STRUCTURE OR AGENCY? PRIOR-GUIDED AGENCY ALLOCATION FOR MULTIMODAL DOCUMENT MINING

Anonymous authors
Paper under double-blind review

ABSTRACT

As large language model (LLM) agents gain stronger planning and tool-use capabilities, a common trend is to grant them greater autonomy. Yet many real-world tasks already benefit from reliable procedural knowledge, making some model-controlled decisions unnecessary and error-prone. This raises a central question: *which decisions should be controlled by the model, and which should be determined by task structure?* To answer this question, we characterize the structure–agency trade-off through Agency Complexity, which measures the decision space exposed to model control, and two competing risks: Decision Error from incorrect model choices and Structural Bias from constraints that exclude successful trajectories. Guided by this analysis, we propose Prior-Guided Agency Allocation (PGAA), a principled framework for allocating control between reliable procedural priors and model judgment, formalized through a Human-Prior Decision Graph. To evaluate PGAA, we construct MineDoc, a challenging and representative benchmark for Web multimodal document mining, combining industrial procedural knowledge with instance-dependent judgments in retrieval, captioning, and LLM-as-a-Judge verification. It includes internal real-world business data and two public-data test sets simulating business scenarios, derived from Encyclopedic-VQA data associated with iNaturalist 2021 and Google Landmarks Dataset v2, respectively. We instantiate PGAA through a Filter–Produce–Reflect-ReAct mechanism and evaluate it on MineDoc. Under matched models, tools, inputs, and inference budget limits, PGAA improves reliability over fixed workflows and autonomous baselines while reducing inference cost relative to autonomous execution. Analyses of agency allocation and prior quality further support the trade-off, suggesting that effective agent design allocates control where model judgment is most valuable.

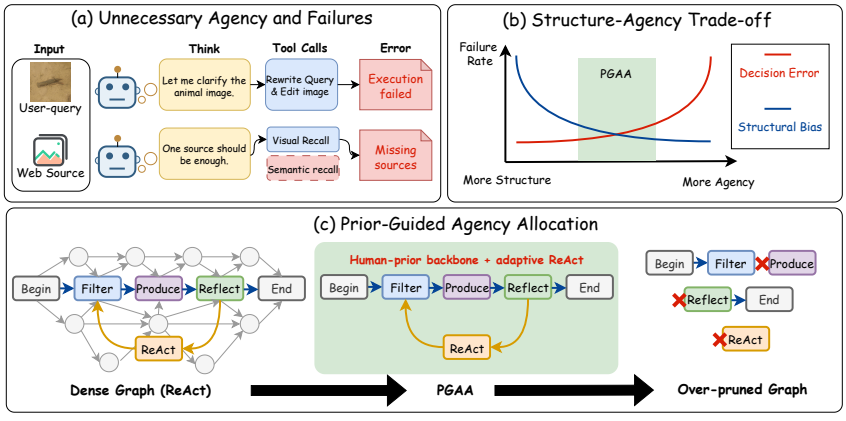


Figure 1: **Prior-guided agency allocation: balancing structure and model judgment.** **(a)** Unnecessary agency causes processing failures or incomplete retrieval. **(b)** Greater agency increases exposure to Decision Error while reducing Structural Bias, motivating PGAA. **(c)** Pruning low-frequency transitions reveals a human-prior backbone; excessive pruning disconnects Begin from End. **Notably! when only one connected path remains, it exactly matches the engineered workflow constructed from accumulated human prior knowledge.** PGAA preserves this FPR backbone and retains adaptive ReAct decisions. The graph is schematic; detailed real-world cases are provided in the appendix F.

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.

1 INTRODUCTION

Recent advances in large language models (LLMs) have extended their use beyond text generation to interactive task execution (Nakano et al., 2021; Yang et al., 2024), enabling agents to combine reasoning, tool use, and feedback from external environments (Yao et al., 2023; Shinn et al., 2023). As these capabilities improve, a common design trend is to grant agents greater autonomy over task execution. Yet many practical tasks already benefit from procedural knowledge accumulated through years of research, engineering, and deployment. Search, for example, commonly follows a structured pipeline of candidate retrieval and ranking (Guo et al., 2022; Nogueira et al., 2019). Such knowledge provides reliable guidance for routine execution, although it cannot anticipate every input-dependent or semantically ambiguous situation. Granting the model agency at uncertain decision points can improve adaptability (Hu et al., 2023), whereas delegating decisions already supported by reliable procedures may introduce unnecessary control errors. Figure 1(a) illustrates concrete examples: an autonomous agent edits the query image and rewrites the query before retrieval, encountering an image-processing failure, or invokes only one of several required retrieval sources, leaving potentially useful candidates unretrieved. Encoding reliable procedural requirements into the execution structure can prevent such errors while preserving model judgment for candidate assessment and subsequent adaptation. This raises a central design question: *which decisions should be controlled by the model, and which should be determined by existing task structure?*

To answer this question, we formalize and analyze the **structure–agency trade-off** under simplifying assumptions. We introduce *Agency Complexity* to quantify the decision space exposed to model control and analyze two competing risks: *Decision Error*, arising from incorrect model-controlled choices, and *Structural Bias*, arising when structural constraints exclude all successful trajectories for an instance. Our analysis shows that exposing additional procedural decisions to model control increases both Agency Complexity and the probability of decision error, creating opportunities for incorrect tool selection, redundant operations, and premature termination (Liu et al., 2024). Such errors can propagate through subsequent execution (Yao et al., 2025; Hu et al., 2025b). Conversely, relaxing structural constraints expands the admissible trajectory set and monotonically reduces or preserves Structural Bias, allowing successful strategies previously excluded by unreliable priors (Shi et al., 2025). As illustrated in Figure 1(b), these opposing effects give rise to the trade-off: greater agency increases exposure to model decision errors while reducing structural bias, whereas stronger structure has the reverse effect. This analysis motivates allocating control according to the relative reliability of procedural priors and model judgment.

Motivated by this trade-off, we propose **Prior-Guided Agency Allocation (PGAA)**, a principled framework for allocating control between reliable procedural priors and model judgment. We formalize execution as a *Human-Prior Decision Graph*, in which prior-specified Operator Nodes encode reliable procedural operations and transitions, while model-controlled Decision Nodes handle instance-dependent choices. Figure 1(c) visualizes the tool-call graph aggregated from execution traces, highlighting the most frequently traversed path. This dominant path aligns with established human procedural knowledge, illustrating recurring execution structure that can be encoded explicitly. PGAA assigns such reliable procedural relations to structural control while preserving model agency where semantic judgment is needed. For example, candidate normalization can follow a predefined procedure, whereas deciding whether the available evidence warrants another retrieval round requires instance-dependent assessment (Asai et al., 2024; Yan et al., 2024). Different inputs can therefore follow different trajectories within the same graph. This distinguishes PGAA from prompt-level guidance: rather than requiring the model to repeatedly interpret and enact every procedural instruction, PGAA directly enforces selected control relations.

To evaluate this framework in a setting that requires both procedural structure and model judgment, we construct **MineDoc**, a benchmark suite for Web multimodal document mining. This task draws on procedural knowledge accumulated through extensive industrial search practice, while requiring systems to jointly perform multimodal retrieval, evidence-grounded caption generation, and LLM-as-a-Judge verification. Given a textual question, a query image, and heterogeneous image–text candidates (Chang et al., 2022; Chen et al., 2022), a system must construct reliable evidence documents to support downstream question answering. Established procedures guide candidate processing, but semantic judgments remain essential: a textually relevant candidate may depict the wrong entity (Chen et al., 2023), and a plausible caption may contain unsupported claims. MineDoc comprises two benchmarks: **MineDoc-Private**, constructed from internal real-world business data,

and **MineDoc-Public**, comprising two test sets designed to simulate business scenarios, constructed from Encyclopedic-VQA (Mensink et al., 2023) associated with iNaturalist 2021 (Van Horn et al., 2021) and Google Landmarks Dataset v2 (Weyand et al., 2020), respectively.

On these benchmarks, we instantiate PGAA through a **FPR-ReAct** (Filter-Produce-Reflect–ReAct) mechanism, which coordinates perceptual hashing (Fridrich & Goljan, 2000; Zauner, 2010), visual and multimodal embeddings (Siméoni et al., 2025; Li et al., 2026), and vision-language models (Qwen Team, 2026) for candidate curation (Wu et al., 2026), evidence-grounded document generation (Chen et al., 2022), and hierarchical verification (Dong et al., 2025; Jing et al., 2024). The FPR structure governs routine execution, while ReAct adaptively selects tools for actions such as query rewriting and image editing when further work is needed. To isolate the effect of agency allocation, we compare FPR–ReAct with fixed workflows, fully autonomous ReAct, and ReAct supplied with the same priors through prompting, under matched foundation models, tools, inputs, and inference budget limits. Beyond end-to-end reliability and inference cost, we vary the amount and placement of model control, perturb prior quality, and analyze execution errors and their downstream propagation. FPR–ReAct achieves the best end-to-end reliability on both MineDoc-Private and MineDoc-Public, while reducing inference cost relative to autonomous execution.

