing tools and their arguments. We formalize each such list as an *acquisition plan*. A fact-revealing request is a structured call $a = (t, x)$ to a fact-revealing tool with an admissible argument tuple. An acquisition plan for w is a finite ordered sequence $P = (a_1, \ldots, a_m)$ of these requests, with $a_i = (t_i, x_i)$. Its cost is

$$\text{cost}_w(P) = \sum_{i=1}^{m} f_c(t_i). \tag{36}$$

Neither the candidate-evaluation tool nor the final-decision tool is part of an acquisition plan. A plan is an offline validation object, not a trajectory or an acquisition strategy prescribed to an evaluated agent.

Normalizing combined responses. Raw Direct and Mediated responses may organize the same instance information through different relational records, and one fact may require records from more than one response. We therefore compare their interface-neutral content rather than their raw fields or tokens. When request a_i returns response o_i, a plan produces the call–response record sequence

$$((a_1, o_1), \ldots, (a_m, o_m)).$$

The world-specific response normalizer NF_w is defined on any finite ordered sequence of executed fact-revealing call–response records. It parses the returned records and performs the joins required across responses. It then removes interface-only identifiers and records, then deduplicates the resulting normalized instance facts. Applying it to the sequence produced by P gives

$$S_w(P) = \text{NF}_w\big((a_1, o_1), \ldots, (a_m, o_m)\big). \tag{37}$$

Sufficient information sets. A plan is sufficient when its nonempty set $S_w(P)$ reconstructs $\mathcal{I}$ and the family-specific construction algorithm applied to that reconstruction recovers the same independently verified optimum. Thus $S_w(P)$ is the formal instance-fact representation of a *sufficient information set* in Section 3.4.

Each world has two validated acquisition plans,

$$\mathcal{P}(w) = \{P_w^{(1)}, P_w^{(2)}\}, \qquad \mathcal{S}(w) = \{S_w(P) : P \in \mathcal{P}(w)\}. \tag{38}$$

The two plans may produce the same normalized fact set, so $|\mathcal{S}(w)|$ may be one or two. Both cases are covered by the trajectory diagnostics in Section 3.4. All 480 plans execute successfully and are sufficient. These sets need not be necessary or minimal, and the two plans do not enumerate every sufficient subset or prescribe an acquisition strategy to the evaluated agent.

Budget conditions. For every fact-revealing tool t, let $\mathcal{X}_w(t)$ be its declared finite set of admissible argument tuples. The complete catalog of requests that can return instance information, and the cost of issuing each distinct request once, are

$$\mathcal{C}_{\text{fact}}(w) = \{(t, x) : t \in \mathcal{T}_{\text{fact}}(w),\ x \in \mathcal{X}_w(t)\}, \qquad c_{\text{all}}(w) = \sum_{(t,x) \in \mathcal{C}_{\text{fact}}(w)} f_c(t). \tag{39}$$

Each distinct admissible request appears once; invalid calls, repeated copies, t_{eval}, and t_{dec} are excluded. Let

$$c_{\text{eval}}(w) = \begin{cases} f_c(t_{\text{eval}}), & t_{\text{eval}} \text{ is available in } w, \\ 0, & \text{otherwise.} \end{cases} \tag{40}$$

Validation requires

$$\begin{aligned} \text{cost}_w(P) + c_{\text{eval}}(w) &\le b && \forall P \in \mathcal{P}(w), \\ |P| + \mathbb{1}[t_{\text{eval}} \text{ is available in } w] &\le 255 && \forall P \in \mathcal{P}(w), \\ b &< c_{\text{all}}(w). \end{aligned} \tag{41}$$

The first two inequalities leave enough budget and calls to execute a validated plan and, where available, one candidate-evaluation call. The reserve $c_{\text{eval}}(w)$ is charged only if the agent invokes t_{eval}. The final inequality states that the budget does not permit issuing every request in the catalog. Together, the conditions show that a sufficient set of calls fits within both trajectory limits whereas the full catalog does not fit within the access budget. All 480 plans satisfy the first two conditions, and all 240 worlds satisfy the full-catalog condition.

Information-separation checks. These checks examine whether decision-relevant information is distributed across the tool interface rather than concentrated in one channel or response. We group normalized instance facts by the kind of information they describe and call each group an *information channel*. A channel is a validation grouping, not necessarily one tool, field, source record, or response. Every validated acquisition plan contains facts from multiple channels.

For each world–channel pair, we construct a counterfactual instance under the family's formal definition. Observations from all other channels remain fixed, while information in the tested channel changes so that the verified optimal decision differs. All 648 world–channel checks pass. Each tested channel can therefore affect the optimal decision under the corresponding counterfactual. This does not imply that every fact in the channel is necessary or that the channel is required to reconstruct every instance.

The response-level check constructs a counterfactual instance that holds one selected information-tool response fixed while modifying an unrevealed normalized instance fact so that the verified optimal decision changes. All 1,464 selected-response checks pass. None of the tested responses alone therefore determines the optimum. This finite result does not extend to every possible request, argument, or response. Neither the separation checks nor the access envelope establishes an information-theoretic lower bound.

Task family	Info. groups	Sufficient set size	Call count	Call cost	Budget / exhaustive	Algorithmic work
Transit Routing	3	1,755–6,963	59–147	226–634	524–884 / 637–1,558	50,737–47,619,285
Basket Assembly	2	262–640	38–180	172–764	260–780 / 1,020–2,554	35,646–39,098,995
Station Siting	2	284–326	172–234	558–774	577–779 / 915–1,293	55,391–34,318,974
Authorization Planning	3	2,633–7,781	17–73	99–291	120–296 / 729–1,257	43,747–35,313,268
Series Portfolio	2	381–1,458	24–141	196–924	1,228–1,924 / 1,491–5,658	54,587–34,402,415
Machine Layout	3	444–1,300	10–42	93–370	129–377 / 963–3,166	111,668–46,226,237
Sequential Matching	3	511–1,463	22–52	122–308	175–313 / 582–1,447	107,654–39,865,720
Fleet Dispatch	3	183–405	11–39	51–175	97–180 / 288–705	42,042–53,964,927
Evidence-Joined Routing	3	2,166–18,150	11–51	72–202	343–452 / 428–1,092	88,935–52,827,071
Migration Portfolio	3	707–1,829	11–151	252–1,542	277–1,547 / 2,142–11,344	61,393–46,070,264

Table 11: Information and access scale by task family. Sufficient-information-set size, call count, and call cost range over 48 world–call-list combinations (two validated call lists for each of 24 worlds); call cost excludes the candidate-feedback reserve. Budget and exhaustive-catalog cost are separate world-level ranges. Algorithmic work ranges over the 12 underlying hidden instances, whose paired worlds inherit the same value defined in Eq. (6).

