DATO: Difficulty-Aware Tool Orchestration for Clinical Pharmacogenomic Reasoning with a Compact Orchestrator

Anonymous ACL submission

Abstract

Pharmacogenomics (PGx) promises personalized medicine, but clinical adoption is hampered by manpower shortages and the complexity of interpreting evolving guidelines. We introduce Difficulty-Aware Tool Orchestration (DATO), to our knowledge the first PGx-focused adaptive agentic framework with a small open-source model as a central tool orchestrator. For each query, the orchestrator adaptively selects exactly one of three predefined tool pipelines according to question complexity. To overcome the limited ability of small models to assess PGx difficulty from raw text, we introduce an input preprocessing step that combines a pharmacogenomic knowledge graph (PGx-KG) with LLM-based extraction to produce structured input (question facets, PGx metadata, and graph-complexity features). The orchestrator is trained with Group Relative Policy Optimization (GRPO) under a composite reward over response quality and inference cost, together with small-actor adaptations for stable learning under limited data. On a 36-question test set spanning hard, intermediate, and common difficulty categories, DATO improves response quality by 28.0% over a non-adaptive baseline that always invokes the full retrieval-and-reasoning pipeline, while simultaneously reducing token usage by 18.4%.

1 Introduction

Pharmacogenomics (PGx) tailors drug selection and dosage to a patient's genetic profile, yet clinical adoption remains limited. Manpower shortages and the complexity of rapidly evolving CPIC (Relling et al., 2020) and FDA guidelines force clinicians to consult gene-type reports, multi-page PDF tables, and API data in parallel for each patient. This process is error-prone and time-consuming, and contributes to the limited routine adoption of PGx testing in most hospitals. Automated assistance is therefore needed to support open-ended PGx guidance in real-world clinical settings.

Existing approaches fall short on two fronts. Recent PGx retrieval-augmented generation (RAG) works such as the CPIC-based assistant of Murugan et al. (2024), PGxRAG (Borishetty et al., 2025), Sherpa Rx (Rector et al., 2025), and the Hierarchical RAG framework of Jeon et al. (2026) are predominantly single-LLM pipelines over static knowledge bases, and lack agentic orchestration or adaptive tool routing. General-purpose agent benchmarks such as ToolLLM (Qin et al., 2023) and AgentBench (Liu et al., 2024), in turn, emphasize breadth across heterogeneous tasks with success verifiable through exact-match answers. PGx queries differ on both axes. They are confined to a single clinical domain but require open-ended free-text responses across diverse question types (e.g., dosage adjustment, drug-drug interaction, and risk assessment) with varying difficulty levels. They also demand sophisticated retrieval and integration of both structured (text, APIs, JSON) and unstructured (figures and tables) data, which single LLMs cannot effectively handle. In contrast, multi-agent orchestration systems excel at stably retrieving and integrating such heterogeneous sources, making consistent performance within one domain far more important than cross-domain generalization.

The PGxQA benchmark (Keat et al., 2025) supplies stakeholder-specific evaluation for PGx question answering, but targets single-LLM chatbots rather than adaptive tool-orchestration systems or their cost-quality trade-offs. Recent medical-AI benchmarks further suggest that multi-agent or complex pipelines do not uniformly outperform simpler alternatives (Zhu et al., 2025), reinforcing the need for task-specific tool routing.

We introduce Difficulty-Aware Tool Orchestration (DATO), an adaptive agentic framework for PGx with a small open-source model as the central orchestrator. Built on Qwen3-8B (Yang et al., 2025), DATO adopts the small-orchestrator-plus-GRPO paradigm of ToolOrchestra (Su et al., 2025),

where GRPO (Shao et al., 2024) is one of the reinforcement learning algorithms used for training, but reframes orchestration at the pipeline level. For each query, the orchestrator selects exactly one of three predefined pipelines: API-only, Reasoning-only, and Full. The orchestrator is trained under a composite reward that captures both answer quality and inference cost within a single specialized domain.

A central design choice addresses a known limitation of small models, which is that they cannot reliably assess PGx query difficulty from raw text alone. We introduce a preprocessing step that combines a pharmacogenomic knowledge graph (PGx-KG) (Faruk, 2025) with LLM-based extraction to supply the orchestrator with structured input consisting of question facets, PGx metadata, and graph-complexity features. Prior PGx-KG work, such as the drug-safety agent of Mathialagan et al. (2026), has focused on retrieval and post-hoc evidence verification. To our knowledge, ours is the first work to use such a graph to structure input for an adaptive agentic framework in PGx. We further introduce small-orchestrator adaptations to the ToolOrchestra training paradigm, including a diversity hint, batch-level advantage normalization, and a clinician-aligned criteria-based reward.

This work makes the following contributions:

- **System.** The first PGx-focused tool-orchestration framework with an 8B orchestrator, showing that a small open-source model can effectively route PGx queries to appropriate tools, yielding better quality clinical PGx responses at lower compute cost.

- **Methodology.** The first work to leverage PGx-KG-augmented structured input to enable effective tool orchestration with a small orchestrator.

- **Empirical.** On a 36-question test set, DATO improves overall response quality by 28.0% over the default always-full-pipeline configuration with 18.4% fewer tokens, demonstrating that small open-source orchestrators can adaptively route reasoning effort to the queries that need it, rather than uniformly invoking the most expensive pipeline.

2 Related Work

2.1 LLM Applications in Pharmacogenomics

Recent years have seen growing interest in applying large language models (LLMs) to pharmacogenomics (PGx). Several retrieval-augmented generation (RAG) systems have been proposed to assist clinicians with gene-drug interpretation, including PGxRAG (Borishetty et al., 2025), Sherpa Rx (Rector et al., 2025), the Hierarchical RAG framework of Jeon et al. (2026), and the statin-focused PGx assistant of Murugan et al. (2024). These works primarily rely on single-LLM pipelines with static knowledge bases, retrieving guideline fragments or drug labels before generation. While they show promising accuracy on narrow tasks, most are limited to textual inputs and lack adaptive tool selection or multi-agent collaboration. Complementing these systems, the PGxQA benchmark (Keat et al., 2025) provides a stakeholder-aware evaluation suite for PGx question answering, but it does not target retrieval or orchestration design, and the underlying scarcity of high-quality labeled PGx data, which still requires extensive manual review by clinical pharmacists and genetic counselors, continues to constrain large-scale evaluation. Recent work such as CoVaPh (Lu et al., 2026) explores multi-agent PGx reasoning with tool augmentation, but relies on large proprietary models and a static workflow rather than a learned cost-aware orchestration policy, and does not address difficulty-aware pipeline selection.

2.2 Knowledge Graph Applications in Pharmacogenomics

Pharmacogenomic knowledge graphs (PGx-KGs) have become a key structured resource for organizing drug-gene-variant-disease relationships. The most comprehensive recent example is the large-scale PGx-KG introduced by Faruk (2025), which integrates PharmGKB, ClinVar, SIDER, and Reactome. Prior work on knowledge graphs in this and adjacent areas has focused primarily on retrieval enhancement, link prediction, and post-hoc evidence verification; for example, Mathialagan et al. (2026) integrate a drug-safety knowledge graph with hallucination-aware query routing for agentic pharmacovigilance, prioritizing deterministic graph retrieval over generative reasoning. However, no existing work has explored feeding structured graph-derived feature directly as input to a tool orchestrator or agent system.