Our contributions are fourfold:

- We formulate agency allocation as an agent-design problem, introduce Agency Complexity to quantify the decision space exposed to model control, and characterize the structure–agency trade-off through Decision Error and Structural Bias.
- We propose PGAA, a principled framework for allocating control between reliable procedural priors and model judgment, formalized through a Human-Prior Decision Graph.
- We construct MineDoc, a benchmark for Web multimodal document mining that combines established industrial procedures with challenging retrieval, caption generation, and verification tasks, covering internal real-world data and two test sets constructed from public datasets.
- We instantiate PGAA through an FPR–ReAct mechanism, achieving the best reliability among compared methods on MineDoc while reducing inference cost relative to autonomous execution. Analyses of agency allocation and prior quality further support the structure–agency trade-off.

2 Problem Formulation

We consider multi-step tasks in which some execution decisions are supported by established procedural knowledge, while others must be resolved from the semantic state of the current instance. Consider a task $\mathcal{T} = (\mathcal{S}, \mathcal{A}, \mathcal{O})$, where $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{O}$ denote the state, action, and observation spaces, respectively. At step t, a fully autonomous agent selects the next action from the execution history $h_t = (s_0, a_0, o_0, \ldots, s_t)$ according to $a_t \sim \pi_\theta(\cdot \mid h_t)$. Under this formulation, the model is responsible for both semantic reasoning and execution control.

Execution decisions, however, do not all require the same source of information. A *procedural decision* determines an execution relation supported by existing task knowledge, such as the ordering of processing stages or a transition associated with a deterministic outcome. A *semantic decision*, in contrast, requires instance-specific evidence, such as whether a candidate is relevant, whether two images depict the same entity, or whether the available evidence is sufficient.

This distinction motivates, but does not uniquely determine, the allocation of control. Whether a decision can be specified structurally depends on the reliability of the available prior. We therefore formulate the Agency Allocation Problem.

To study this problem, we introduce a Human-Prior Decision Graph to represent where control is assigned and Agency Complexity to quantify the decision space exposed to model control.

2.1 Human-Prior Decision Graph

We represent an execution structure as a directed graph $G = (V, E, v_{\text{start}}, V_{\text{term}})$ with a start node v_{start} and a set of terminal nodes V_{term}. Its nodes are partitioned into Operator Nodes V_O and Decision Nodes V_D, such that $V = V_O \dot{\cup} V_D$.

At an Operator Node $v \in V_O$, a prior-specified operator f_v determines the action and associated transition. At a Decision Node $v \in V_D$, the model selects among the admissible actions $\mathcal{A}_v$, which may lead to different subsequent trajectories. The resulting control policy is

$$\pi_G(a_t \mid h_t, v_t) = \begin{cases} \delta_{f_{v_t}(s_t)}(a_t), & v_t \in V_O, \\ \pi_\theta(a_t \mid h_t, \mathcal{A}_{v_t}), & v_t \in V_D, \end{cases} \tag{1}$$

where $\delta_{f_{v_t}(s_t)}$ denotes a deterministic policy that always selects the operator-specified action, and the subsequent graph transition is given by $v_{t+1} = \tau_G(v_t, a_t)$, where τ_G denotes the transition relation induced by the graph structure. Thus, model agency is restricted to Decision Nodes V_D, whereas actions at Operator Nodes V_O are determined by the structural prior. We call this mixed representation a Human-Prior Decision Graph. It fixes only reliable procedural relations, so different inputs can still induce different trajectories through the same graph.

2.2 Agency Complexity

Existing measures of agent complexity often focus on execution length, such as the total number of model invocations or tool calls. However, these quantities do not directly characterize the amount of execution control exposed to the model. Two agents may involve the same number of calls while differing substantially in the number of alternatives available to the model at each decision point.

We therefore define *Agency Complexity* from the total number of control configurations exposed to the model. We define $|\mathcal{A}_v|$ as the number of possible actions available at node v. The full control space therefore has size $\prod |\mathcal{A}_v|$. Taking the logarithm yields an additive measure:

$$\mathcal{C}_A(G) = \log \prod_{v \in V} |\mathcal{A}_v| = \sum_{v \in V} \log |\mathcal{A}_v| = \sum_{v \in V_D} \log |\mathcal{A}_v|, \tag{2}$$

where Operator Nodes satisfy $|\mathcal{A}_v| = 1$ and therefore contribute zero. Agency Complexity captures both the number of model-controlled decisions and the size of their corresponding action spaces. It characterizes the execution space exposed to model control, rather than the model's parameter count, reasoning capability, or the raw number of model or tool calls.

Under this formulation, we have $\boxed{\mathcal{C}_A(G_{\text{Fixed}}) = 0 \le \mathcal{C}_A(G_{\text{any}}) \le \mathcal{C}_A(G_{\text{Auto}})}$, where G_{Fixed} denotes a fully fixed workflow, G_{any} denotes any execution graph obtained by introducing structural constraints into the autonomous execution process, and G_{Auto} denotes the fully autonomous formulation. The corresponding derivation is provided in Appendix A.

3 Theoretical Analysis

Building on the Human-Prior Decision Graph and Agency Complexity introduced in Section 2, we analyze how the allocation of model control affects execution reliability. We focus on two distinct sources of failure: *decision error*, which arises from incorrect model-controlled decisions, and *structural bias*, which arises when structural constraints exclude all successful trajectories for some task instances. Their interaction gives rise to a structure–agency trade-off.

We organize the analysis around four questions: (i) whether unnecessary model-controlled decisions increase decision error? (ii) whether excessive structural constraints can exclude correct execution paths? (iii) how these two risks jointly determine an appropriate level of agency? (iv) when a Decision Node should be replaced by a prior-specified Operator Node? Full proofs are provided in Appendix B.

3.1 Decision Error

We first consider the cost of exposing procedural decisions to model control. Suppose an execution graph G_K contains K model-controlled procedural Decision Nodes encountered during execution. Let $\epsilon_j^p \in (0, 1)$ denote the probability that the model makes an incorrect control decision at node

j. Under a simplifying conditional-independence assumption, the probability that at least one such control error occurs is

$$e(G_K) = 1 - \prod_{j=1}^{K} (1 - \epsilon_j^p). \tag{3}$$

Proposition 1 (Decision-error accumulation). Consider adding a model-controlled procedural node v_{K+1} to G_K, producing G_{K+1}. If $|\mathcal{A}_{K+1}| > 1$ and $\epsilon_{K+1}^p > 0$, then $\mathcal{C}_A(G_{K+1}) > \mathcal{C}_A(G_K)$ and $e(G_{K+1}) > e(G_K)$. In particular,

$$e(G_{K+1}) - e(G_K) = \epsilon_{K+1}^p \prod_{j=1}^{K} (1 - \epsilon_j^p) > 0. \tag{4}$$

Proposition 1 shows that exposing unnecessary procedural decisions to model control increases both Agency Complexity and the probability of control error. Thus, when a procedural decision can be specified reliably in advance, delegating it to the model introduces avoidable execution risk. The proof is provided in Appendix B.1.

3.2 Structural Bias

We next consider the risk introduced by structural constraints. For an input $x \sim \mathcal{D}$, let $\mathcal{T}^*(x)$ denote the set of successful trajectories and $\mathcal{T}_G(x)$ the set of trajectories permitted by graph G. Structural Bias Risk measures the probability that the graph excludes all successful trajectories for an input, and is defined as

$$\delta(G) = \Pr_{x \sim \mathcal{D}} \left[\mathcal{T}^*(x) \cap \mathcal{T}_G(x) = \varnothing \right]. \tag{5}$$

Proposition 2 (Structural-bias bound). For any model policy π_θ, the success probability under graph G is upper-bounded by the probability that the graph admits at least one successful trajectory. Moreover, enlarging the set of admissible trajectories cannot increase Structural Bias Risk.

In particular,

$$P_{\text{success}}(G, \pi_\theta) \le 1 - \delta(G), \qquad \mathcal{T}_{G_1}(x) \subseteq \mathcal{T}_{G_2}(x)\, \forall x \implies \delta(G_2) \le \delta(G_1). \tag{6}$$

Proposition 2 shows that structural constraints impose a fundamental limit on achievable success: once all successful trajectories are excluded, model reasoning cannot recover them within the graph. Thus, overly restrictive structures can introduce structural bias by removing valid execution paths. The proof is provided in Appendix B.2.

3.3 Structure–Agency Trade-off

Propositions 1 and 2 identify two opposing sources of execution risk: increasing model control can increase Decision Error Risk, while imposing stronger structural constraints can increase Structural Bias Risk. We now characterize their joint effect. Under a simplifying analytical assumption that structural feasibility and model-control correctness factorize, the two risks yield the following decomposition of end-to-end reliability.