Information and access scale. Table 11 reports information channels, sufficient-set sizes, call counts, and call costs alongside algorithmic work. For each family, the sufficient-set, call-count, and call-cost ranges cover the two validated acquisition plans for each of its 24 worlds. Set size counts normalized instance facts after the relevant responses have been combined. Call cost excludes the candidate-evaluation reserve included in Eq. (41).

Budget and full-catalog cost are separate world-level ranges over the 24 worlds. Algorithmic work is summarized over the 12 underlying hidden instances and shared by each Direct–Mediated pair. The columns use different scales and aggregation units. Their range endpoints need not describe the same world and cannot be combined to infer a within-world access guarantee.

F.3 Paired-Interface Equivalence

Recall from Appendix E that $R_{\mathcal{I}}^{(\mathrm{D})}$ and $R_{\mathcal{I}}^{(\mathrm{M})}$ are the complete private record collections used by a Direct–Mediated pair. Let $N^{(\mathrm{D})}$ and $N^{(\mathrm{M})}$ be the interface-specific mappings that reconstruct an instance from these complete record collections. Pair validation requires

$$N^{(\mathrm{D})}\left(R_{\mathcal{I}}^{(\mathrm{D})}\right) = \mathcal{I} = N^{(\mathrm{M})}\left(R_{\mathcal{I}}^{(\mathrm{M})}\right). \tag{42}$$

All 120 pairs pass this equality. The common hidden instance recovered by normalization defines the same feasible decisions and objective function, and both worlds inherit its verified optimum, uniqueness result, and algorithmic- work record. A separate structural check confirms that the interface-specific relational records differ before normalization.

The pair-level mappings $N^{(\mathrm{D})}$ and $N^{(\mathrm{M})}$ consume the complete private record collections. They are distinct from NF_w, which combines only the responses returned by the information-tool calls in a particular plan or trajectory. This distinction allows a pair to encode the same complete instance even when an agent observes only different subsets of its records.

As specified in Section 3.2, the paired worlds also retain the same textual question, tool-call costs, access budget, final-decision format, and, when included, candidate-evaluation tool. The equivalence check does not require identical raw responses or tool-use trajectories, nor does it imply identical model performance. Its conclusions apply only to the validated interface pairs.

F.4 Release Integrity

The release manifest binds each of the 240 world bundles to its serialized hidden instance, textual question, tool schemas, scoring record, validation records, and supporting documentation through content digests. A missing, altered, extra, or mismatched component causes the integrity check to fail. Only bundles that pass all applicable validation checks above are included in the released benchmark.

The verified optimum and uniqueness result, work record from the construction algorithm, validated acquisition plans, and sufficient information sets remain offline benchmark records and are not supplied to an evaluated agent. Model evaluation uses the fixed world and scoring records and does not rerun either exact algorithm. Evaluation-record integrity is discussed separately in Appendix I.

F.5 Worked Fleet Dispatch Validation

We now apply the preceding checks to the released pair introduced in Appendix B. Its tools and per-call costs are listed in Table 7.

Two validated ways of gathering sufficient information. For the Direct world, the two call lists described in the main text become the acquisition plans P_{row} and P_{window}. P_{row} uses one policy call, 14 job calls, five contract calls, and five route-row calls; P_{window} uses one policy call, five contract calls, and five window calls. The five window calls collectively cover all 14 jobs; their responses supply the loads and eligibility relation in addition to the route records. Their call counts and costs are

$$\begin{aligned} |P_{\mathrm{row}}| &= 25, \qquad \mathrm{cost}_{w^{(\mathrm{D})}}(P_{\mathrm{row}}) = 2 + 14(1) + 5(4) + 5(3) = 51, \\ |P_{\mathrm{window}}| &= 11, \quad \mathrm{cost}_{w^{(\mathrm{D})}}(P_{\mathrm{window}}) = 2 + 5(4) + 5(14) = 92. \end{aligned} \tag{43}$$

Normalizing the combined responses returned by either plan yields a sufficient information set that reconstructs $\mathcal{I}$ and recovers its verified optimum. Thus

$$\begin{aligned}\mathcal{P}(w^{(\mathrm{D})}) &= \{P_{\mathrm{row}}, P_{\mathrm{window}}\},\\ \mathcal{S}(w^{(\mathrm{D})}) &= \{S_{w^{(\mathrm{D})}}(P_{\mathrm{row}}), S_{w^{(\mathrm{D})}}(P_{\mathrm{window}})\}.\end{aligned} \tag{44}$$

The two normalized information sets need not be distinct. The corresponding Mediated plans have the same call compositions and costs. Their response records pass through the interface-specific joins illustrated above before the normalized instance facts are compared.

Budget arithmetic. The candidate-evaluation tool costs five units, so the two validated plans satisfy

$$51 + 5 \le 97, \qquad 92 + 5 = 97, \qquad 97 < 401, \tag{45}$$

where 401 is the cost of the full fact-request catalog. This total expands as $2 + 14(1) + 5(4) + 5(3) + (13 + 12)(14) = 401$: with 14 ordered jobs, there are 13 admissible length-two windows and 12 admissible length-three windows. The five-unit reserve is not part of either plan and is charged only if the agent invokes the candidate-evaluation tool. The plans and their sufficiency checks are offline validation objects, not trajectories prescribed to the agent.

Offline validation results. The construction and independent verification algorithms agree that the hidden instance has a unique optimum. Its objective value is 262,456,246,691, and the construction algorithm records algorithmic work of 45,799 operations. Because $w^{(\mathrm{D})}$ and $w^{(\mathrm{M})}$ share $\mathcal{I}$, both inherit this instance-level optimum and work record. A feasible submitted mapping is exactly optimal when its recomputed objective equals the verified value. The fixed reference decision and its gap remain part of the scoring record.

The verified optimal decision, uniqueness result, work record, validated plans, and sufficient information sets are offline records. They are not included in q, the tool schemas, or model-visible tool responses. If invoked, the candidate-evaluation tool reveals only the checker result for the mapping supplied by the agent; it does not disclose the verified optimum as a reference value.

F.6 Validation Summary

The checks above operate on several different units. Table 12 reports each unit separately.