2.3 Clinical and Medical Domain Multi-Agent Systems

In the broader clinical and medical domain, multi-agent systems have begun to emerge for complex reasoning tasks. AgentClinic (Schmidgall et al., 2026) is a multimodal benchmark for tool-using clinical AI agents that simulates realistic patient interactions across nine medical specialties. ClinicalAgents (Ge et al., 2026) proposes a multi-agent orchestration framework that mimics expert clinicians' cognitive workflow with dual-memory mechanisms. MedAgent-Pro (Wang et al., 2025) further benchmarks multi-agent collaboration for medical QA, diagnosis, and evidence-based reasoning. While these systems demonstrate strong performance on medical tasks, they typically rely on large proprietary models and static workflows, are rarely tailored to pharmacogenomics, and seldom operate under the strict computational constraints needed for deployment in resource-limited hospital environments.

2.4 Tool Orchestration and Agentic Frameworks

Tool orchestration has emerged as a central research direction in LLM agent systems. Foundational works include ReAct (Yao et al., 2023), which interleaves reasoning and acting; Toolformer (Schick et al., 2023), which enables language models to teach themselves tool use through self-supervision; and AutoGen (Wu et al., 2023), a multi-agent conversation framework. More recently, ToolOrchestra (Su et al., 2025) introduced an efficient reinforcement-learning approach using Group Relative Policy Optimization (GRPO) (Shao et al., 2024) to train a small model as a central tool orchestrator, achieving strong results on general-purpose benchmarks such as HLE, FRAMES, and GPQA. These general frameworks are designed for broad, cross-domain scenarios and are not optimized for a single specialized domain such as PGx. In our setting, the tool set is fixed and the priority is consistent performance across varying question types and difficulty levels within a single domain.

3 Methodology

3.1 Overall Architecture

The Tool Orchestrator (actor) is the core decision-making component, built on the open-source Qwen3-8B model. The orchestrator receives input for every user query and selects exactly one of three tool pipelines. The selected pipeline then generates the final response. The three tool pipelines are illustrated in Figure 1:

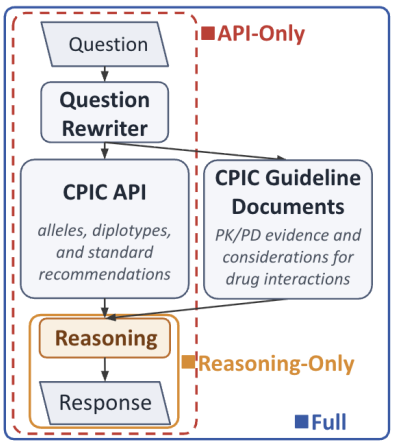

Figure 1: Three tool pipelines.

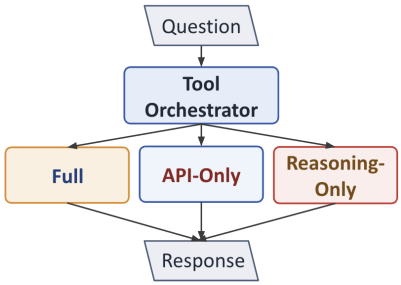

Figure 2: Tool-Orchestrator-Based DATO.

- **API-only:** extracts data from the CPIC API and then performs reasoning over the retrieved content.
- **Reasoning-only:** skips both API calls and RAG retrieval, generating an answer directly from the LLM's parametric knowledge.
- **Full:** calls the CPIC API and retrieves the complete CPIC guideline via hybrid multi-modal RAG, then performs reasoning over both sources.

Figure 2 presents the DATO framework after the addition of the Tool Orchestrator. Appendix A.1 contains the full prompt used by the Tool Orchestrator.

Precise pipeline selection requires the actor to accurately assess each query, particularly its question facets and difficulty level. Small models have clear limitations in directly assessing PGx query difficulty, which is a critical step for appropriate

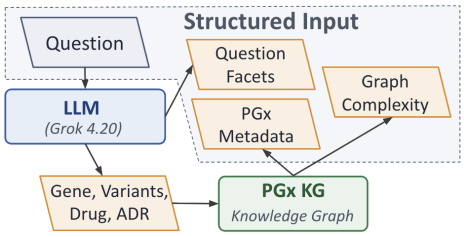

Figure 3: Structured input generation.

pipeline selection. To overcome this, we introduce a dedicated preprocessing step before the actor makes any decision.

3.2 Question Preprocessing with the PGx Knowledge Graph

We add a question-preprocessing step before the orchestrator that combines the pharmacogenomic knowledge graph (PGx-KG, illustrated in Appendix B.1) with LLM-based processing. As shown in Figure 3, an LLM first extracts question facets, namely gene, variant, drug, and adverse drug reaction (ADR), from the query (prompts in Appendices A.2 and A.3). The extracted gene, variant, drug, and ADR entities are then fed into the PGx-KG as nodes to obtain graph-validated PGx metadata and to compute graph-complexity features from the query-formed induced subgraph. Question facets themselves bypass the KG, as they capture query intent rather than entity-level information. The resulting three structured pieces, namely question facets, PGx metadata, and graph complexity (details in Appendix B.2), together with the original query, are fed directly to the Tool Orchestrator as input. Those elements together are called structured input.

An example of structured input is provided in Appendix B.3. We also experimented with additional graph-complexity metrics in Appendix B.4.

3.3 Reinforcement Learning Process

We train the tool orchestrator (actor) with Group Relative Policy Optimization (GRPO) (Shao et al., 2024), a reinforcement-learning algorithm extending Proximal Policy Optimization (PPO) (Schulman et al., 2017). GRPO estimates advantages by comparing responses within a group generated for the same query. This eliminates the critic network, reducing memory and compute overhead.

For each query, the actor generates a group of candidate responses. Each response is scored by one or multiple reward functions. The algorithm then computes each response's relative advantage against others in the group and updates the policy via a clipped objective. GRPO offers two advantages over PPO in our setting. First, removing the critic eliminates the need for expert-labeled critic fine-tuning data. Second, group-relative advantage estimation produces a lower-variance signal than PPO's absolute advantages, yielding more stable convergence when rewards are noisy or sparse.

3.4 Tailored Reward Evaluation for Tool Pipeline Selection

The reward function shapes the GRPO policy by assigning higher scores to better tool-pipeline selections for each PGx query. Low-variance and directionally consistent rewards are critical. They enable reliable relative-advantage estimation within each group and support stable policy updates.