Proposition 3 (Structure–agency risk decomposition). Let $e(G)$ and $\delta(G)$ denote the Decision Error Risk and Structural Bias Risk of graph G, respectively. The end-to-end success probability and overall failure risk are

$$P_{\text{success}}(G) = \big[1 - \delta(G)\big]\big[1 - e(G)\big], \qquad R(G) = \delta(G) + e(G) - \delta(G)e(G). \tag{7}$$

For a candidate family of execution graphs $\mathcal{G}$, the minimum-risk allocation is

$$G^* = \arg\min_{G \in \mathcal{G}} R(G), \qquad \mathcal{C}_A^* = \mathcal{C}_A(G^*). \tag{8}$$

Proposition 3 shows that increasing model agency may increase Decision Error Risk, while stronger structural constraints may increase Structural Bias Risk. These opposing effects jointly form the structure–agency trade-off. The proof is provided in Appendix B.3.

3.4 PRIOR-GUIDED AGENCY ALLOCATION

We finally consider when a model-controlled Decision Node should be replaced by a prior-specified Operator Node. For a procedural node v_j, let ϵ_j^p denote the probability of an incorrect control outcome under model control, as defined in Section 3.1, and let ϵ_j^o denote the corresponding error probability under a prior-specified Operator Node. The latter also accounts for cases in which the procedural prior is incompatible with the current instance.

Proposition 4 (Prior-guided agency allocation). Consider a procedural Decision Node $v_j \in V_D$. If $|\mathcal{A}_j| > 1$ and $\epsilon_j^o < \epsilon_j^p$, then replacing model control with a prior-specified Operator Node reduces Agency Complexity and increases the probability of correct control at v_j. In particular,

$$\epsilon_j^o < \epsilon_j^p \quad \Longrightarrow \quad v_j : V_D \to V_O, \qquad \mathcal{C}_A(G') = \mathcal{C}_A(G) - \log|\mathcal{A}_j| < \mathcal{C}_A(G). \tag{9}$$

At the same time,

$$1 - \epsilon_j^o > 1 - \epsilon_j^p. \tag{10}$$

More generally, for a set of replaced nodes $J \subseteq V_D$, under the same conditional-independence assumption,

$$\epsilon_j^o < \epsilon_j^p \; \forall j \in J \quad \Longrightarrow \quad \prod_{j \in J}(1 - \epsilon_j^o) > \prod_{j \in J}(1 - \epsilon_j^p). \tag{11}$$

Proposition 4 gives the node-level allocation rule underlying PGAA: a procedural Decision Node should be converted into an Operator Node when prior-specified control is more reliable than model control, while model agency should be retained otherwise. The proof is provided in Appendix B.4.

4 MINEDOC: BENCHMARK CONSTRUCTION AND PGAA INSTANTIATION

Web multimodal document mining provides a representative setting for studying agency allocation. Industrial search systems have accumulated reliable procedures for candidate retrieval, filtering, ranking, and verification. However, these procedures must operate alongside instance-dependent judgments about textual relevance, visual identity, and evidence consistency. This combination makes the task well suited to examining the structure–agency trade-off. We therefore construct MineDoc as a benchmark suite and instantiate PGAA through an FPR–ReAct mechanism.

4.1 BENCHMARK CONSTRUCTION

MineDoc comprises two benchmarks for evaluating Web multimodal document mining on real-world business data and publicly sourced test sets. Both benchmarks require systems to construct reliable multimodal evidence documents for a given textual question and query image.

MineDoc-Private. MineDoc-Private is constructed from internal real-world business data and captures challenges encountered in industrial search, including hard negatives, inconsistencies across retrieval sources, and ambiguous image–text evidence. It evaluates document-mining reliability under practical business conditions.

MineDoc-Public. MineDoc-Public comprises two test sets constructed from public data to simulate business scenarios. They are derived from Encyclopedic-VQA (Mensink et al., 2023) examples associated with iNaturalist-21 (Van Horn et al., 2021) and Google Landmarks Dataset v2 (Weyand et al., 2020), respectively. Covering natural species and landmarks, these sets provide complementary settings for evaluating fine-grained visual identification and evidence-grounded document construction beyond the internal business data.

Both benchmarks follow the same task formulation and document-validity criteria, enabling consistent comparison of execution strategies across the evaluated settings.

4.2 PGAA INSTANTIATION: FPR–REACT

Connection to the allocation principle. Proposition 4 assigns a procedural decision to prior-specified control when the prior is more reliable than model-controlled execution. In our setting, procedural priors come from accumulated operational knowledge, while in their absence, execution traces can provide candidate structure. Appendix F provides an empirical illustration through

progressive graph pruning, while Appendix G gives the corresponding pseudocode for extracting candidate procedural backbones.

In real-world business practice, the established Filter–Produce–Reflect (FPR) procedure exhibits a low rate of procedural control errors, motivating its use as a reliable structural prior. The appropriate action after each pass, however, depends on instance-specific evidence and verification feedback and lacks an equally reliable fixed prescription. Guided by this allocation principle, we design the **FPR–ReAct** mechanism: the Human-Prior Decision Graph encodes the fixed FPR procedure as Operator Nodes and assigns the subsequent action selection to a model-controlled Decision Node. This allocation is based on operational experience rather than online estimation of the two error probabilities. Appendix Figure 3 illustrates the architecture.

Filter. Filter processes heterogeneous search results through a prescribed sequence of normalization, deduplication, relevance assessment, and visual entity consistency checks. These operations are implemented as Operator Nodes: their invocation and execution order are specified by the procedure, while their outputs depend on the current candidates. Model-based assessments, where used, execute a predefined task within this sequence.

Produce. Produce constructs multimodal documents from the retained candidates through a fixed sequence of evidence aggregation, caption generation, and document assembly. These operations are also implemented as Operator Nodes. Vision-language models generate evidence-grounded captions, but do not choose which operation executes next.

Reflect. Reflect follows a predefined verification procedure to assess evidence faithfulness, visual consistency, and query answerability. Its verification operations are implemented as Operator Nodes and produce feedback for the subsequent decision. Thus, model-generated verification judgments serve as outputs of prescribed operations, while control over the next action is assigned separately.

Post-FPR decision. After each FPR pass, a Decision Node uses the evidence and verification feedback to select the next action. The model may terminate execution when the documents are sufficient, or select an adaptive action such as query rewriting when further retrieval is needed. Following an adaptive action, returns to the prescribed FPR procedure. The decision to invoke rewriting is model-controlled, whereas the operations within each FPR pass remain structurally specified.

5 Experiments

5.1 Experimental Setup

All methods use Qwen3.5-122B-A10B-FP8 (Qwen Team, 2026) as the backbone model. We evaluate on MineDoc-Private and the two MineDoc-Public test sets derived from iNaturalist 2021 (Van Horn et al., 2021) and Google Landmarks Dataset v2 (Weyand et al., 2020). Comparisons use matched backbone models, available tools, initial inputs, and inference budget limits to examine the effect of execution control.

Baselines. We compare four execution strategies. **Fixed Workflow** follows the prescribed procedure without adaptive action selection. **ReAct** autonomously selects execution steps and tools. **ReAct + Prior** receives the FPR procedural knowledge through prompting while retaining control over execution. **FPR–ReAct**, our PGAA instantiation, structurally enforces FPR and assigns subsequent adaptive action selection to the model. These comparisons examine the contributions of adaptation, procedural knowledge, and structural enforcement.

Metrics. We report Pass@Doc as document-level precision: the fraction of returned documents satisfying all validity criteria, including evidence faithfulness, visual consistency, and query answerability. We use precision rather than accuracy because invalid candidate documents greatly outnumber valid ones, making accuracy misleadingly high for a system that returns nothing. We report Pass@Query as query-level accuracy under a strict all-or-nothing criterion: a query is correct only if all required documents are found and every returned document satisfies all validity criteria. Any missing required document or invalid returned document makes the query incorrect. For queries with no valid result, only an empty output is considered correct. We additionally measure estimated text-token usage, LLM calls, and embedding calls.

Table 1: Document-mining reliability on MineDoc. Higher values are better for both metrics. Values in parentheses indicate the total number of documents produced by each method on each test set. Detailed results using conventional evaluation metrics are provided in the appendix H.

Method	MineDoc-Private		MineDoc-Public			
			iNaturalist 2021		Google Landmarks Dataset v2	
	Pass@Doc	Pass@Query	Pass@Doc	Pass@Query	Pass@Doc	Pass@Query
Fixed Workflow	96.8% (252)	86.0%	49.8% (201)	68.8%	100.0% (130)	79.6%
ReAct	88.4% (121)	85.5%	61.2% (134)	65.6%	100.0% (70)	77.8%
ReAct + Prior	93.3% (119)	88.5%	**69.3%** (88)	71.8%	100.0% (67)	66.2%
FPR-ReAct (Ours)	**97.3% (263)**	**94.0%**	60.8% **(286)**	**76.2%**	100.0% (**151**)	**83.0%**

Table 2: Effect of control allocation on document-mining reliability. Intermediate settings fix the first N execution steps, with $N \in \{1, 3, 5\}$. Higher values are better for both metrics.