Validated property	Validation unit	Count	Pass criterion
Fixed construction membership	Hidden instance	120	Each predetermined family–level–index coordinate yields its reproducible instance.
Optimum agreement and uniqueness	Hidden instance	120	The construction and independent verification algorithms agree on the optimal decision and value and confirm a unique optimum.
Decision checking	Algorithmic world	240	The checker accepts the verified optimal decision and reproduces its objective value.
Reference decision	Algorithmic world	240	The reference is feasible and has $\Delta_w^{\mathrm{ref}} > 0$.
Sufficient information acquisition	Acquisition plan	480	The responses reconstruct $\mathcal{I}$, recover its verified optimum, span multiple information channels, and fit within budget with the candidate-evaluation reserve and within the information-call cap.
Full-catalog budget	Algorithmic world	240	Issuing every request in the finite fact-revealing catalog exceeds b.
Channel counterfactual	World–channel pair	648	Other channel observations remain fixed while the verified optimal decision changes.
Response isolation	Selected response	1,464	The tested response remains fixed while an unrevealed instance fact and the verified optimal decision change.
Paired-interface equivalence	Direct–Mediated pair	120	Complete private records reconstruct the same $\mathcal{I}$ while retaining different relational structures.
Release integrity	World bundle	240	The released bundle matches the validated inventory, manifest entries, and content digests.

Table 12: Offline validation in ALGOWORLDS. Every listed unit passes its row's criterion. Counts use the validation unit shown in each row.

G METRIC ACCOUNTING AND SECONDARY DIAGNOSTICS

The three final-decision metrics and two trajectory diagnostics are defined in Section 3.4. This section explains how trajectory information is computed and how the five measures are aggregated. It then defines the secondary diagnostics used in the supplementary analyses.

Quantity	Per-evaluation value	Within-trial summary
Exact optimality	$E_w(y) \in \{0, 1\}$; one only for a feasible decision at the verified optimum.	Percentage with value one.
Feasibility	$F_w(y) \in \{0, 1\}$; one when the family-specific checker accepts the submitted decision.	Percentage with value one.
Reference utility	$U_w(y) \in [0, 100]$, defined in Eq. (4).	Arithmetic mean on its 0–100 scale.
Information sufficiency	$C_w(\tau) \in \{0, 1\}$; one when the information-tool responses contain at least one sufficient information set in full.	Percentage with value one.
Discovery coverage	$D_w(\tau) \in [0, 1]$; the largest fraction of any sufficient information set contained in those responses.	Arithmetic mean multiplied by 100.

Table 13: Per-evaluation values and within-trial aggregation for the three final-decision metrics and two trajectory diagnostics.

Missing final decisions. The three final-decision metrics in Table 13 follow the convention in Section 3.4. If no final decision is submitted, we write $y = \perp$ and set $F_w(\perp) = E_w(\perp) = U_w(\perp) = 0$. An evaluation record labeled as a protocol violation likewise has no scorable final decision. It retains a separate empirical outcome label but receives zero for all three metrics. Objective value $v_w(y)$, optimality gap $\Delta_w(y)$, and normalized regret, defined below, are available only for feasible submitted decisions. Trajectory diagnostics remain computable from information-tool responses returned before the evaluation ends.

Computing the trajectory information set. Let $\mathcal{R}_{\text{fact}}(\tau)$ be the ordered sequence of executed fact-revealing call–response records in τ. Applying the response normalizer from Appendix F.2 gives the trajectory information set used in Section 3.4:

$$\mathcal{A}_w(\tau) = \text{NF}_w(\mathcal{R}_{\text{fact}}(\tau)) \,. \tag{46}$$

The normalizer combines records across responses, performs the required joins, removes interface-specific content, and deduplicates the resulting instance facts. The diagnostics therefore compare instance information rather than raw text or record overlap. Candidate-evaluation calls, the final-decision tool, and calls rejected before execution do not enter $\mathcal{R}_{\text{fact}}(\tau)$ and do not contribute to $\mathcal{A}_w(\tau)$. If no fact-revealing call is executed, $\mathcal{A}_w(\tau) = \varnothing$ and both discovery coverage and information sufficiency are zero.

Aggregation. For a reported slice within trial r, let N_r be its number of evaluation records, and let E_e, F_e, U_e, C_e, and D_e denote the five per-evaluation values for record e. The trial-level summaries are

$$\begin{aligned} \overline{Z}_r &= \frac{100}{N_r} \sum_{e=1}^{N_r} Z_e, \qquad Z \in \{E, F, C\}, \\ \overline{D}_r &= \frac{100}{N_r} \sum_{e=1}^{N_r} D_e, \qquad \overline{U}_r = \frac{1}{N_r} \sum_{e=1}^{N_r} U_e. \end{aligned} \tag{47}$$

Each statistic is first computed within a trial. A full trial contains 240 evaluations. Within that trial, a task-family slice contains 24 evaluations, a workload-level slice contains 60, and a Direct or Mediated interface slice contains 120. Thus N_r is 240, 24, 60, or 120 for these summaries. The benchmark-wide results in Table 2 report the arithmetic mean and sample standard deviation of the three trial-level values. Supplementary tables likewise use trial-first aggregation. Each caption states the cross-trial summary shown; not every supplementary table reports a sample standard deviation.

Informative trajectories and conditional outcomes. Following Section 5.2, a trajectory is *informative* if and only if $C_w(\tau) = 1$. The conditional analysis includes evaluations whose trajectories satisfy this condition. Their final decisions are divided into three groups: globally optimal when $E_w(y) = 1$; feasible but suboptimal when $F_w(y) = 1$ and $E_w(y) = 0$; and other failure otherwise. The last group combines infeasible decisions, absent final decisions, and protocol violations.

Conditional exact optimality given information sufficiency is computed within a trial as

$$100 \frac{\sum_{e=1}^{N_r} C_e E_e}{\sum_{e=1}^{N_r} C_e}, \tag{48}$$

when the denominator is positive. The other conditional outcome percentages use the same within-trial denominator. Each percentage is computed within a trial and then averaged across trials. Information sufficiency concerns the information contained in the tool responses; it does not establish that the agent recognized or correctly used that information. These percentages describe final outcomes within the information-sufficient subset rather than a causal effect of information acquisition.

Normalized regret. For a feasible decision, the secondary decision-quality diagnostic *normalized regret* is

$$R_w(y) = \frac{\Delta_w(y)}{\Delta_w^{\mathrm{ref}}}.$$

It equals zero at the optimum and lies strictly between zero and one for a non-optimal decision better than the fixed reference. It equals one for any decision with the same gap as the reference and exceeds one for a feasible decision with a larger gap. Within each trial, median and 90th-percentile regret are computed among feasible non-exact decisions. Later tables report the arithmetic mean of the three trial-specific quantiles.

For any positively scaled and translated objective $v'_w(y) = \alpha v_w(y) + \beta$ with $\alpha > 0$, both the decision's optimality gap and Δ_w^{ref} are multiplied by α. Both normalized regret and reference utility are therefore unchanged.

Entity grounding. As a secondary trajectory-level diagnostic, entity grounding compares the identifiers in a submitted final decision with those previously returned to the model. Identifier normalization maps the decision and response fields to the canonical entity identifiers used by the world. Entity grounding is the fraction of normalized decision identifiers that also occur in model-visible tool responses before the final-decision call and is reported as a percentage. It measures identifier overlap rather than whether the responses contain a sufficient information set or whether the agent correctly uses the associated costs and constraints.