We design a tailored evaluator to score the answers produced by the three tool pipelines on open-ended PGx questions. A criteria-based LLM evaluator, backed by Grok 4.3, follows a domain-expert-validated prompt (Appendix A.4) to rate each response on four clinical criteria: accuracy, relevance, completeness, and clarity. The final reward is called the response quality score. It is a weighted average with $(w_{\text{accuracy}}, w_{\text{relevance}}, w_{\text{completeness}}, w_{\text{clarity}}) = (0.333, 0.167, 0.333, 0.167)$, placing greater emphasis on accuracy and completeness. The resulting scores are interpretable to clinicians and let us sanity-check whether reward trends are reasonable.

Single-question scoring exhibited high variance during development. LLM evaluators exhibit high variance when assigning scores on a fixed numerical scale. To stabilize the evaluation signal, we adopt a group-based strategy. Each target question is evaluated together with two randomly selected PGx questions from the same difficulty category, forming a batch of three. The procedure is repeated five times, and the mean is taken as the final reward. This multi-example, multi-run procedure reduces variance while preserving the relative ranking among pipelines. Per-category statistics (mean, standard deviation, min, max) for the three pipelines are reported in Appendix D.1. Two observations stand out:

- The standard deviation, minimum, and maximum values confirm that averaging effectively stabilizes the scores.
- For intermediate questions, the full pipeline

scored significantly higher than both API-only and Reasoning-only, while the gap between the latter two was small. We examine the implications of this score distribution for reward design and orchestrator behavior in Sections 5.1 and 5.3.

4 Experimental Setup

4.1 Data Sets Construction and Training Protocol

PGx queries form a single-domain task in English spanning varying difficulty levels. We construct a test set of 36 questions across three difficulty categories (hard, intermediate, and common), with 12 questions per category. The hard questions were annotated and validated by three academic researchers, all licensed pharmacists with prior experience interpreting CPIC guidelines; the annotation protocol, compensation, and ethics details are provided in Appendix F. The intermediate questions were generated by reverse-engineering CPIC API outputs with an LLM and then format-validated by the same group of experts. They address the same topical scope as the hard questions but are designed such that the CPIC API alone suffices to answer them. The common questions were LLM-synthesized to simulate everyday medication inquiries from the general public and fall outside CPIC's scope. The hard category is capped at 12 questions by the throughput of expert validation, with the other two categories matched in size to maintain balance.

Because the 36-question set is small and shared between training and evaluation, we adopt a 2-fold cross-validation protocol that splits the data into two equal folds of 18 questions, each containing a balanced mix of the three difficulty categories. Each fold is used alternately as train and test set so all 36 questions serve as test set. To further support stable training of a small 8B actor on this limited dataset, we apply two adaptations to the standard GRPO pipeline: full details of each adaptation are provided in Appendix C.

4.2 Balancing Quality and Cost

Tool orchestration does not select pipelines based on response quality alone; it must also account for computational cost (Su et al., 2025). We measure cost as token consumption and total inference time. The strategies the actor learns under different quality-cost trade-offs are therefore a key object of analysis.

We simulate these trade-offs by generating multiple weighted reward variants. The reward begins by independently normalizing the response quality score, token usage, and total runtime, then computing a weighted sum. We evaluate two weight configurations of $(w_{\text{quality}}, w_{\text{token}}, w_{\text{runtime}})$:

- **WS0** $(1, 0, 0)$: full emphasis on quality.
- **WS02** $(1, 0.02, 0.02)$: slight cost incorporation while prioritizing quality. The 0.02 weights were derived from the average observed differences among the three normalized metrics across question difficulty categories.

Ideally, the optimal weights should lead the actor to select the full pipeline for hard questions, reasoning-only for common questions, and API-only for intermediate questions. In practice, although intermediate questions were reverse-engineered from API-only outputs, the full pipeline still outperforms API-only on most of them. Since the full pipeline is more costly, reweighting quality and cost could in principle reverse this ranking. Empirically, however, any weighting that lifts API-only above full also lifts reasoning-only above API-only. No setting in our calculation makes API-only top-ranked for the intermediate category; we therefore selected 0.02 as the configuration closest to the goal. This pattern suggests a possible evaluator bias toward longer or more information-dense responses, which we discuss further in the Limitations section.

5 Ablation Study

For each case study, we selected the training checkpoint with the highest % favored tool on the cross-validation held-out fold. At the selected checkpoint, we verified convergence using three additional signals: the average reward had plateaued, the step-wise policy KL between consecutive updates was small, and the KL divergence against the reference policy had stabilized. All cross-validation predictions reported here are from a single training run per fold. We did not run multiple training seeds due to compute constraints, but note this limitation in the Limitations section.

5.1 Effect of Structured Input on Untrained Actor

We compare structured against plain input under two reward configurations, WS0 (quality only) and

WS02 (quality with a small cost term), and report tool selection both before and after GRPO training in Table 1. The arrow indicates prediction from the untrained to the trained actor, and numbers without an arrow indicate unchanged predictions. This setup lets us isolate the contribution of input format from the contribution of the reward configuration.

- **Structured input already helps before any training.** Comparing the pre-arrow numbers in Table 1, the untrained actor reaches 69% favored selection with structured input versus 50% with plain input, a gap driven almost entirely by common questions, where plain input routes 10 of 12 to the full pipeline. Reasoning traces explain why: with structured input, the actor's reasoning is anchored on explicit complexity cues; with plain input, it falls back on pharmacokinetic concepts evoked by the question text and defaults to the full pipeline that the system prompt describes as the richest evidence source.

- **Pure quality reward (WS0) degrades both cases.** Under WS0, structured input regresses from 69% to 64%, and plain input remains at 50%. We attribute the structured-input regression to the full pipeline's dominant quality scores relative to API-only and reasoning-only (Appendix D.1), which produce a highly imbalanced reward landscape in which roughly two-thirds of questions theoretically favor full. GRPO amplifies this imbalance rather than correcting it.

- **A small cost term (WS02) recovers and improves on the untrained actor.** Under WS02, structured input reaches 72%, the highest across all four cases, and plain input jumps from 50% to 64%. The cost penalty offsets the quality bias toward full just enough to let GRPO move common questions toward reasoning-only without biasing hard-question routing. Structured input still leads plain by 8% after training, confirming that input format and reward configuration contribute independently.

One residual limitation remains visible even in the best case: under WS02 with structured input, the trained actor still routes only 2 of 12 intermediate questions to API-only. We trace this to evaluator-bias effects.

Weight	Input & Tool	Hard	Inter.	Common
WS0	*Structured* (69%→64%)			
	Full	**11→12**	8→10	1→3
	API-Only	1→0	**4→2**	1→0
	Reasoning-Only	0	0	**10→9**
	Plain (50%→50%)			
	Full	**11→10**	7→6	10
	API-Only	2	**5→6**	0
	Reasoning-Only	0	0	**2**
WS02	*Structured* (69%→72%)			
	Full	**11→12**	8→10	1→0
	API-Only	0	**4→2**	1→0
	Reasoning-Only	0	0	**10→12**
	Plain (50%→64%)			
	Full	**11**	7→8	10→0
	API-Only	1	**5→4**	0→2
	Reasoning-Only	0	0	**2→10**

Table 1: Tool selection by structured and plain input cases under WS0 and WS02. Bold numbers indicate the (theoretically) preferred pipeline for each difficulty category. The arrow indicates prediction from the untrained to the trained actor

5.2 Deeper Dive Into Components of Structured Input

To isolate the contribution of each structured-input component, we ablate the full structured input from Table 1 (WS02 structured, our best-performing scenario) by removing one or two components, and report the resulting tool selection in Table 2. We use M for PGx metadata, G for graph complexity, and F for question facets.