Setting	MineDoc-Private		iNaturalist 2021		Google Landmarks Dataset v2	
	Pass@Doc	Pass@Query	Pass@Doc	Pass@Query	Pass@Doc	Pass@Query
Free all (ReAct)	88.4%	85.5%	61.2%	65.6%	100.0%	77.8%
Fix first 1 step	90.2%	86.4%	69.3%	77.0%	100.0%	82.6%
Fix first 3 steps	90.2%	86.5%	**69.8%**	**77.1%**	100.0%	80.7%
Fix first 5 steps	90.1%	**87.3%**	**69.8%**	72.9%	100.0%	**89.9%**
Fix all (Fixed Workflow)	**96.8%**	86.0%	49.8%	68.8s%	100.0%	79.6%

5.2 Main Results

Table 1 shows that FPR–ReAct achieves the highest query-level reliability while producing the most documents across all test sets. A recurring failure mode of ReAct is premature termination after mining a single valid document, potentially leaving additional required evidence unretrieved. This helps explain why high document precision can coexist with limited output and lower query-level success. The advantage over ReAct + Prior further suggests that providing procedural knowledge through prompting is insufficient to ensure complete execution, supporting structural enforcement of reliable procedures alongside adaptive model control.

5.3 Effect of Control Allocation

We fix the first N execution steps according to the prescribed procedure, with $N \in \{1, 3, 5\}$, and then return control to ReAct. Subsequent actions are selected autonomously from the complete tool set. **Free all** uses ReAct throughout execution, while **Fix all** follows the complete Fixed Workflow. Table 2 shows that the partially fixed settings achieve higher query-level reliability than both unconstrained ReAct and Fixed Workflow. The gains vary with prefix length, potentially reflecting changes in the composition and completeness of the evidence produced downstream.

5.4 Inference Cost

We measure inference usage through the average number of text tokens, including input and output, and LLM calls per run. Table 3 compares these metrics across execution strategies.

The results distinguish the efficiency gains from procedural guidance and structural enforcement. ReAct + Prior reduces call frequency, but token usage changes little or decreases less substantially, suggesting that fewer invocations can involve more textual processing per call. FPR-ReAct reduces both metrics while producing more documents and achieving higher query-level reliability. This pattern is consistent with reducing repeated procedural decision-making while maintaining productive evidence processing. Compared with Fixed Workflow, FPR-ReAct spends additional inference on adaptation and achieves higher query-level success, supporting the allocation of model control to decisions where adaptation provides value.

Table 3: Average inference usage per run on each MineDoc test set, measured by text tokens (input and output) and LLM calls. Token counts are reported in thousands (k). Lower values indicate less inference usage.

Method	MineDoc-Private		MineDoc-Public			
			iNaturalist 2021		Google Landmarks Dataset v2	
	Text Tokens	LLM Calls	Text Tokens	LLM Calls	Text Tokens	LLM Calls
Fixed Workflow	41.55k	20.81	37.44k	15.62	20.76k	12.31
ReAct	97.48k	50.84	147.59k	80.52	139.82k	102.15
ReAct + Prior	98.05k	47.83	139.82k	51.57	81.67k	50.09
FPR-ReAct (Ours)	55.19k	28.20	50.12k	27.42	32.34k	23.86

6 Related Work

LLM agents and tool execution. ReAct (Yao et al., 2023) interleaves reasoning and actions, while Reflexion (Shinn et al., 2023) and CRITIC (Gou et al., 2024) use reflective memory and tool-based feedback, respectively, to improve subsequent outputs. ReWOO (Xu et al., 2023) separates reasoning from tool observations to reduce repeated prompting, and LLMCompiler (Kim et al., 2023) orchestrates parallel function calls. AgentBench (Liu et al., 2024) identifies persistent difficulties in reasoning, decision-making, and instruction following. PGAA examines which execution decisions should remain model-controlled when reliable procedural knowledge is available.

Workflow structure and procedural knowledge. AVIS (Hu et al., 2023) uses human-derived transition graphs to constrain visual information seeking, while FlowAgent (Shi et al., 2025) combines procedural compliance with adaptive execution. FlowBench (Xiao et al., 2024) evaluates planning with different workflow representations. Agent Workflow Memory (Wang et al., 2024) and AutoManual (Chen et al., 2024) acquire reusable routines and instruction rules from experience. DSPy (Khattab et al., 2024) optimizes modular language-model programs, while AFlow (Zhang et al., 2025) and ADAS (Hu et al., 2025a) search over workflow and agent implementations. These approaches establish the value of structured execution and reusable knowledge. PGAA formalizes control allocation through prior-specified Operator Nodes and model-controlled Decision Nodes, analyzing the competing risks of incorrect model choices and restrictive structures. Our experiments examine prompting and programmatic enforcement of the same procedural priors.

Multimodal retrieval and verification. WebQA (Chang et al., 2022) and MuRAG (Chen et al., 2022) study evidence retrieval and reasoning across images and text. Self-Refine (Madaan et al., 2023) iteratively revises outputs using self-feedback; Self-RAG (Asai et al., 2024) and Corrective RAG (Yan et al., 2024) connect evidence assessment with retrieval decisions. FaithScore (Jing et al., 2024) evaluates visual faithfulness through atomic factual claims. MineDoc combines retrieval, document construction, and verification to study how prescribed procedures and adaptive actions jointly affect multimodal evidence reliability.

7 Conclusion and Limitations

We propose Prior-Guided Agency Allocation (PGAA) to allocate control between procedural priors and model judgment, and construct MineDoc to evaluate its FPR–ReAct instantiation. Our results support selective agency as a practical design principle: reliable procedures provide a stable execution backbone, while model-controlled adaptation addresses cases requiring further evidence or revised actions. This combination improves query-level reliability while reducing estimated text-token usage and LLM calls relative to autonomous execution. The appropriate allocation depends on the reliability of the available priors and the need for instance-specific judgment.

Our pruning analysis currently focuses on MineDoc, where it recovers a procedural backbone consistent with established human priors. Future work will extend this validation to other tasks and domains, particularly settings without established procedural knowledge. We will further investigate automated methods for discovering and validating priors from execution traces and allocating model control according to their reliability.

AI USE STATEMENT

During the preparation of this work, the authors used GPT-6-Astra and Claude-Opus-5 in the following manner:

- **Language Editing:** GPT-6-Astra was used for translation, language polishing, and organization of the manuscript, including improving grammar, clarity, and readability.
- **Code Assistance:** Claude-Opus-5 was used to assist with code generation and implementation for this work.
- **Ideation/Analysis:** Claude-Opus-5 was used to assist with the visualization of research results.

The authors have reviewed all AI-assisted outputs, including text, code, and visualizations, and take full and ultimate responsibility for the integrity and accuracy of the content in this submission.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*, volume 2024, pp. 9112–9141, 2024.

Yingshan Chang, Mridu Narang, Hisami Suzuki, Guihong Cao, Jianfeng Gao, and Yonatan Bisk. Webqa: Multihop and multimodal qa. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16495–16504, 2022.

Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. Automanual: Constructing instruction manuals by llm agents via interactive environmental learning. *Advances in Neural Information Processing Systems*, 37:589–631, 2024.

Wenhu Chen, Hexiang Hu, Xi Chen, Pat Verga, and William Cohen. Murag: Multimodal retrieval-augmented generator for open question answering over images and text. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 5558–5570, 2022.

Yang Chen, Hexiang Hu, Yi Luan, Haitian Sun, Soravit Changpinyo, Alan Ritter, and Ming-Wei Chang. Can pre-trained vision and language models answer visual information-seeking questions? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14948–14968, 2023.

Guanting Dong, Jiajie Jin, Xiaoxi Li, Yutao Zhu, Zhicheng Dou, and Ji-Rong Wen. Rag-critic: Leveraging automated critic-guided agentic workflow for retrieval augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3551–3578, 2025.

Jiri Fridrich and Miroslav Goljan. Robust hash functions for digital watermarking. In *Proceedings International Conference on Information Technology: Coding and Computing (Cat. No. PR00540)*, pp. 178–183. IEEE, 2000.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Nan Duan, Weizhu Chen, et al. Critic: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, volume 2024, pp. 57734–57811, 2024.

Jiafeng Guo, Yinqiong Cai, Yixing Fan, Fei Sun, Ruqing Zhang, and Xueqi Cheng. Semantic models for the first-stage retrieval: A comprehensive review. *ACM Transactions on Information Systems (TOIS)*, 40(4):1–42, 2022.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, volume 2025, pp. 21344–21377, 2025a.