Executed calls and spend. Let $\mathcal{K}_{\mathrm{exec}}(\tau)$ index the executed calls in τ whose tools belong to $\mathcal{T}_{\mathrm{info}}(w)$. Executed information-tool calls and spend are

$$N_{\mathrm{call}}(\tau) = |\mathcal{K}_{\mathrm{exec}}(\tau)|, \qquad \mathrm{Spend}_w(\tau) = \sum_{i \in \mathcal{K}_{\mathrm{exec}}(\tau)} f_c(t_i), \qquad B_w(\tau) = \frac{\mathrm{Spend}_w(\tau)}{b}. \tag{49}$$

An executed candidate-evaluation call is included. The zero-cost final-decision tool is excluded. A call rejected before execution does not increment N_{call} and contributes no spend or normalized instance fact.

Within each trial, calls and spend are averaged across evaluation records. Spend relative to budget is the mean of the per-world percentages $100B_w(\tau)$, not a ratio formed after averaging spend and budgets. Conditional spend is averaged over evaluations satisfying the stated condition. The trial-level summaries are then averaged across the three trials. These quantities describe tool use and expenditure; they are not an efficiency score.

Termination and operational records. Two termination indicators record whether a trajectory reaches the information-call cap or the budget cap. Each indicator is attached to the finalized evaluation record and counted within a 240-world trial. Later tables report the mean of the three trial-level counts. These counts are neither percentages nor final-decision outcome categories.

H EXECUTED-WORK ACCOUNTING AND CALIBRATION

Section 4.2 defines $\mathrm{Work}(\mathcal{I})$ and the four shared workload bands. This section defines the operation counters, summarizes the parameters that govern each family-specific exact algorithm, and reports the operation counts of the released hidden instances.

Operation categories. Each hidden instance receives one work record from the execution of its construction algorithm. The independent verification algorithm and later validation checks do not contribute to this record. Table 14 defines the five terms in Eq. (6).

Counter	Executed operation represented
N_{state}	A recurrence, graph, frontier, subset, or mask state visited by the exact algorithm.
$N_{\mathrm{transition}}$	A predecessor, successor, assignment, or other candidate transition examined from a state.
N_{check}	A feasibility, capacity, compatibility, residue, or other constraint condition checked during the computation.
N_{update}	An update to a stored objective or dynamic-programming value.
$N_{\mathrm{reconstruct}}$	A component of the structured optimal decision recovered during reconstruction.

Table 14: The five operation categories summed by $\mathrm{Work}(\mathcal{I})$ in Eq. (6).

These counters measure executed operations rather than the size of the full candidate space or a worst-case bound. In Basket Assembly, N_{state} counts reached masks and $N_{\mathrm{transition}}$ counts examined ticket extensions rather than treating all 2^{3n} masks as visited. In Transit Routing, $N_{\mathrm{transition}}$ counts the predecessor candidates examined by the construction algorithm. Within a family, every hidden instance uses the same construction algorithm and counting rules. Across families, the five categories can occur in different proportions but receive equal unit weight.

Complexity parameters. The bounds in Appendix C identify the parameters that enlarge each exact algorithm's state space or transition set. Table 15 summarizes these parameters and guides the choice of family-specific size and structural settings. A generated instance is assigned to a workload band using its recorded operations, not the analytic bound.

Task family	Complexity parameters	Worst-case running time
Transit Routing	Stages d, legs per stage k, lines p, and residue modulus M	$O(p^2 \log p + dk^2 M)$
Basket Assembly	Items per group n and examined ticket extensions Δ	$O(2^{3n}\Delta)$
Station Siting	Zones n, counter range $\bar{H}+1$ for $\bar{H}=n-H$, and frontier width ω	$O(n(\bar{H}+1)2^{\omega}(\omega+1))$
Authorization Planning	Residue modulus M and authorization-graph vertex and edge counts V, E	$O(M(V+E))$
Series Portfolio	Titles n, factor span ω, and factors closing per position f_ω	$O(n2^{\omega}(1+f_\omega))$
Machine Layout	Machines n and machine-subset states	$O(n2^n)$
Sequential Matching	Participants n, used-right subsets, and final matches	$O(n^2 2^n)$
Fleet Dispatch	Jobs n, vehicles m, and mixed-radix load space $P=\prod_v(C_v+1)$	$O(nmP)$
Evidence-Joined Routing	Residue modulus M and graph counts n_V, n_E	$O(M(n_V+n_E))$
Migration Portfolio	Support domains n_D and per-mask work bound I	$O(I2^{n_D})$

Table 15: Complexity parameters and worst-case running times for the family-specific exact algorithms. Workload levels use recorded operations rather than these bounds.

Realized calibration. For each family, construction fixes one size and structural setting for each workload level. The operation count from the construction algorithm must fall within the correspond-

ing band in Section 4.2 and increase strictly across levels within the family. These settings and checks are fixed before model evaluation; model outcomes do not enter the calibration.

Task family	Level 1	Level 2	Level 3	Level 4
Transit Routing	50,737–54,053	1,132,766–1,201,734	8,922,755–9,191,138	45,715,806–47,619,285
Basket Assembly	35,646–56,392	773,416–1,558,838	6,430,549–7,063,570	32,683,874–39,098,995
Station Siting	55,391–55,424	1,196,297–1,214,030	5,829,214–6,245,492	34,289,268–34,318,974
Authorization Planning	43,747–44,176	994,240–999,048	7,036,179–7,045,627	35,238,569–35,313,268
Series Portfolio	54,587	973,647	6,698,087	34,402,415
Machine Layout	111,668–113,178	1,093,813–1,107,918	10,405,214–10,470,928	46,053,123–46,226,237
Sequential Matching	107,654–108,839	1,446,260–1,463,310	7,787,593–7,824,252	39,493,414–39,865,720
Fleet Dispatch	42,042–47,089	629,346–656,846	4,416,131–6,235,068	42,862,082–53,964,927
Evidence-Joined Routing	88,935–88,999	1,472,138–1,472,331	10,616,092–10,621,525	52,824,682–52,827,071
Migration Portfolio	61,393–64,262	945,495–1,095,815	8,509,678–9,384,131	40,832,977–46,070,264

Table 16: Executed algorithmic-work ranges over the three hidden instances at each family–workload-level setting. A singleton is shown when all three records coincide.

Each cell in Table 16 gives the range over the three hidden instances at one family–level setting. A singleton is shown when all three records coincide. The paired worlds for one hidden instance share its work record rather than contributing separate executions of the exact algorithm. Across the released instances, the displayed values range from 35,646 to 53,964,927 operations, a factor of about 1.5×10^3.