For two-component configurations, the overall % favored tool changes little before and after training, so this aggregate metric is not sensitive enough to reveal per-component effects. The per-category routing patterns differ substantially across configurations even when the aggregate score is identical, and single-component cases additionally show whether each component can stand alone.

- **PGx metadata is the primary decision signal.** M only after training reproduces the same per-category routing as the structured input with full components scenario, showing that metadata alone is sufficient to recover the trained behaviour.

- **Graph complexity is redundant once metadata is present.** M+G and M only yield identical per-category routing after training. Although reasoning-trace inspection confirms that the actor frequently references graph-complexity features, these references do not change routing decisions, so graph complexity plays a verbal rather than functional role.

- **Graph complexity alone is unusable.** Only

Components	Tool	Hard	Inter.	Common
M+G	Full	**11→12**	6→10	2→0
(72%→72%)	API-Only	1→0	**6→2**	1→0
	Reasoning-Only	0	0	**9→12**
M+F	Full	**12→11**	10→6	1→4
(69%→69%)	API-Only	0→1	**2→6**	0
	Reasoning-Only	0	0	**11→8**
F+G	Full	**11→12**	7→5	2→3
(67%→67%)	API-Only	1→0	**5→7**	2→4
	Reasoning-Only	0	0	**8→5**
Only M	Full	**12**	8→10	2→0
(72%→72%)	API-Only	0	**4→2**	0
	Reasoning-Only	0	0	**10→12**
Only G	Full	**12**	5→10	2→6
(72%→42%)	API-Only	0	**7→2**	3→5
	Reasoning-Only	0	0	**7→1**
Only F	Full	**12→11**	10	6→4
(42%→53%)	API-Only	0→1	**2→0**	5→0
	Reasoning-Only	0	0	**1→8**

Table 2: Tool selection from different component sets. M = PGx metadata, G = graph complexity, F = question facets. Bold numbers indicate the preferred pipeline for each difficulty category. The arrow indicates prediction from the untrained to the trained actor. WS02 is the selected weight set

G drops from 72% to 42% favored after training, with common questions scattered across all three pipelines. Without entity grounding from metadata, graph-structural features become noise that GRPO actively amplifies in the wrong direction.

- **Question facets need grounding from other components.** M+F and F+G are the only configurations that increase intermediate API-only selections after training (2 → 6 and 5 → 7), while Only F drops them to 0. Facets encode query intent but need entity or graph grounding to translate it into routing decisions, making them functionally complementary to the other components rather than redundant.

5.3 Overview of Quality Score and Token Usage

Since the % favored tool metric is not sensitive enough on its own, and to obtain a more holistic understanding of the quality–cost trade-off, we now examine quality score and token usage across configurations (Table 3). We use the always-full-pipeline configuration as the baseline, since without actor orchestration the default is to invoke every module for every query. Average runtime correlates strongly with average token count (Pearson $R^2 \approx 0.9$), so we report token usage only.

From the six component configurations analysed in the previous subsection, three (M+F, Only F, Only G) exhibited substantial post-training routing degradation or unstable convergence and are therefore omitted from this quality–cost comparison; we revisit their behaviour, and the conditions under which question facets and graph complexity might contribute meaningfully, in the Limitations section.

Two findings establish the value of structured input and cost-aware training. First, the trained Only-M WS02 actor reaches a 28.0% higher quality score than the baseline with 18.4% fewer tokens. The token saving comes from selectively invoking the full pipeline only when needed; the orchestration value lies in directing reasoning effort to where it pays off, not in indiscriminate token reduction (the untrained plain-input configuration uses 31.1% fewer tokens than the baseline but only achieves a 3.0% quality improvement, far below DATO's 28.0%). Second, untrained actors with structured input already reach 4.04–4.07 (+20.9% to +21.9%), so most of the quality gain comes from input format rather than training. The biased reward landscape appears to drive what training adds on top: WS0 regresses to 3.98 while consuming more tokens than the untrained configuration, and only WS02 recovers and surpasses the untrained level.

Two further findings refine the component analysis from the previous subsection. Only M achieves the highest quality of all retained configurations at 4.28 (+28.0%) with token cost matching full components. Notably, Only M already leads M+G+F before any training (4.07 vs. 4.04), and training widens this gap (4.28 vs. 4.17), indicating that the additional components are not merely redundant but slightly noisy under the current reward. M+G matches Only M on both quality (4.26) and token count (8,000 vs. 7,955), confirming that graph complexity is fully redundant once metadata is present.

Based on these results, we recommend the Only M (WS02) configuration as DATO's deployment setting. The latent potential of question facets and graph complexity, together with the specific conditions under which they currently fail to contribute, is discussed in the Limitations section.

6 Summary

We introduced Difficulty-Aware Tool Orchestration (DATO), the first pharmacogenomics-focused adaptive agentic framework built around a small open-source orchestrator. Our experiments on hard, intermediate, and common PGx questions surface three findings that we believe generalize beyond

Actor / Weight Set	Input Style	Avg. Quality	% Score Δ	Avg. Token	% Token Δ
Always-full-pipeline	Plain	3.34	–	9,754	–
Untrained	Plain	3.44	3.0%	6,724	−31.1%
Untrained	M+G+F	4.04	20.9%	7,438	−23.7%
Untrained	Only M	4.07	21.9%	7,569	−22.4%
WS0	M+G+F	3.98	19.3%	7,781	−20.2%
WS02	M+G+F	4.17	24.7%	7,928	−18.7%
WS02	Only M	**4.28**	**28.0%**	7,955	**−18.4%**
WS02	M+G	4.26	27.7%	8,000	−18.0%

Table 3: Overview of quality score and token usage across configurations. Percentage changes are relative to the always-full-pipeline baseline.

this work. First, knowledge-graph-derived PGx metadata is essential for small actors, since without explicit entity grounding an 8B model systematically over-selects the most comprehensive pipeline even on clearly simple queries; the additional facets and graph-complexity features show limited effect under the current reward landscape but remain a target for future work (see Limitations). Second, reinforcement learning alone cannot correct a biased reward landscape, and in our setting GRPO amplifies rather than mitigates the dominance of the full pipeline when quality scores favor it disproportionately. Third, a modest cost-aware penalty in the reward function is sufficient to correct routing on out-of-scope common questions. Together, these findings suggest that small open-source orchestrators, when paired with knowledge-graph-derived structured input and a carefully calibrated reward, offer a scalable path toward deploying tool-augmented clinical reasoning in resource-limited hospital settings.

Limitations

Several limitations point to clear directions for future work.