Yiheng Hu, Xiaoyang Wang, Qing Liu, Xiwei Xu, Qian Fu, Wenjie Zhang, and Liming Zhu. Mmapg: A training-free framework for multimodal multi-hop question answering via adaptive planning graphs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 14195–14211, 2025b.

Ziniu Hu, Ahmet Iscen, Chen Sun, Kai-Wei Chang, Yizhou Sun, David Ross, Cordelia Schmid, and Alireza Fathi. Avis: Autonomous visual information seeking with large language model agent. *Advances in Neural Information Processing Systems*, 36:867–878, 2023.

Liqiang Jing, Ruosen Li, Yunmo Chen, and Xinya Du. Faithscore: Fine-grained evaluations of hallucinations in large vision-language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 5042–5063, 2024.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Saiful Haq, Ashutosh Sharma, Thomas Joshi, Hanna Moazam, Heather Miller, et al. Dspy: Compiling declarative language model calls into state-of-the-art pipelines. In *International Conference on Learning Representations*, volume 2024, pp. 54928–54958, 2024.

Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W Mahoney, Kurt Keutzer, and Amir Gholami. An llm compiler for parallel function calling. *arXiv preprint arXiv:2312.04511*, 2023.

Mingxin Li, Yanzhao Zhang, Dingkun Long, Keqin Chen, Sibo Song, Shuai Bai, Zhibo Yang, Pengjun Xie, An Yang, Dayiheng Liu, et al. Qwen3-vl-embedding and qwen3-vl-reranker: A unified framework for state-of-the-art multimodal retrieval and ranking. *arXiv preprint arXiv:2601.04720*, 2026.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Thomas Mensink, Jasper Uijlings, Lluis Castrejon, Arushi Goel, Felipe Cadar, Howard Zhou, Fei Sha, André Araujo, and Vittorio Ferrari. Encyclopedic vqa: Visual questions about detailed properties of fine-grained categories. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3090–3101. IEEE, 2023.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424*, 2019.

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

Yuchen Shi, Siqi Cai, Zihan Xu, Yuei Qin, Gang Li, Hang Shao, Jiawei Chen, Deqing Yang, Ke Li, and Xing Sun. Flowagent: Achieving compliance and flexibility for workflow agents. *arXiv preprint arXiv:2502.14345*, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Grant Van Horn, Elijah Cole, Sara Beery, Kimberly Wilber, Serge Belongie, and Oisin Mac Aodha. Benchmarking representation learning for natural world image collections. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12884–12893, 2021.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.

Tobias Weyand, Andre Araujo, Bingyi Cao, and Jack Sim. Google landmarks dataset v2–a large-scale benchmark for instance-level recognition and retrieval. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2572–2581. IEEE, 2020.

Yin Wu, Quanyu Long, Jing Li, Jianfei Yu, and Wenya Wang. Visual-rag: Benchmarking text-to-image retrieval augmented generation for visual knowledge intensive queries. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3473–3480, 2026.

Ruixuan Xiao, Wentao Ma, Ke Wang, Yuchuan Wu, Junbo Zhao, Haobo Wang, Fei Huang, and Yongbin Li. Flowbench: Revisiting and benchmarking workflow-guided planning for llm-based agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 10883–10900, 2024.

Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. Rewoo: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint arXiv:2305.18323*, 2023.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*, 2024.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

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

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*, volume 2025, pp. 9965–10017, 2025.

Christoph Zauner. Implementation and benchmarking of perceptual image hash functions. 2010.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. In *International Conference on Learning Representations*, volume 2025, pp. 34040–34077, 2025.

A Proof of the Agency Complexity Ordering

For each Operator Node $v \in V_O$, the action is uniquely determined by the structural prior. We therefore define

$$|\mathcal{A}_v| = 1, \tag{12}$$

which gives

$$\log |\mathcal{A}_v| = \log 1 = 0. \tag{13}$$

Hence,

$$\mathcal{C}_A(G) = \sum_{v \in V} \log |\mathcal{A}_v| = \sum_{v \in V_D} \log |\mathcal{A}_v|. \tag{14}$$

For a fixed workflow G_{Fixed}, all execution decisions are determined by the structural prior, so there are no model-controlled choices. Therefore,

$$\mathcal{C}_A(G_{\text{Fixed}}) = 0. \tag{15}$$

Now consider an arbitrary execution graph G_{any}. For every Decision Node $v \in V_D$,

$$|\mathcal{A}_v| \geq 1, \tag{16}$$

and thus

$$\log |\mathcal{A}_v| \geq 0. \tag{17}$$

Since Agency Complexity is a sum of non-negative terms,

$$\mathcal{C}_A(G_{\text{any}}) = \sum_{v \in V_D} \log |\mathcal{A}_v| \geq 0. \tag{18}$$

Therefore,

$$\mathcal{C}_A(G_{\text{Fixed}}) = 0 \leq \mathcal{C}_A(G_{\text{any}}). \tag{19}$$

Finally, let G_{Auto} denote the fully autonomous formulation, in which all available execution decisions remain under model control. By construction, G_{any} is obtained by fixing some subset of these decisions through structural priors, while leaving the remaining decisions under model control. Consequently, the effective decision space exposed by G_{any} cannot exceed that of G_{Auto}, yielding

$$\mathcal{C}_A(G_{\text{any}}) \leq \mathcal{C}_A(G_{\text{Auto}}). \tag{20}$$

Equality holds when the structural constraints in G_{any} do not reduce the effective model-controlled decision space. The inequality becomes strict whenever at least one decision with more than one admissible action is removed from model control.

Combining the above results gives

$$\boxed{\mathcal{C}_A(G_{\text{Fixed}}) = 0 \leq \mathcal{C}_A(G_{\text{any}}) \leq \mathcal{C}_A(G_{\text{Auto}})}. \tag{21}$$

B Proofs for Theoretical Analysis

B.1 Proof of Proposition 1

Proposition 1 (Decision-error accumulation). Consider an execution graph G_K containing K model-controlled procedural Decision Nodes. Let G_{K+1} be obtained by adding another model-controlled procedural node v_{K+1} such that $|\mathcal{A}_{K+1}| > 1$ and $\epsilon^p_{K+1} > 0$. Then

$$\mathcal{C}_A(G_{K+1}) > \mathcal{C}_A(G_K)$$

and

$$e(G_{K+1}) > e(G_K).$$

Proof. By the definition of Agency Complexity,

$$\mathcal{C}_A(G) = \sum_{v \in V_D} \log |\mathcal{A}_v|.$$

Since G_{K+1} contains all Decision Nodes in G_K together with the additional model-controlled node v_{K+1}, we have

$$\mathcal{C}_A(G_{K+1}) = \mathcal{C}_A(G_K) + \log |\mathcal{A}_{K+1}|. \tag{22}$$

Because $|\mathcal{A}_{K+1}| > 1$,

$$\log |\mathcal{A}_{K+1}| > 0,$$

and therefore

$$\mathcal{C}_A(G_{K+1}) > \mathcal{C}_A(G_K).$$

We next consider decision error. Under the conditional-independence assumption, the probability that no model-controlled procedural error occurs in G_K is

$$\prod_{j=1}^{K} (1 - \epsilon_j^p),$$

so the probability that at least one such error occurs is

$$e(G_K) = 1 - \prod_{j=1}^{K} (1 - \epsilon_j^p).$$

After adding node v_{K+1}, the corresponding error probability becomes

$$e(G_{K+1}) = 1 - \prod_{j=1}^{K+1} (1 - \epsilon_j^p) \tag{23}$$

$$= 1 - \left[\prod_{j=1}^{K} (1 - \epsilon_j^p) \right] (1 - \epsilon_{K+1}^p). \tag{24}$$

Subtracting $e(G_K)$ yields

$$e(G_{K+1}) - e(G_K) = \left[1 - \prod_{j=1}^{K} (1 - \epsilon_j^p)(1 - \epsilon_{K+1}^p) \right] - \left[1 - \prod_{j=1}^{K} (1 - \epsilon_j^p) \right] \tag{25}$$

$$= \prod_{j=1}^{K} (1 - \epsilon_j^p) - \prod_{j=1}^{K} (1 - \epsilon_j^p)(1 - \epsilon_{K+1}^p) \tag{26}$$

$$= \epsilon_{K+1}^p \prod_{j=1}^{K} (1 - \epsilon_j^p). \tag{27}$$

Since $\epsilon_j^p \in (0, 1)$ for all $j \leq K + 1$, each factor $1 - \epsilon_j^p$ is strictly positive. Hence

$$\epsilon_{K+1}^p \prod_{j=1}^{K} (1 - \epsilon_j^p) > 0.$$

Therefore,

$$e(G_{K+1}) > e(G_K).$$

Thus, adding a nontrivial model-controlled procedural Decision Node strictly increases both Agency Complexity and the probability of at least one model-induced control error. □

B.2 Proof of Proposition 2

Proposition 2 (Structural-bias bound). For any model policy π_θ, the success probability under graph G is upper-bounded by the probability that the graph admits at least one successful trajectory. Moreover, if two graphs G_1 and G_2 satisfy $\mathcal{T}_{G_1}(x) \subseteq \mathcal{T}_{G_2}(x)$ for all x, then $\delta(G_2) \leq \delta(G_1)$.