Interpretive scope. The work record sums five equally weighted operation categories from the family-specific construction algorithm. This record is reproducible for the released benchmark but depends on the algorithm and accounting convention. It does not measure wall-clock time, information-tool use, access cost, or an algorithm-independent lower bound on the work needed to solve an instance. A different exact algorithm or accounting convention could require a different calibration. The relationship between workload level and LLM performance is reported in Section 5.3 and Appendix J.3.

I Operational Protocol and Record Integrity

An evaluation is one execution of a fixed model configuration on one algorithmic world in one trial. Each model is evaluated in three trials over the same 240 worlds, comprising 120 hidden instances presented through the Direct and Mediated interfaces. This produces 720 scheduled evaluations per model and 5,040 across the seven models. These 5,040 evaluations reuse the same 240 worlds across models and trials. This section records the request settings, interaction controls, and evaluation-record accounting used in the experiments. Integrity of the benchmark world bundles is described separately in Appendix F.4.

The trajectory in Eq. (2) gives the interaction form when the agent submits a final decision. A finalized trajectory may instead end without invoking t_{dec}; the metric convention for that case is given in Appendix G.

Model	Model ID / API	Thinking / reasoning	Output cap	Temperature
GPT-5.6 Terra	`gpt-5.6-terra` *Responses API*	reasoning.effort=max	128K	Omitted
GPT-5.6 Sol	`gpt-5.6-sol` *Responses API*	reasoning.effort=max	128K	Omitted
Qwen 3.5 Plus	`qwen3.5-plus` *Chat Completions API*	thinking=true; budget=81,920	65,536	0.6
DeepSeek V4 Pro	`deepseek-v4-pro-202606` *Chat Completions API*	thinking=true; reasoning_effort=max	128K	Omitted
Claude Opus 4.8	`claude-opus-4-8` *Anthropic Messages API*	adaptive thinking; effort=max	128K	Omitted
Claude Sonnet 5	`claude-sonnet-5` *Anthropic Messages API*	adaptive thinking; effort=max	128K	Omitted
GLM 5.2	`glm-5.2` *Chat Completions API*	thinking=true; reasoning_effort=max	128K	1.0

Table 17: Model-specific request settings used in the reported experiments. The output cap is the returned-output token limit applied to each model request, not a trajectory-level token budget, and is distinct from any internal thinking budget. Here, K denotes 1,024 tokens, and Qwen's listed thinking budget is also measured in tokens. All models use a configuration with thinking or reasoning enabled. Temperature is marked *Omitted* when no explicit temperature value was supplied in the reported request configuration.

Fixed request configurations. All seven models use fixed configurations with the provider's reasoning feature enabled; some providers call this feature thinking. Before the full evaluation, we verify each configuration with a protocol-compatibility run that uses tools over multiple rounds and makes a valid structured final submission. This run confirms that the configuration can follow the interaction protocol; it does not require every subsequent evaluation to end with a valid decision.

Table 17 reports the exact model identifiers, API families, reasoning or thinking controls, output-token limits, and temperatures used in the experiments. Provider-specific thinking and reasoning controls are reported as configured. They are not treated as numerically equivalent compute budgets.

Shared interaction controls. In each evaluation, the agent receives the textual question q, the task-specific tool schemas $\mathcal{T}$, their costs under f_c, and the access budget b. It does not receive the hidden instance $\mathcal{I}$ or offline benchmark fields such as the verified optimum and sufficient information sets. Parallel tool calls are disabled, so each response is recorded before the next call is issued.

Each trajectory permits at most 255 executed information-tool calls in addition to its world-specific access budget. The information-call cap uses $N_{\mathrm{call}}(\tau)$ from Eq. (49). An executed candidate-evaluation call counts toward this cap; the final-decision tool and calls rejected before execution do not. A rejected call also incurs no benchmark cost. The zero-cost final-decision tool may be invoked at most once and ends the trajectory.

The agent-facing tool suite contains neither a general-purpose code execution environment nor an optimization solver. In particular, the exact algorithms used for benchmark construction and validation are not exposed to the evaluated agents.

Scheduled evaluations, attempts, and finalization. Each model configuration–world–trial combination defines one scheduled evaluation. The reported results contain exactly one finalized record for every scheduled evaluation. This record includes the trajectory and its metrics. If it contains no final-decision submission, we write $y = \perp$ and apply the metric convention in Section 3.4.

Provider and network attempts are individual request records linked to the scheduled evaluation. They are not trajectory outcomes and do not enter the final-decision outcome or cap-termination counts. Information-call-cap and budget-cap terminations are status indicators on the finalized record. Absent final decisions and protocol violations are instead final-decision outcome categories.

Manifest integrity. The benchmark manifest identifies the validated 240-world inventory through manifest entries and content digests, as described in Appendix F.4. The experiment manifest associates each finalized record with its model, trial, world, and protocol and links any provider or network

attempts. Before table generation, integrity checks verify these identifiers and content digests and require 240 finalized records for every model–trial pair. The empirical tables use only records that match these manifest entries and content digests.

J Supplementary Experimental Results

This section provides detailed outcome, workload, interface, and trajectory analyses that support the results in Section 5.

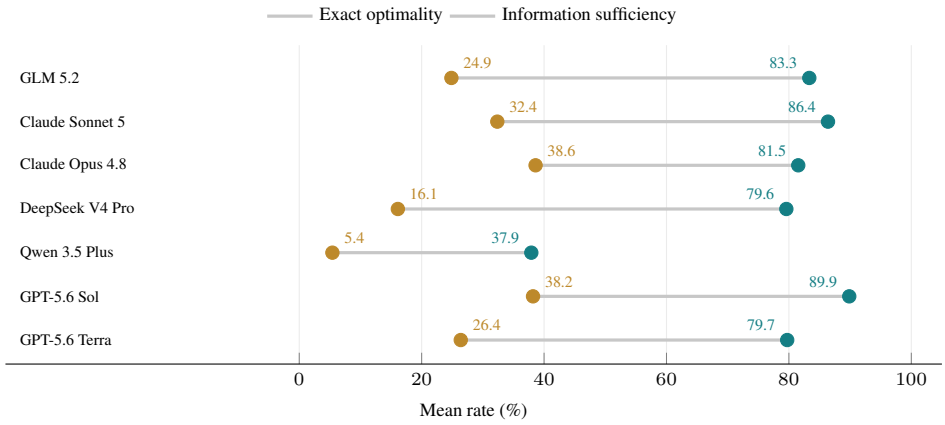


Figure 2: Mean exact-optimality and information-sufficiency rates over three trials on the same fixed worlds. Each line connects the two overall rates for one model. The figure does not condition final decisions on information sufficiency within the same evaluation. Across-trial standard deviations appear in Table 2, and repeated-trial profiles appear in Table 22.