Evaluator bias. The LLM-based quality evaluator appears to exhibit a preference for longer, more information-dense responses on queries with PGx content. We infer this indirectly from the weight-search analysis in Section 3.4 (and Section 4.2's discussion), where no weighting recovered API-only as the top-ranked pipeline for intermediate questions; we did not run a controlled study to isolate evaluator behavior directly. This apparent bias may prevent the API-only pipeline from being correctly identified as optimal for intermediate questions even when the intermediate questions were designed to be answerable by API-only alone, biasing the actor toward the full pipeline whenever PGx entities are present. Independently of whether this bias is systematic, a finer-grained alternative is to evaluate responses at the atomic-claim level, following the VeriFact framework of Chung et al. (2026), which decomposes generated text into individual factual claims and verifies each independently. Such an evaluator yields a reward signal that distinguishes which specific claims are correct, missing, or unsupported, rather than collapsing the judgment into a single comprehensiveness-leaning score. Adapting this approach to PGx with CPIC and PharmGKB as the verification base is a natural next step.

Data scale. Even with evaluator bias resolved, the 36-question test set (12 per difficulty category) is likely too small for the actor to learn sharp decision boundaries between adjacent difficulty levels. Our current dataset was constructed in the forward direction, where questions were written first and graph complexity and question facets were then computed from them for scoring purposes. A natural way to scale up is to invert this process and generate questions from targeted values of graph complexity and question facets, producing a controlled distribution that spans hard, intermediate, and finer difficulty levels in between. This gives the actor more separable training signals across category boundaries.

Underutilization of question facets and graph complexity. Our ablation identifies PGx metadata as the dominant decision signal, with question facets and graph complexity contributing little beyond it. We attribute this to the comprehensiveness-favoring reward landscape discussed above rather than a fundamental limitation of either component. Once evaluator bias and data scale are addressed, the inverted data-generation procedure above provides a direct route to revisit this question: by spanning a controlled range of facet and graph-complexity values, the resulting dataset would expose the actor to settings where these components must contribute to correct routing. Table 2 already hints at a non-trivial learning signal in this direction, with Only F improving from 42% to 53% favored after training.

The details of these future directions are in Appendix E.

Use of AI Assistants

We made extensive use of Claude Opus 4.7 and Cursor (an AI-powered code editor) as our primary AI coding assistants throughout the development of the DATO system. These tools were primarily employed to support framework design, code implementation, preprocessing pipeline construction (including PGx-KG integration), GRPO training setup, ablation studies, and the execution of all experimental test tasks. In addition, Claude Opus 4.7 and Grok 4.3 were used for grammar polishing and prose smoothing of the manuscript. Separately, Grok 4.3 served as the LLM backend for our criteria-based quality evaluator (Section 3.4). All scientific claims, experimental design, analysis, and conclusions presented in this paper are the authors' own work.

Ethics Statement

Broader Impact. DATO is intended as a research prototype for tool-augmented clinical reasoning, not for direct deployment in patient care. PGx recommendations from any LLM-based system require validation by licensed clinicians before clinical use. Misuse could include over-reliance on tool outputs without clinical oversight, particularly in resource-limited settings where verification capacity is limited. In addition, deployment without adequate clinician training risks treating AI-generated PGx recommendations as definitive rather than advisory, potentially undermining evidence-based clinical judgment. Additional risks include potential hallucination in high-stakes clinical recommendations and over-reliance by clinicians in resource-limited settings; we emphasize that all outputs require licensed pharmacist verification.

Annotation. Hard-difficulty PGx questions were annotated on a voluntary basis by three licensed pharmacists who are research collaborators on this project, as described in Appendix F. The annotation task involved no patient data.

References

Dhanush Borishetty, Peter Banda, Nikhilesh Andhi, Aakash Desai, Bharath Ram Uppili, Naga Mithil Samudrala, Shravani Shriya Palanki, Dheeraj Reddy Bobbili, and Gayatri Rangarajan Iyer. 2025. PGxRAG: A retrieval augmented generation supported pharmacogenomics assistant. *medRxiv*.

Philip Chung, Akshay Swaminathan, Alex J. Goodell, Yeasul Kim, S. Momsen Reincke, Lichy Han, Ben Deverett, Mohammad Amin Sadeghi, Abdel-Badih Ariss, Marc Ghanem, David Seong, Andrew A. Lee, Caitlin E. Coombes, Brad Bradshaw, Mahir A. Sufian, Hyo Jung Hong, Teresa P. Nguyen, Mohammad R. Rasouli, Komal Kamra, and 10 others. 2026. Verifying facts in patient care documents generated by large language models using electronic health records. *NEJM AI*, 3(1).

Muhammad Omar Faruk. 2025. A large-scale pharmacogenomic knowledge graph for drug-gene-variant-disease discovery. *medRxiv*.

Zhuohan Ge, Haoyang Li, Yubo Wang, Nicole Hu, Chen Jason Zhang, and Qing Li. 2026. ClinicalAgents: Multi-agent orchestration for clinical decision making with dual-memory. *arXiv preprint arXiv:2603.26182*.

Yaejin Jeon, Mi Seon Youn, Sunghoon Kang, Jonghyung Park, Eun Sil Kim, Juyoung Kim, and Ju Han Kim. 2026. Hierarchical RAG enhances a pharmacogenomic AI assistant in guideline related queries. *Computers in Biology and Medicine*, 200:111323.

Karl Keat, Rasika Venkatesh, Yidi Huang, Rachit Kumar, Sony Tuteja, Katrin Sangkuhl, Binglan Li, Li Gong, Michelle Whirl-Carrillo, Teri E. Klein, Marylyn D. Ritchie, and Dokyoon Kim. 2025. PGxQA: A resource for evaluating LLM performance for pharmacogenomic QA tasks. In *Pacific Symposium on Biocomputing*, volume 30, pages 229–246.

Qintong Li, Jiahui Gao, Sheng Wang, Renjie Pi, Xueliang Zhao, Chuan Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Forewarned is forearmed: Harnessing LLMs for data synthesis via failure-induced exploration. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

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, and 3 others. 2024. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations (ICLR)*.

Shang-Chun L. Lu, Hsin Yang, Hui-Hsin Xue, Ping Lin Tsai, Yu Jing Weng, Shiou-Chi Li, Jen-Wei Huang, and Hui Hua Chang. 2026. CoVaPh: A vision-language multi-agent dialogue system for tool-augmented pharmacogenetic reasoning and personalized guidance. In *Proceedings of the 16th International Workshop on Spoken Dialogue System Technology (IWSDS)*, pages 441–451, Trento, Italy.

Clint Solomon Mathialagan, Alexander Nip, and Ashwani Bhat. 2026. Drug safety agents using graphs and ontologies. *medRxiv*.

Mullai Murugan, Bo Yuan, Eric Venner, Christie M. Ballantyne, Katherine M. Robinson, James C. Coons, Liwen Wang, Philip E. Empey, and Richard A. Gibbs. 2024. Empowering personalized pharmacogenomics with generative AI solutions. *Journal of the American Medical Informatics Association*, 31(6):1356–1366.

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. 2023. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. *arXiv preprint arXiv:2307.16789*.