Proof. For an input x, define the structural-feasibility event

$$F_G(x) = \{\mathcal{T}^*(x) \cap \mathcal{T}_G(x) \neq \varnothing\}.$$

By the definition of Structural Bias Risk,

$$\Pr_{x\sim\mathcal{D}}[F_G(x)] = 1 - \delta(G).$$

Successful execution under graph G requires that the graph admit at least one successful trajectory. Therefore, for any model policy π_θ, the success event is contained in $F_G(x)$, and hence

$$P_{\text{success}}(G, \pi_\theta) \leq \Pr_{x\sim\mathcal{D}}[F_G(x)] = 1 - \delta(G).$$

We next consider two graphs G_1 and G_2 satisfying $\mathcal{T}_{G_1}(x) \subseteq \mathcal{T}_{G_2}(x)$ for all x. If

$$\mathcal{T}^*(x) \cap \mathcal{T}_{G_2}(x) = \varnothing,$$

then, since $\mathcal{T}_{G_1}(x) \subseteq \mathcal{T}_{G_2}(x)$, it must also hold that

$$\mathcal{T}^*(x) \cap \mathcal{T}_{G_1}(x) = \varnothing.$$

Therefore,

$$\{\mathcal{T}^*(x) \cap \mathcal{T}_{G_2}(x) = \varnothing\} \subseteq \{\mathcal{T}^*(x) \cap \mathcal{T}_{G_1}(x) = \varnothing\}.$$

Taking probabilities over $x \sim \mathcal{D}$ gives

$$\delta(G_2) \leq \delta(G_1).$$

Thus, Structural Bias Risk imposes an upper bound on achievable success, and enlarging the admissible trajectory set cannot increase Structural Bias Risk. □

B.3 Proof of Proposition 3

Proposition 3 (Structure–agency risk decomposition). Let $e(G)$ and $\delta(G)$ denote the Decision Error Risk and Structural Bias Risk of graph G, respectively. Under the simplifying assumption that structural feasibility and error-free model control are independent, the end-to-end success probability is

$$P_{\text{success}}(G) = \big[1 - \delta(G)\big]\big[1 - e(G)\big],$$

and the corresponding failure risk is

$$R(G) = \delta(G) + e(G) - \delta(G)e(G).$$

Proof. Let F_G denote the event that graph G admits at least one successful trajectory. By the definition of Structural Bias Risk,

$$\Pr(F_G) = 1 - \delta(G).$$

Let C_G denote the event that no model-induced decision error occurs during execution. By the definition of Decision Error Risk,

$$\Pr(C_G) = 1 - e(G).$$

Successful execution requires both structural feasibility and correct model control. Therefore,

$$P_{\text{success}}(G) = \Pr(F_G \cap C_G).$$

Under the simplifying independence assumption between F_G and C_G,

$$P_{\text{success}}(G) = \Pr(F_G)\Pr(C_G) \tag{28}$$

$$= \big[1 - \delta(G)\big]\big[1 - e(G)\big]. \tag{29}$$

The overall failure risk is therefore

$$R(G) = 1 - P_{\text{success}}(G) \tag{30}$$
$$= 1 - \big[1 - \delta(G)\big]\big[1 - e(G)\big] \tag{31}$$
$$= \delta(G) + e(G) - \delta(G)e(G). \tag{32}$$

Thus, end-to-end execution risk is jointly determined by Decision Error Risk and Structural Bias Risk. Increasing model control may increase the former, whereas imposing stronger structural constraints may increase the latter, giving rise to the structure–agency trade-off.

□

B.4 Proof of Proposition 4

Proposition 4 (Prior-guided agency allocation). Consider a procedural Decision Node $v_j \in V_D$. If $|\mathcal{A}_j| > 1$ and $\epsilon_j^o < \epsilon_j^p$, then replacing model control with a prior-specified Operator Node reduces Agency Complexity and increases the probability of correct control at v_j. More generally, under the conditional-independence assumption, replacing a set of nodes $J \subseteq V_D$ satisfying $\epsilon_j^o < \epsilon_j^p$ for all $j \in J$ increases the probability of correct control across those nodes.

Proof. By the definition of Agency Complexity,

$$\mathcal{C}_A(G) = \sum_{v \in V_D} \log |\mathcal{A}_v|.$$

Replacing Decision Node v_j with an Operator Node removes its contribution $\log |\mathcal{A}_j|$ from the model-controlled decision space. Therefore,

$$\mathcal{C}_A(G') = \mathcal{C}_A(G) - \log |\mathcal{A}_j|.$$

Since $|\mathcal{A}_j| > 1$,

$$\log |\mathcal{A}_j| > 0,$$

and hence

$$\mathcal{C}_A(G') < \mathcal{C}_A(G).$$

We next compare the reliability of the two control mechanisms. Under model control, the probability of a correct control outcome at v_j is

$$1 - \epsilon_j^p,$$

whereas under the prior-specified Operator Node it is

$$1 - \epsilon_j^o.$$

Since $\epsilon_j^o < \epsilon_j^p$, it follows that

$$1 - \epsilon_j^o > 1 - \epsilon_j^p.$$

Thus,

$$\epsilon_j^o < \epsilon_j^p \quad \Longrightarrow \quad v_j : V_D \to V_O$$

reduces Agency Complexity while increasing the probability of correct control at node v_j.

We finally consider a set of replaced nodes $J \subseteq V_D$. Under the conditional-independence assumption, the probability of correct control across these nodes under model control is

$$P_{\text{model}} = \prod_{j \in J} (1 - \epsilon_j^p),$$

while under prior-specified control it is

$$P_{\text{prior}} = \prod_{j \in J} (1 - \epsilon_j^o).$$

Because $\epsilon_j^o < \epsilon_j^p$ for every $j \in J$, we have

$$1 - \epsilon_j^o > 1 - \epsilon_j^p, \qquad \forall j \in J.$$

Multiplying these positive inequalities gives

$$\prod_{j \in J}(1 - \epsilon_j^o) > \prod_{j \in J}(1 - \epsilon_j^p),$$

and therefore

$$P_{\text{prior}} > P_{\text{model}}.$$

Thus, when prior-specified control is more reliable than model control, converting the corresponding Decision Nodes into Operator Nodes reduces Agency Complexity while increasing control reliability. □

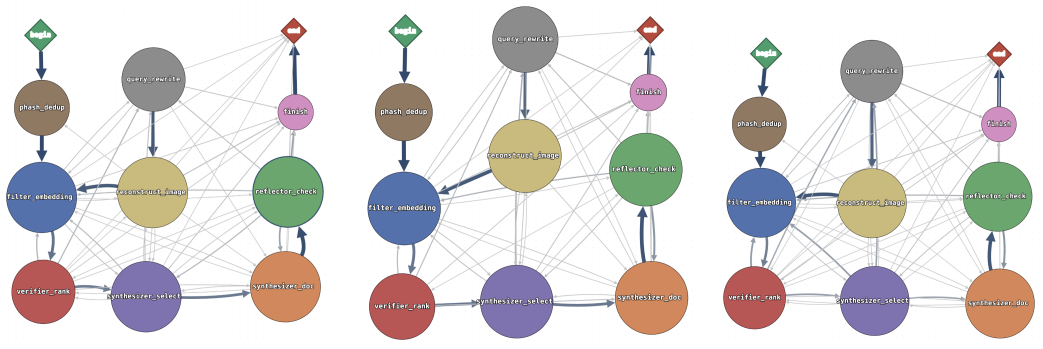
C Execution Routing Visualization

Figure 2 visualizes PGAA execution routes on MineDoc-Private and the MineDoc-Public test sets derived from iNaturalist 2021 and Google Landmarks Dataset v2. We aggregate execution traces into transition graphs, where nodes represent execution operations or decision points and directed edges represent observed transitions. Operator Nodes and Decision Nodes are visually distinguished, and edge widths indicate transition frequencies normalized within each test set. These visualizations show how the prescribed FPR procedure interacts with model-controlled actions after each pass, enabling comparison of recurring execution paths and adaptive behavior across datasets.

D Execution Routing Visualization

Figure 2 presents aggregated ReAct execution routes on MineDoc-Private and the two MineDoc-Public test sets derived from iNaturalist 2021 and Google Landmarks Dataset v2. We aggregate execution traces into directed transition graphs, where nodes represent execution operations and edges represent observed transitions between them. All three panels use the same node layout to facilitate comparison across datasets.