The corresponding within-evaluation analysis appears in Section 5.2 and Table 3.

J.1 Final-Decision Outcomes and Quality

Model	Globally optimal	Feasible suboptimal	Infeasible	No decision	Protocol violation
GPT-5.6 Terra	63.3	147.3	29.3	0.0	0.0
GPT-5.6 Sol	91.7	142.3	6.0	0.0	0.0
Qwen 3.5 Plus	13.0	104.3	121.7	0.0	1.0
DeepSeek V4 Pro	38.7	162.7	33.7	5.0	0.0
Claude Opus 4.8	92.7	138.3	5.7	3.3	0.0
Claude Sonnet 5	77.7	143.3	17.7	1.3	0.0
GLM 5.2	59.7	95.3	15.0	58.7	11.3

Table 18: Mean counts by final-decision category per 240-world trial, averaged across the three trials. The five categories partition the evaluation records before rounding. Protocol violations remain separate from infeasible structured decisions and absent final decisions.

Model	Reference utility	Median regret	90th percentile regret	Globally optimal	Better than reference	Reference or worse	Infeasible, absent, or protocol violation
GPT-5.6 Terra	54.10 ± 1.95	0.526	1.582	63.3	91.3	56.0	29.3
GPT-5.6 Sol	74.98 ± 1.77	0.214	1.000	91.7	119.3	23.0	6.0
Qwen 3.5 Plus	15.82 ± 0.87	1.000	4.277	13.0	30.7	73.7	122.7
DeepSeek V4 Pro	28.49 ± 0.91	1.000	2.582	38.7	55.7	107.0	38.7
Claude Opus 4.8	73.14 ± 11.81	0.354	1.000	92.7	107.0	31.3	9.0
Claude Sonnet 5	67.20 ± 9.53	0.357	0.973	77.7	118.7	24.7	19.0
GLM 5.2	35.48 ± 2.67	1.000	1.655	59.7	41.0	54.3	85.0

Table 19: Reference-normalized decision quality. Utility is the mean ± sample standard deviation across three trial-level means; regret entries average the trial-specific quantiles among feasible non-exact decisions; and outcome columns are mean counts per 240-world trial. Better than reference means normalized regret strictly between zero and one, whereas reference or worse means normalized regret at least one. The four outcome columns partition each trial before rounding; the final column combines infeasible, absent, and protocol-violation outcomes.

Outcome categories. Table 18 partitions all evaluation records by final-decision outcome. Feasible but suboptimal decisions are the largest category for six of the seven models. Qwen is the exception, with more infeasible than feasible but suboptimal decisions. GLM has substantially more absent final decisions than the other models.

Quality among feasible suboptimal decisions. Table 19 reports reference utility and outcome counts over all evaluations. Its normalized-regret quantiles are instead computed among feasible non-exact decisions. The mean of the three trial-specific median regrets is below 1.0 for Sol (0.214), Opus (0.354), Sonnet (0.357), and Terra (0.526), and equals 1.0 for Qwen, DeepSeek, and GLM. A regret of 1.0 corresponds to the fixed reference gap. The mean trial-specific 90th percentile is at least 1.0 for six models; Sonnet's is 0.973, whereas Qwen's is 4.277. Because the table averages trial-specific quantiles, an entry of 1.0 does not mean that every feasible non-exact decision has the reference gap.

	Conditioning count	Final-decision outcomes				
Model	Information sufficiency	Globally optimal	Feasible suboptimal	Infeasible	No decision	Protocol violation
GPT-5.6 Terra	191.3	56.0	114.3	21.0	0.0	0.0
GPT-5.6 Sol	215.7	89.7	121.3	4.7	0.0	0.0
Qwen 3.5 Plus	91.0	3.0	45.3	42.3	0.0	0.3
DeepSeek V4 Pro	191.0	34.0	132.7	21.3	3.0	0.0
Claude Opus 4.8	195.7	71.7	116.3	4.3	3.3	0.0
Claude Sonnet 5	207.3	70.7	120.0	15.3	1.3	0.0
GLM 5.2	200.0	52.0	81.0	13.0	54.0	0.0

Table 20: Mean counts by final-decision category among evaluations satisfying information sufficiency. Each count is computed within a 240-world trial and then averaged across three trials, so the five decision columns sum to the information-sufficiency count in each row up to rounding.

Informative trajectories. As defined in Appendix G, an informative trajectory satisfies $C_w(\tau) = 1$. Table 20 partitions the corresponding evaluations into globally optimal, feasible but suboptimal, infeasible, absent-final-decision, and protocol-violation outcomes. The "Other failure" column in Table 3 combines the final three categories.

Within the informative subset, feasible but suboptimal decisions are the largest final-decision category for all seven models. Qwen averages 42.3 infeasible decisions per trial in this subset, whereas GLM

averages 54.0 absent final decisions. The detailed partition shows that these models reach the "Other failure" category in different ways.

J.2 Task-Level Reference Utility and Discovery Coverage

	Reference utility / discovery coverage						
Task family	Terra	Sol	Qwen	DeepSeek	Opus	Sonnet	GLM
Transit Routing	40.0/90.6	77.1/98.4	0.5/55.2	3.6/98.2	53.5/98.2	19.8/99.7	1.3/98.6
Basket Assembly	100.0/99.3	100.0/99.5	59.0/80.9	91.6/98.7	100.0/89.6	100.0/95.7	98.5/94.8
Station Siting	88.8/94.9	99.9/100.0	2.5/48.7	10.0/99.1	98.8/95.6	92.1/96.3	59.5/97.9
Authorization Planning	18.9/96.9	75.6/100.0	0.0/99.2	4.2/99.2	65.0/100.0	65.7/100.0	19.4/98.6
Series Portfolio	20.8/96.9	34.5/100.0	0.2/68.2	1.2/100.0	48.9/100.0	49.6/98.3	7.4/98.6
Machine Layout	33.1/94.1	56.1/95.4	0.0/90.4	0.6/92.9	40.0/99.7	38.3/96.3	7.1/95.8
Sequential Matching	92.7/99.6	95.7/99.8	15.9/88.0	66.0/90.4	95.6/100.0	98.2/100.0	47.0/60.4
Fleet Dispatch	99.3/100.0	98.6/100.0	80.1/99.0	86.9/99.6	99.9/100.0	84.2/99.7	91.5/95.8
Evidence-Joined Routing	24.9/94.1	45.2/97.9	0.0/86.6	15.4/99.9	43.2/90.3	53.4/99.9	1.2/98.6
Migration Portfolio	22.4/100.0	67.2/100.0	0.0/99.2	5.4/99.5	86.3/100.0	70.6/100.0	21.9/100.0

Table 21: Task-level reference utility and discovery coverage. Within each trial, each task-family slice contains 24 worlds. Each cell reports the mean across the three trial-level reference-utility means, followed by the mean across the three trial-level discovery-coverage percentages. Reference utility is on its $[0, 100]$ scale, whereas discovery coverage is a percentage. These are marginal task-level summaries of decision quality and the information contained in the recorded tool responses.