Ashley Rector, Keaton Minor, Kamden Minor, Jeff McCormack, Beth Breeden, Ryan Nowers, and Jay Dorris. 2025. Validating pharmacogenomics generative artificial intelligence query prompts using retrieval-augmented generation (RAG). *arXiv preprint arXiv:2507.21453*.

Mary V. Relling, Teri E. Klein, Roseann S. Gammal, Michelle Whirl-Carrillo, James M. Hoffman, and Kelly E. Caudle. 2020. The clinical pharmacogenetics implementation consortium: 10 years later. *Clinical Pharmacology & Therapeutics*, 107(1):171–175.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*.

Samuel Schmidgall, Rojin Ziaei, Carl Harris, Ji Woong Kim, Eduardo Pontes Reis, Jeffrey Jopling, and Michael Moor. 2026. AgentClinic: A multimodal benchmark for tool-using clinical AI agents. *npj Digital Medicine*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Yu Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. HybridFlow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems (EuroSys '25)*.

Hongjin Su, Shizhe Diao, Ximing Lu, Mingjie Liu, Jiacheng Xu, Xin Dong, Yonggan Fu, Peter Belcak, Hanrong Ye, Hongxu Yin, Yi Dong, Evelina Bakhturina, Tao Yu, Yejin Choi, Jan Kautz, and Pavlo Molchanov. 2025. ToolOrchestra: Elevating intelligence via efficient model and tool orchestration. *arXiv preprint arXiv:2511.21689*.

Ziyue Wang, Junde Wu, Linghan Cai, Chang Han Low, Xihong Yang, Qiaxuan Li, and Yueming Jin. 2025. MedAgent-Pro: Towards evidence-based multi-modal medical diagnosis via reasoning agentic workflow. *arXiv preprint arXiv:2503.18968*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. AutoGen: Enabling next-gen LLM applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Yinghao Zhu, Ziyi He, Haoran Hu, Xiaochen Zheng, Xichen Zhang, Zixiang Wang, Junyi Gao, Liantao Ma, and Lequan Yu. 2025. MedAgentBoard: Benchmarking multi-agent collaboration with conventional methods for diverse medical tasks. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*.

A Prompts

A.1 Tool Orchestrator

In the Tool Orchestrator prompt, the three pipelines are exposed to the actor as `tool_a` (API-only), `tool_b` (Reasoning-only), and `tool_d` (Full). It was done to avoid the use of specific keywords. For clarity, we use the descriptive names throughout the main text and reserve the `tool_*` labels for the prompt itself:

> You are a pharmacogenomics tool orchestrator.
>
> Your job is to choose exactly one available tool that best fits the user's query.
>
> **Available Tools**
>
> 1. `tool_a`
> Description: [Lookup tables which contain standardized phenotype mappings, allelic definitions/functions/frequencies, and computational prescribing recommendations] → [Reasoning]
>
> 2. `tool_b`
> Description: [Reasoning] (no external data retrieval)
>
> 3. `tool_d`
> Description: [Lookup tables which contain standardized phenotype mappings, allelic definitions/functions/frequencies, and computational prescribing recommendations] + [Manuscripts detail pharmacokinetic mechanisms, clinical evidence grading, strength of recommendation, therapy justifications, and clinical caveats] → [Reasoning]
>
> **Input**
>
> You will receive: `original_query` (the patient's clinical question), `num_question_facets` (integer count of distinct clinical aspects in the query), `facets_description` (list of the clinical aspect names, one entry per facet), `graph_complexity` (quantitative metrics from the knowledge graph: `seed_count`, `total_nodes`, and `used_relations`), and `pgx_metadata` (key extracted elements: genes, variants, drugs, etc.).
>
> **Instructions**
>
> - Read the `original_query` together with `num_question_facets` and `facets_description` to understand the clinical intent and complexity.
> - Analyze the `seed_count` and `used_relations` metrics to assess structural complexity.
> - Based on all the provided information, choose exactly ONE tool that is most suitable.
>
> **Output Format.** Respond with ONLY a single valid JSON object containing `reasoning` (brief rationale based on the question, facets, and `graph_complexity`), `name` (one of `tool_a`, `tool_b`, or `tool_d`), and `parameters` (empty object). Do not output anything else besides the JSON.

A.2 PGx Entity Extraction

The prompt instructs the LLM to act as a pharmacogenomics expert and to extract all biomedical entities that would appear as node names in the PGx-KG. Entities are extracted into five categories: **genes** (gene symbols, e.g., CYP2C19, CYP2D6, VKORC1, DPYD, TPMT, UGT1A1, SLCO1B1); **variants** (GRCh38 (hg38) genomic coordinates

in `chr:pos:ref>alt` format, converted from star alleles, rsIDs, or HGVS notations); **drugs** (drug names, e.g., warfarin, clopidogrel, tamoxifen, omeprazole, codeine); **adr** (Adverse Drug Reactions as UMLS CUI codes); and **diseases** (disease or condition names in plain English, matching ClinVar condition naming).

A series of explicit conversion examples is provided, including star allele → GRCh38 mappings such as CYP2C19*2 (rs4244285) → `10:94781859:G>A`, CYP2D6*4 (rs3892097) → `22:42128945:C>T`, and SLCO1B1*5 (rs4149056) → `12:21178615:T>C`, as well as ADR-to-UMLS CUI mappings such as neutropenia → `C0027947`, hepatotoxicity → `C0235378`, and Stevens–Johnson syndrome → `C0038325`.

The prompt enforces the following rules: use INN (International Nonproprietary Names) for drugs; if unsure which name the KG uses, list both the local and international name; strip clinical modifiers (e.g., "omeprazole-based triple therapy" → "omeprazole"); for combination therapies, extract each drug separately; convert allele/variant notations to GRCh38 coordinates; do not extract non-entity terms (e.g., "poor metabolizer", "62-year-old", "female"); be comprehensive, since missing an entity is worse than including an extra one; every category must be present in the output (use empty list if none found); and only output coordinates the model is confident about. The output is a single JSON object under the key `key_elements` with sub-keys `genes`, `variants`, `drugs`, `adr`, and `diseases`.

A.3 Question Facets Preprocessing

The prompt instructs the LLM to act as a pharmacogenomics expert and to classify each clinical pharmacogenetics question into zero or more of the following facets based on what the question is actually asking about:

1. **Pharmacogenetic Risk & Drug Response** — how genetic variants affect drug efficacy, metabolism, or PK/PD response.

2. **Dosing Recommendation & Adjustment** — specific dosing guidance, dose modifications, or titration based on genotype or phenotype.

3. **Safety & Toxicity Risk Assessment** — adverse drug reactions, toxicity risk, contraindications, or safety concerns related to genetic variants.

4. **Monitoring & Follow-up Strategy** — therapeutic drug monitoring, lab tests, follow-up schedules, or clinical surveillance after initiating therapy.

5. **Alternative Therapy & Clinical Context** — alternative drug choices, switching therapies, or broader clinical decision-making context.