Despite differences in data distributions, the aggregated ReAct traces exhibit a consistent dominant procedural backbone. **The bold edges highlight this shared backbone, which aligns with the Filter–Produce–Reflect (FPR) workflow constructed from accumulated human procedural knowledge.** Other observed transitions capture variations around this common procedure. Since these traces are collected from ReAct rather than from the structurally constrained PGAA system, the shared backbone reflects recurring autonomous execution behavior rather than enforced FPR routing. This observation motivates encoding the recurring procedure explicitly while preserving model control over instance-dependent adaptation; it does not, by itself, establish that the dominant route is optimal for every input.



(a) MineDoc-Private (b) iNaturalist 2021 (c) Google Landmarks Dataset v2

Figure 2: Aggregated ReAct execution routes on MineDoc-Private and the two MineDoc-Public test sets. Nodes represent execution operations, and directed edges represent observed transitions. **Bold edges highlight the shared dominant procedural backbone, which aligns with the human-designed FPR workflow across different data distributions.** The remaining edges show other observed transitions. All panels use the same node layout for comparison.

E MINEDOC ARCHITECTURE

Figure 3 presents the detailed FPR–ReAct architecture used in MineDoc.

Given a user query and query image, MineDoc first retrieves candidate Web evidence and processes it through a fixed three-stage FPR pipeline. In **Filter**, retrieved candidates are deduplicated, recalled through complementary visual and vision-language retrieval paths, and further checked for content consistency. In **Produce**, the retained candidates are organized into evidence units, selected through a prescribed procedure, and converted into multimodal documents with image captions and supporting textual evidence. In **Reflect**, the resulting documents are verified along three dimensions: entity consistency, image–text faithfulness, and query answerability. Documents that pass these checks are retained as the final evidence set.

When reflection indicates that the current evidence is insufficient, control is transferred to the adaptive ReAct branch. The model reasons over the verification feedback and may rewrite the query, edit the query image, or reconstruct the retrieval input before launching another retrieval-and-FPR pass. This design instantiates PGAA by encoding Filter, Produce, and Reflect as structurally prescribed Operator Nodes, while assigning post-verification adaptation to a model-controlled Decision Node.

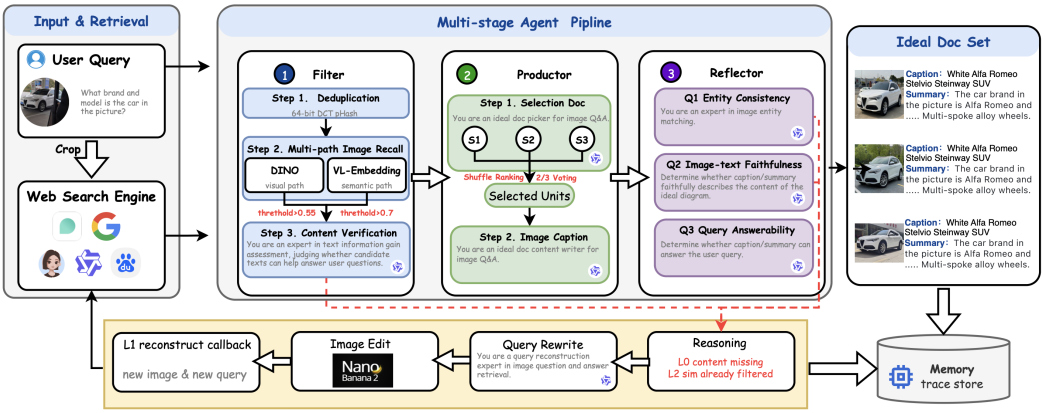


Figure 3: Overview of the MineDoc FPR–ReAct architecture. Given a user query and query image, the system first retrieves candidate Web evidence and then executes the prescribed Filter–Produce–Reflect pipeline. Filter performs deduplication, multi-path image recall, and content verification; Produce selects evidence units and constructs multimodal documents with image captions; Reflect verifies entity consistency, image–text faithfulness, and query answerability. When the current evidence is insufficient, ReAct uses the verification feedback to trigger adaptive actions such as query rewriting, image editing, or retrieval-input reconstruction before the next FPR pass.

F PROGRESSIVE GRAPH PRUNING ON MINEDOC-PUBLIC

Figure 4 visualizes the progressive pruning process on MineDoc-Public. Starting from the original execution graph, we progressively remove low-frequency transitions in increments of 10. As pruning becomes more aggressive, infrequent branches and alternative transitions are gradually eliminated, exposing the dominant procedural structure of the graph.

A particularly informative point occurs at pruning level 30. At this stage, only a single connected path from `Begin` to `End` remains. Notably, this surviving path coincides with the engineered FPR workflow constructed from accumulated human prior knowledge. Further pruning removes transitions from this backbone and eventually compromises end-to-end connectivity. This progression provides an empirical illustration of the structure–agency trade-off: moderate pruning suppresses low-frequency model-controlled variations and reveals the stable human-prior procedure, whereas excessive pruning removes transitions required for complete execution.

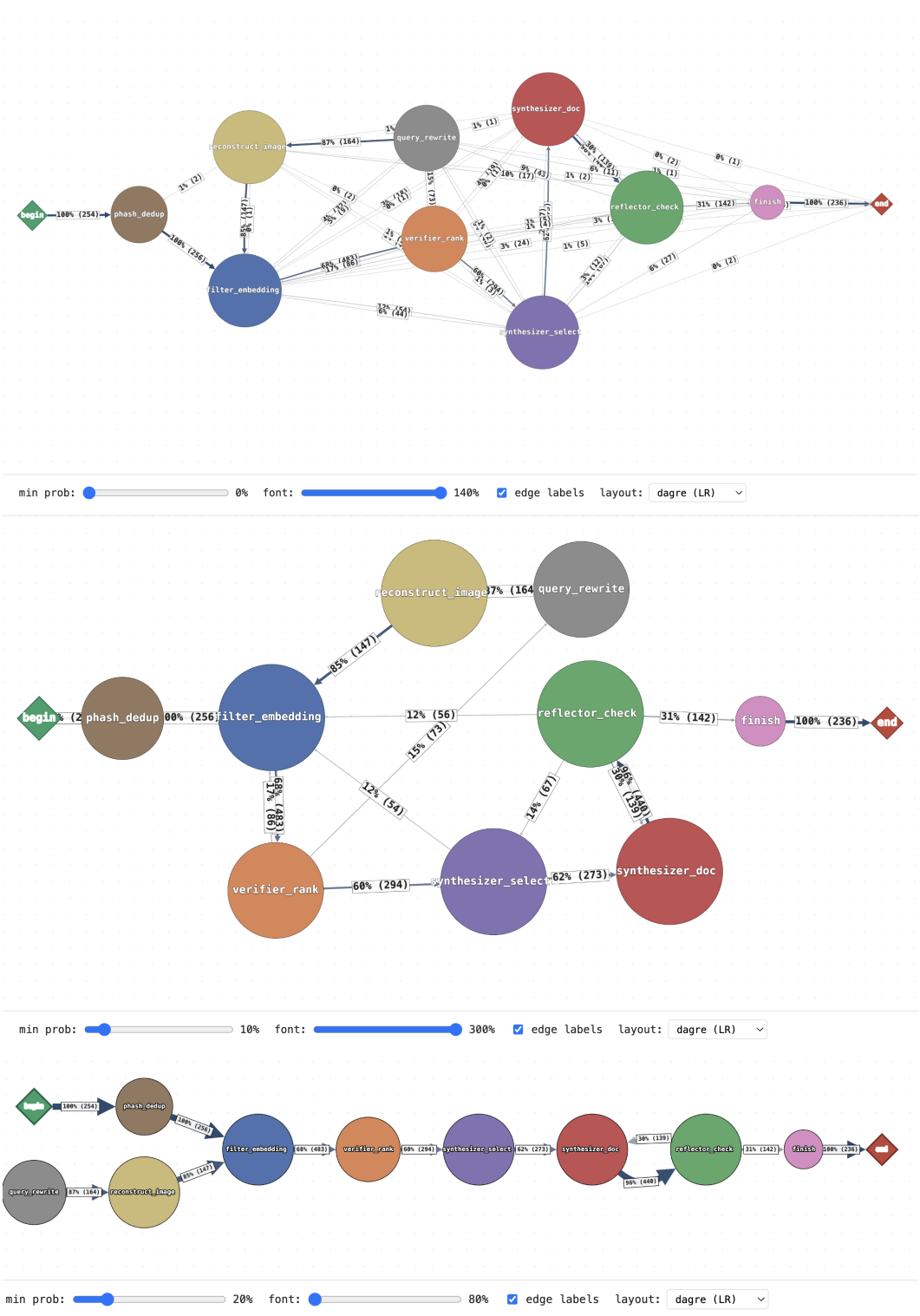


Figure 4: Progressive pruning of the execution graph on MineDoc-Public. From top to bottom, the pruning level increases from 0 to 20 in increments of 10. Low-frequency transitions are progressively removed, causing the graph to contract toward its dominant procedural backbone. At pruning level 30, only one connected path from `Begin` to `End` remains; notably, this path exactly matches the engineered FPR workflow derived from accumulated human prior knowledge. Further pruning begins to remove transitions from this backbone.