Table 21 shows that high discovery coverage can accompany very different levels of decision quality. In Authorization Planning, Qwen and DeepSeek both have 99.2% discovery coverage but reference utilities of 0.0 and 4.2, respectively. In Series Portfolio, DeepSeek has 100.0% discovery coverage and reference utility 1.2. Fleet Dispatch provides a contrasting profile: all seven models have discovery coverage of at least 95.8% and reference utility of at least 80.1.

These are marginal task-level summaries. The within-evaluation analysis based on information sufficiency appears in Section 5.2 and Table 3. The summaries do not identify which family properties cause the observed differences. The complexity bounds in Appendix C describe the offline exact algorithms, not the reasoning process of an evaluated agent.

J.3 Workload-Level Patterns and Scope

Table 5 reports exact optimality and reference utility by workload level. Each level contains 60 worlds within a trial: 30 hidden instances—three from each of ten families—presented through both paired interfaces. The three trials repeat these same fixed worlds rather than adding new hidden instances.

The main trend is consistent across models. All seven have lower exact optimality and reference utility at L4 than at L1. Reference utility decreases at every successive level for six models; DeepSeek V4 Pro is the exception, with a partial recovery from 22.3 at L3 to 28.1 at L4 while remaining below its L1 value of 35.9. Exact-optimality rates fluctuate more across intermediate levels, even though every model ends lower at L4 than at L1.

The levels are assigned from the work executed by the construction algorithms before model evaluation; model outcomes do not enter their calibration (Appendix H). Every model performs worse at L4 than at L1, indicating that this scale captures a dimension of challenge for the evaluated agents. It does not imply strict monotonicity for every model or adjacent level. The four levels contain different fixed instances rather than four versions of the same instance, so adjacent-level differences may reflect instance composition. With three trials per model, small reversals may also reflect run-to-run variation. The observed relationship is descriptive rather than a causal effect of workload.

J.4 Detailed Paired-Interface Accounting

This subsection gives the pairing and interval construction used for Table 6.

For a fixed model, hidden instance i, and trial r, let $E_{i,r}^{(\mathrm{D})}$ and $E_{i,r}^{(\mathrm{M})}$ be the binary exact-optimality outcomes under the Direct and Mediated interfaces, and define

$$d_{i,r} = E_{i,r}^{(\mathrm{D})} - E_{i,r}^{(\mathrm{M})}. \tag{50}$$

A Direct win has $d_{i,r} = 1$, a Mediated win has $d_{i,r} = -1$, and a tie has $d_{i,r} = 0$. Each model contributes $120 \times 3 = 360$ matched instance–trial pairs. A tie means only that the two evaluations share the same binary exact-optimality status: both may be exact or both non-exact. It does not imply identical submitted decisions, objective values, or tool trajectories. If W_{D} and W_{M} denote the Direct- and Mediated-win counts, respectively, the displayed rate contrast satisfies

$$\Delta_{\mathrm{D-M}} = \frac{100}{360} \sum_{i=1}^{120} \sum_{r=1}^{3} d_{i,r} = 100 \frac{W_{\mathrm{D}} - W_{\mathrm{M}}}{360}. \tag{51}$$

For interval construction, the three paired outcomes for an instance are first averaged as $\bar{d}_i = \frac{1}{3}\sum_{r=1}^{3} d_{i,r}$. The paired bootstrap resamples the 120 values $\{\bar{d}_i\}_{i=1}^{120}$ with replacement 10,000 times and recomputes their percentage-point mean. Thus the resampling unit is the hidden instance after trial averaging, not one of the 360 repeated instance–trial pairs. The displayed contrasts are computed from unrounded rates.

Across the 360 matched pairs for each model, Direct wins range from 11 to 34, Mediated wins from 6 to 13, and ties from 315 to 343. A positive aggregate contrast therefore does not imply pointwise Direct dominance. Only GPT-5.6 Sol and Claude Sonnet 5 have 95% intervals that exclude zero. The comparison is limited to the 120 validated Direct–Mediated pairs in this benchmark and does not establish a Direct advantage for other interface designs. Structural equivalence checks for these pairs appear in Appendix F.3.

J.5 Repeated-Trial Profiles

	Trial exact-optimality rate			**Trials attaining exact optimality per world**				**Pairwise agreement**
Model	T1	T2	T3	0/3	1/3	2/3	3/3	
GPT-5.6 Terra	27.9	25.4	25.8	164	8	22	46	91.7
GPT-5.6 Sol	39.2	38.8	36.7	135	16	8	81	93.3
Qwen 3.5 Plus	5.8	5.4	5.0	218	9	9	4	95.0
DeepSeek V4 Pro	17.1	16.2	15.0	187	14	15	24	91.9
Claude Opus 4.8	33.3	40.4	42.1	127	18	25	70	88.1
Claude Sonnet 5	27.9	34.6	34.6	141	22	20	57	88.3
GLM 5.2	24.2	25.0	25.4	166	8	27	39	90.3

Table 22: Repeated-execution profiles on the same fixed worlds in AlgoWorlds. T1–T3 are exact-optimality rates in percent. Columns 0/3 through 3/3 count worlds on which the model attains exact optimality in that many trials. For each pair of trials, pairwise agreement is the percentage of the 240 fixed worlds whose binary exact-optimality statuses agree; the table reports the mean over the three trial pairs. The four frequency counts sum to 240 worlds, and together with pairwise agreement describe binary exact-optimality repeatability on this fixed suite, not agreement in objective value or submitted decision.

Two summaries describe variation across the three trials. First, the trial-level exact-optimality rates in Table 2 have sample standard deviations of 0.42–1.34 percentage points for GPT-5.6 Terra, GPT-5.6 Sol, Qwen 3.5 Plus, DeepSeek V4 Pro, and GLM 5.2. The corresponding values are 4.65 and 3.85 points for Claude Opus 4.8 and Claude Sonnet 5. Second, Table 22 holds the model and algorithmic

world fixed and compares binary exact-optimality status across trials. Mean pairwise agreement ranges from 88.1% to 95.0% across models.

Qwen has the highest pairwise agreement, at 95.0%, yet it never attains exact optimality on 218 of the 240 worlds and does so in all three trials on only four. GPT-5.6 Sol has 93.3% pairwise agreement and attains exact optimality in all three trials on 81 worlds. Pairwise agreement therefore measures repeatability of the binary exact-optimality status; it is neither an exact-optimality rate nor a measure of agreement in submitted decisions or objective values.