Rules: select ALL facets that apply (a question may match 0 to 5 facets); only select a facet if the question explicitly or clearly implicitly asks about that aspect; if the question is purely definitional or conceptual with no clinical action implied, return an empty list. The output is a single JSON object `{"question_facets": [...]}`.

A.4 Quality Score Evaluation

The prompt instructs the evaluator to act as a healthcare professional/pharmacist applying rigorous first-principles reasoning to pharmacogenetics queries. The evaluator is asked to read model responses to clinical pharmacogenetic questions, assess them meticulously, and grade each one against reference answers and the most up-to-date official information from CPIC and PharmGKB. Scores increment by 0.5, with explicit emphasis on accuracy/factuality as the most important factor in PGx, including verification of every citation, reference data point, and date.

Each response is scored on four criteria, each on a 1–5 scale:

Accuracy of Content (1: completely inaccurate; 2: mostly inaccurate with some correct points; 3: somewhat accurate with significant errors; 4: mostly accurate with minor errors; 5: completely accurate and up-to-date).

Relevance to Query (1: not related; 2: slightly related but misses the main point; 3: moderately relevant; 4: mostly relevant; 5: fully relevant and directly answers the question).

Completeness of Information (1: very incomplete; 2: somewhat incomplete; 3: adequate but missing some details; 4: mostly complete with minor gaps; 5: fully complete and easy to follow).

Clarity and Coherence (1: confusing and poorly structured; 2: somewhat unclear; 3: mostly clear with organizational issues; 4: clear and well-organized; 5: very clear and easy to understand).

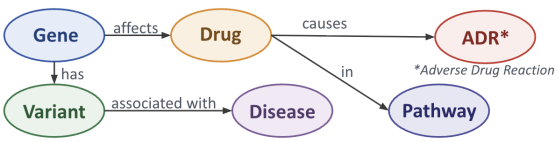

Figure 4: Schema of the PGx Knowledge Graph (PGx-KG). Node types include Gene, Variant, Drug, Disease, Pathway, and ADR (Adverse Drug Reaction).

B Structured Input Data

B.1 Diagram of the PGx KG

Figure 4 shows the schema of the pharmacogenomic knowledge graph used in this work.

B.2 Graph Complexity Metrics and Calculation

To compute graph-complexity metrics, we define a *seed* as a PGx-KG node that matches a gene, variant, drug, or ADR extracted from the query. Based on these seeds and the induced subgraph they form, we use the following three metrics as our complexity signals:

- **Seed counts:** the count of matched genes, variants, drugs, and ADRs in the PGx-KG.
- **Total nodes:** the total number of nodes in the seed-induced subgraph.
- **Used relationships:** the number of edge types in the seed-induced subgraph (e.g., gene–variant is one type).

B.3 Example

Question. A 12-year-old pediatric patient of mixed European-African descent with acute lymphoblastic leukemia is on mercaptopurine maintenance therapy. Genotyping reveals heterozygosity for TPMT*3A (intermediate activity). Analyze the risk of myelosuppression, recommend dose adjustments (e.g., 30–80% reduction per CPIC), and discuss therapeutic drug monitoring for 6-TGN levels to balance efficacy and toxicity.

PGx Analysis.

- **Question Facets:** `num_question_facets=4, facets_description=[Pharmacogenetic risk analysis, Dosing recommendation, Safety & toxicity risk, Monitoring strategy]`
- **Graph Complexity:** `seed_count=2, used_relations=1, total_nodes=6`
- **Key Elements:** `genes=[TPMT], variants=[TPMT*3A], drugs=[mercaptopurine], conditions=[acute lymphoblastic leukemia]`

B.4 Other Graph Complexity Metrics

In our PGx clinical queries, all other tested metrics were highly correlated with the three selected, or showed little information, indicating that PGx clinical queries are relatively simple compared to large-scale, wide-spectrum medical tasks such as drug discovery. Nevertheless, the structured-input concept remains promising for other medical domains.

- Graph density ($2 \times \text{edges}/(\text{nodes} \times (\text{nodes} - 1))$)
- Graph density variation ($\log(\text{nodes}) \times$ graph density)
- Total edges
- Average degree
- Shortest-path length (diameter)
- Graph entropy
- Relationship count

C Details of Training Adaptations

C.1 Diversity Hint

During testing we observed that the Qwen3-8B actor frequently converged to a single trajectory too early, selecting the same tool across all candidates for a given query. Raising temperature or entropy did not help and increased parsing errors. We suspect keyword-sensitivity effects similar to those described above. To mitigate this, we appended a round-robin diversity hint to the actor prompt during early training epochs (e.g., "encourage using API-only"), cycling through the three pipelines to ensure exposure. This simple intervention prevented premature convergence.

C.2 Batch-Level Normalization

Standard GRPO normalizes advantages of rollouts within each query group (query-level). With our small actor, premature convergence persisted in later rounds even with the diversity hint: once the actor settled on one pipeline for most queries,

Category	Pipeline	Mean	SD	Min	Max
Hard	Full	4.24	0.42	3.48	4.75
Hard	API-Only	4.07	0.30	3.49	4.39
Hard	Reason.-Only	3.88	0.30	3.47	4.37
Inter.	Full	4.80	0.18	4.47	5.00
Inter.	API-Only	4.01	0.53	3.00	4.71
Inter.	Reason.-Only	3.98	0.51	3.15	4.57
Common	Full	1.00*	0.00	1.00*	1.00*
Common	API-Only	1.00*	0.00	1.00*	1.00*
Common	Reason.-Only	3.84	0.51	3.16	4.77

Table 4: Quality score breakdown by question category and pipeline. Inter. = Intermediate, Reason.-Only = Reasoning-Only. The asterisk (*) marks scores corresponding to the response "No CPIC Guideline Related," which is treated as the lowest-quality answer.

query-level normalization produced no learning signal.

We therefore switched to batch-level normalization, which jointly normalizes across all rollouts in a batch. Even when the actor selects the same tool for every rollout of a given query, different queries within the batch typically select distinct tools, preserving the relative-advantage signal for policy updates. This also improves stability under limited data. The main drawback is slower learning from a weaker per-query signal, which we addressed by raising the learning rate and running more training rounds.

D Supporting Tables

D.1 Quality Score Overview

Mean, standard deviation, minimum, and maximum response quality scores of the three tool pipelines across question categories are reported in Table 4.

E Future Work

E.1 Adjust or Change Reward Functions

The current criteria for assessing response quality across pipelines require refinement. At present, there is no penalty for excessively long answers that contain irrelevant information (false positive), which makes the performance gap between the full pipeline and the API-only pipeline less pronounced. On the other hand, the criteria do not allow us to understand which points in the response do not match the ground truth, and at what level. Future work could address this by introducing the concept of "atomic claim" (Chung et al., 2026) to do a more detailed and stable evaluation.

E.2 Novel Data Generation

A natural extension is targeted data generation that uses an LLM proposer to produce questions concentrated in the specific aspects where the current actor struggles, following the failure-induced data synthesis paradigm of Li et al. (2025), rather than sampling broadly across topics. By identifying failure-prone regions from training trajectories, a generation pipeline can synthesize new questions concentrated on those decision boundaries. For DATO, the key issue is intermediate-difficulty cases where the actor consistently mis-routes between full-pipeline and API-only; generating a targeted set of boundary questions in this region reduces expert-review burden by orders of magnitude while maximizing the marginal training signal per reviewed question, making iterative DATO refinement feasible under the expert-bottleneck constraints of clinical pharmacogenomics.

E.3 Improve Actor Reasoning Efficiency

After incorporating structured data as input, we observed that the actor rarely leverages question facets for decision-making, even though question facets are among the key features that could be used to distinguish hard from intermediate questions. We experimented with both explicit prompt guidance and a small positive reward for mentioning question facets. The former proved largely ineffective, while the latter led to reward hacking, in which the actor simply repeated the entire structured input in its thinking trace to artificially trigger the reward. Encouraging the actor to genuinely focus on question facets and other important features that it tends to neglect remains a significant challenge.

F Annotation Details

Annotators. Hard questions were annotated and validated by three academic researchers, all of whom are licensed pharmacists with prior experience interpreting CPIC pharmacogenomic guidelines. The annotators are research collaborators on this project; no external recruitment or crowdsourcing platform was used.

Task and Instructions. For each candidate question, annotators were instructed to consult the relevant CPIC guidelines and supporting literature while reviewing LLM-generated drafts of ground-truth answers prepared in the format conventionally used in clinical pharmacogenomic practice. They

were further asked to confirm that resolving the question requires reasoning beyond direct CPIC API lookup—thereby justifying its classification as hard difficulty, and to flag any ambiguous or out-of-scope items for removal. Disagreements were resolved by discussion.

Compensation. Annotation was carried out by the annotators on a voluntary basis as part of their participation in this collaborative research project. No separate compensation was provided. This arrangement is standard for collaborative academic research, where annotators are co-investigators on the project and receive academic credit through publication.

Consent and Ethics. Annotators were informed of the research purpose and the intended use of the data, including the planned anonymous publication of the resulting dataset, and consented to these terms. The annotation task involved no patient data and was therefore exempt from IRB review.

G Computational Setup and Artifacts

G.1 Computational Setup

All training and inference experiments were conducted on a cluster of 8×NVIDIA A100 80 GB GPUs. The Tool Orchestrator is based on Qwen3-8B (8.2B parameters; Yang et al., 2025), and GRPO training was carried out using 2 of the 8 GPUs in a data-parallel configuration via PyTorch FSDP, with the remaining GPUs reserved for parallel inference, evaluation, and ablation runs. Each full 2-fold cross-validation run took approximately 2 hours on 2 A100s; the total compute used for all reported experiments, including preliminary ablations, was approximately 120 GPU-hours.

G.2 Use of Artifacts

All software, models, and data resources used in this work are publicly available under open-source or open-data licenses, and our use is consistent with their intended research purpose:

- **Qwen3-8B** (Yang et al., 2025): Apache License 2.0
- **verl** (Sheng et al., 2025): Apache License 2.0
- **Ray** (distributed runtime): Apache License 2.0
- **PyTorch / FSDP** (model sharding): BSD-3-Clause
- **vLLM** (inference backend): Apache License 2.0
- **Hugging Face transformers**: Apache License 2.0
- **CPIC guidelines and CPIC API** (Relling et al., 2020): CC BY-SA 4.0 (academic use)
- **PharmGKB** (queried via the PGx-KG and indirectly via CPIC): CC BY-SA 4.0 (academic use)
- **PGx-KG** (Faruk, 2025): released on medRxiv under CC BY 4.0

We do not redistribute any underlying clinical data. Our 36-question evaluation set is newly constructed by the authors (Appendix F) and contains no patient-identifying information. Any code released with this work will be under the Apache 2.0 license.

G.3 Implementation Details

Training was implemented in **verl 0.4.0.dev**, an open-source RL training framework that orchestrates distributed workers via **Ray 2.54.1** and uses **PyTorch 2.10.0 FSDP** (Fully Sharded Data Parallel) as the model-sharding backend. Inference and rollout generation use **vLLM 0.18.1**. All models and tokenizers were loaded via **Hugging Face transformers 5.5.3**. Knowledge-graph construction and graph-complexity computation use **NetworkX 3.6.1**. The criteria-based reward evaluator (Appendix A.4) is served via **Grok 4.3**.

H Training Hyperparameters

GRPO training hyperparameters for the Tool Orchestrator are summarized in Table 5. We did not perform exhaustive hyperparameter search due to compute constraints; values were chosen based on verl defaults and small-actor adaptations described in Section C. The configuration listed below was applied to both cross-validation folds.

Hyperparameter Design Notes. Three choices deserve brief justification:

(1) **Group size $G = 3$.** Standard GRPO uses larger groups (often 8 or 16) to obtain stable group-relative advantages. We adopt $G = 3$ because the actor selects among exactly three discrete pipelines (API-only, Reasoning-only, Full), so any group size beyond 3 produces redundant rollouts once the actor begins to favor one pipeline. Combined with

Hyperparameter	Value
Optimizer	AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay $= 1\mathrm{e}{-2}$)
Learning rate	$2\mathrm{e}{-5}$, cosine decay to $2\mathrm{e}{-6}$ (`min_lr_ratio = 0.1`)
Warmup	None (`lr_warmup_steps_ratio = 0`)
Gradient clipping	$\lVert g \rVert_2 \leq 1.0$
Training length	6 epochs (12 update steps)
Global batch size	9 prompts per step ($\times$ 3 rollouts = 27 sequences)
Mini-batch size	3 prompts
Group size G (rollouts per query)	3
KL penalty β	$1\mathrm{e}{-3}$, applied in the reward (`use_kl_in_reward = True`)
Clip ratio	asymmetric (dual-clip): $\varepsilon_{\text{low}} = 0.1$, $\varepsilon_{\text{high}} = 0.3$
Rollout temperature	1.2 (top-p = 0.95, top-k = 10)
Max sequence length	8192 prompt / 4096 response tokens
Diversity hint window	global steps 1–9 (off for steps 10–12)

Table 5: GRPO training hyperparameters.

batch-level advantage normalization (Appendix C), this yields a stable learning signal under our small-data regime.

(2) **Asymmetric (dual-clip) PPO ratio bounds.** We use $\varepsilon_{\text{low}} = 0.1$ and $\varepsilon_{\text{high}} = 0.3$ rather than a symmetric ratio. The wider upper bound allows the actor to upweight high-reward rollouts more aggressively, which is important given the small number of update steps. The narrower lower bound limits over-suppression of currently-disfavored pipelines, mitigating the premature-convergence behavior we observed during preliminary runs.

(3) **Diversity hint window (steps 1–9 of 12).** The diversity hint (described in Appendix C) is applied during the first $\sim$75% of training to encourage exploration across all three pipelines, then disabled for the final three steps to let the policy commit to its preferred routing. This schedule was selected based on training-log inspection showing that policy entropy stabilizes by around step 9.