J.6 Additional Tool-Use and Termination Diagnostics

	Trajectory-level summaries			Tool use	Conditional and termination summaries		
Model	Discovery	Grounding	Sufficiency	Calls	Exact given sufficiency	Info.-call cap	Budget cap
GPT-5.6 Terra	96.65	100.00	79.72	54.9	29.24	0.0	0.0
GPT-5.6 Sol	99.10	100.00	89.86	57.4	41.57	0.0	0.0
Qwen 3.5 Plus	81.55	97.91	37.92	34.6	3.31	0.0	0.0
DeepSeek V4 Pro	97.74	97.27	79.58	69.0	17.82	2.0	0.7
Claude Opus 4.8	97.33	98.60	81.53	58.2	36.33	3.3	0.0
Claude Sonnet 5	98.59	99.44	86.39	59.9	33.92	1.3	0.0
GLM 5.2	93.91	70.82	83.33	56.8	25.97	57.3	1.3

Table 23: Trajectory-level, tool-use, conditional, and termination summaries. Discovery coverage and information sufficiency are the two trajectory diagnostics defined in Section 3.4; entity grounding is a secondary diagnostic defined in Appendix G. Each numerical entry is the mean of the three trial-level summaries. These three quantities and conditional exact optimality are percentages. Calls are mean executed information-tool calls per evaluation; information-call- and budget-cap terminations are mean counts per 240-world trial.

Entity grounding. Six models have entity grounding between 97.27% and 100.00%. GLM differs from this pattern: its mean discovery coverage is 93.91%, while its mean entity grounding is 70.82%. Discovery coverage measures overlap with sufficient information sets, whereas entity grounding measures identifier overlap with earlier tool responses. Neither quantity shows whether the agent correctly used the returned costs and constraints.

Cap terminations. GLM reaches the information-call cap in a mean of 57.3 evaluations per 240-world trial. Mean information-call-cap counts are also nonzero for DeepSeek, Claude Opus 4.8, and Claude Sonnet 5, at 2.0, 3.3, and 1.3, respectively. Mean budget-cap counts are nonzero only for DeepSeek and GLM, at 0.7 and 1.3 per trial. These termination indicators are separate from final-decision categories and from the provider and network attempts described in Appendix I.

	Overall acquisition			Conditional spend	
Model	**Calls**	**Spend**	**Spend / budget**	**Given information sufficiency**	**Given exact optimality**
GPT-5.6 Terra	54.9	335.4	68.1%	360.1	367.3
GPT-5.6 Sol	57.4	448.2	80.6%	464.0	323.8
Qwen 3.5 Plus	34.6	291.9	66.5%	273.4	311.7
DeepSeek V4 Pro	69.0	424.5	83.0%	455.1	335.4
Claude Opus 4.8	58.2	367.6	73.3%	353.7	328.9
Claude Sonnet 5	59.9	369.8	74.8%	373.3	357.7
GLM 5.2	56.8	357.2	67.6%	390.4	325.4

Table 24: Descriptive acquisition accounting. Calls and spend follow Eq. (49): they include candidate-evaluation calls when executed and exclude the final-decision tool and calls rejected before execution. Calls, spend, and spend relative to budget are computed per evaluation, averaged within each trial, and then averaged across three trials. Spend and both conditional-spend columns are in benchmark cost units; spend/budget is instead the mean of the per-world percentages. Conditional columns are likewise computed within each trial for evaluations satisfying information sufficiency or exact optimality and then averaged across the three trials; they are not efficiency scores or estimates of a spending effect.

Executed calls and spend. Table 24 reports executed calls and expenditure in benchmark cost units. It keeps absolute spend separate from the mean per-world spend-to-budget percentage. Qwen has the lowest mean call count, absolute spend, and budget percentage, at 34.6, 291.9, and 66.5%, respectively. Sol has the highest absolute spend at 448.2, whereas DeepSeek has the highest mean per-world budget percentage at 83.0%.

Conditional spend follows different patterns across models. For Sol, the overall mean, the mean among information-sufficient evaluations, and the mean among exact-optimal evaluations are 448.2, 464.0, and 323.8, respectively. For Qwen, the corresponding values are 291.9, 273.4, and 311.7. These conditional summaries describe the selected subsets; they are not efficiency scores or estimates of a spending effect.

J.7 Structured Trajectory Examples

Field	Recorded trajectory
Archetype	**Tool responses do not contain a sufficient information set in full**
World	Station Siting, Mediated, L2, instance 1; DeepSeek V4 Pro, trial 2
Recorded calls	read zone window (8); read zone (3); read demand (11)
Final result	No final decision; discovery coverage 35.0%.
Archetype	**Sufficient information, infeasible**
World	Evidence-Joined Routing, Mediated, L1, instance 2; GPT-5.6 Terra, trial 1
Recorded calls	read tariff card (1); read manifest window (4); read exceptions window (4); read manifest stage (1); read exceptions stage (1); submit path (1)
Final result	Final decision infeasible; discovery coverage and entity grounding both 100.0%.
Archetype	**Sufficient information, feasible suboptimal**
World	Series Portfolio, Direct, L2, instance 3; Claude Sonnet 5, trial 1
Recorded calls	read catalog window (22); read series page (20); submit portfolio (1)
Final result	Feasible non-optimum; normalized regret 0.507, reference utility 49.3.

Table 25: Three examples drawn from recorded trajectories. The recorded-call rows give a compact inventory of tool names and counts in first-use order, include the final-decision submission when one occurs, and exclude private reasoning text. They do not reproduce the full call sequence and are not the N_{call} diagnostic, which excludes t_{dec}. The examples illustrate distinct information-availability and final-decision outcomes; corresponding aggregate results appear in Tables 2, 3, 19 and 23. They are illustrative examples rather than estimates of outcome prevalence.

The three examples illustrate the gap between information availability and final-decision quality examined in Section 5.2 and Table 3. In the first example, the returned information covers 35% of the sufficient information set with which it has the greatest overlap. Hence $D_w(\tau) = 0.35$ and $C_w(\tau) = 0$, and the trajectory ends with $y = \perp$. This statement is relative to $\mathcal{S}(w)$: as established in Appendix F.2, $C_w(\tau) = 0$ does not rule out a different way of determining the optimum. In the infeasible example, $C_w(\tau) = 1$ and entity grounding is 100%, yet the submitted decision has $F_w(y) = E_w(y) = 0$. In the feasible but suboptimal example, $C_w(\tau) = F_w(y) = 1$ and $E_w(y) = 0$; its normalized regret is 0.507 and its reference utility is 49.3. The examples show why discovery coverage, entity grounding, feasibility, and objective quality are reported separately. They illustrate selected information and final-decision gaps rather than all outcome categories or their prevalence.