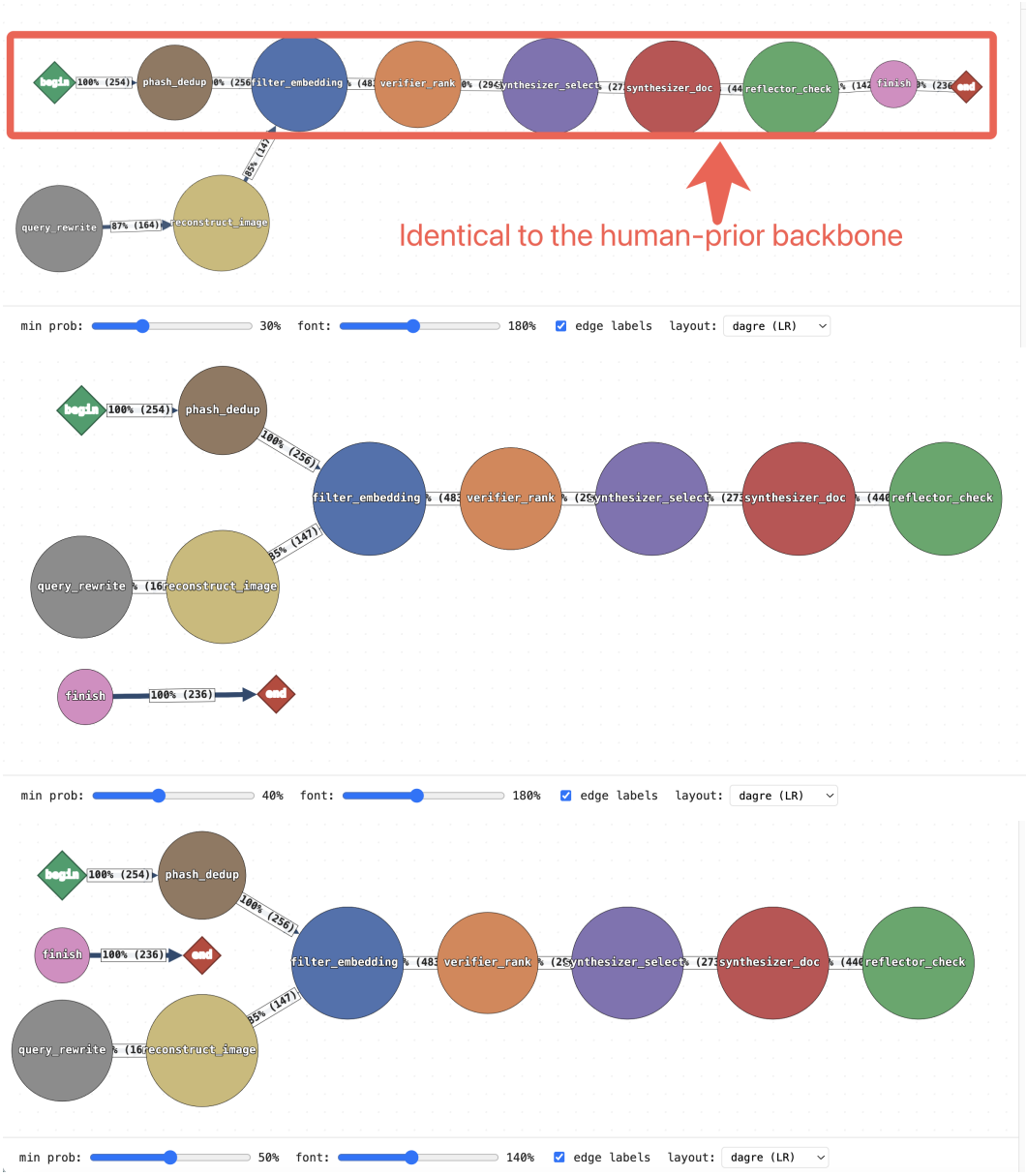


Figure 4: Progressive pruning of the execution graph on MineDoc-Public (continued). From top to bottom, the pruning level increases from 30 to 50 in increments of 10. Low-frequency transitions are progressively removed, causing the graph to contract toward its dominant procedural backbone. At pruning level 30, only one connected path from `Begin` to `End` remains; notably, this path exactly matches the engineered FPR workflow derived from accumulated human prior knowledge. Further pruning begins to remove transitions from this backbone.

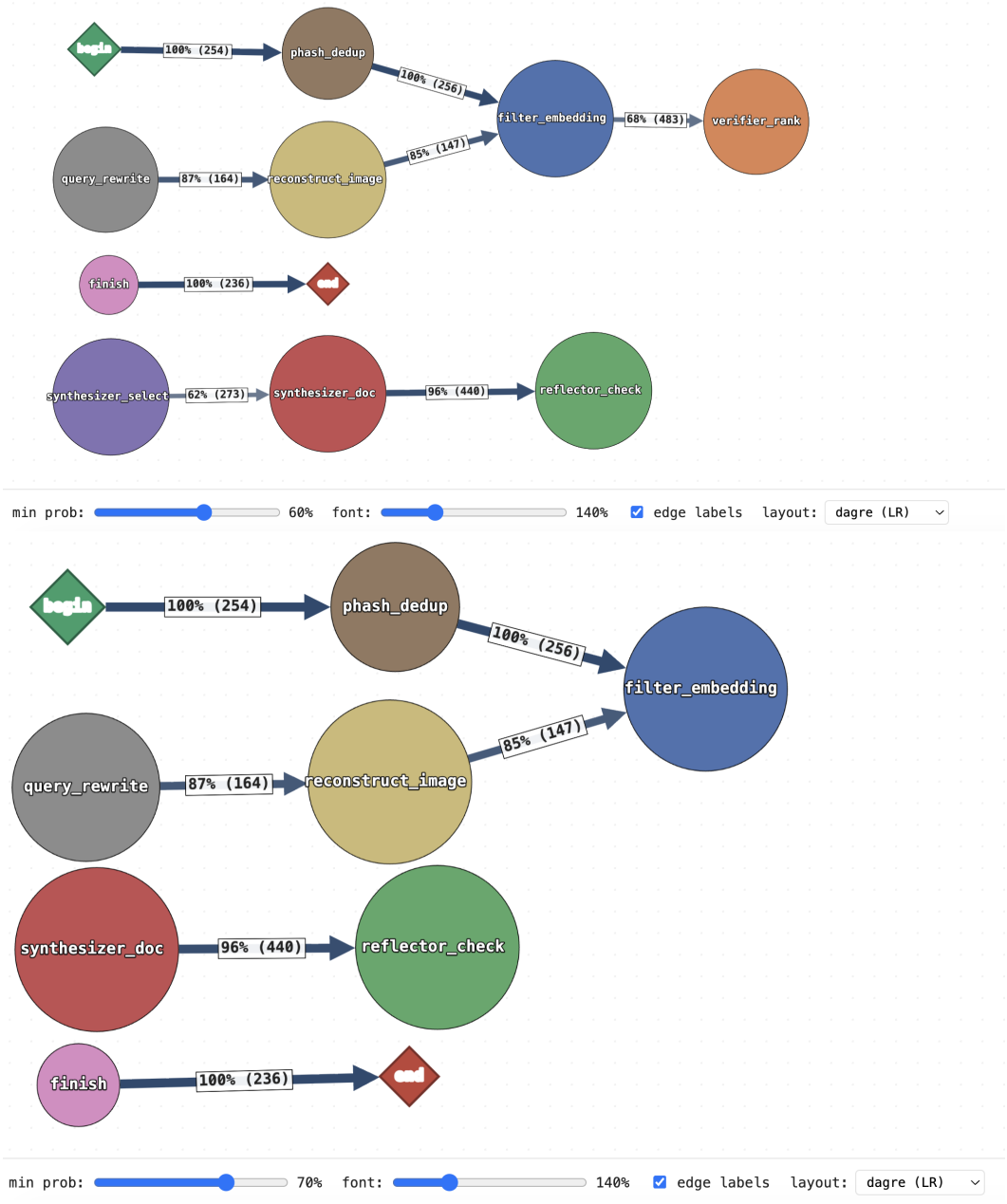


Figure 4: Progressive pruning of the execution graph on MineDoc-Public (continued). From top to bottom, the pruning level increases from 60 to 70 in increments of 10. Beyond the critical pruning level, transitions belonging to the dominant procedural backbone are progressively removed, eventually disrupting complete connectivity between `Begin` and `End`.

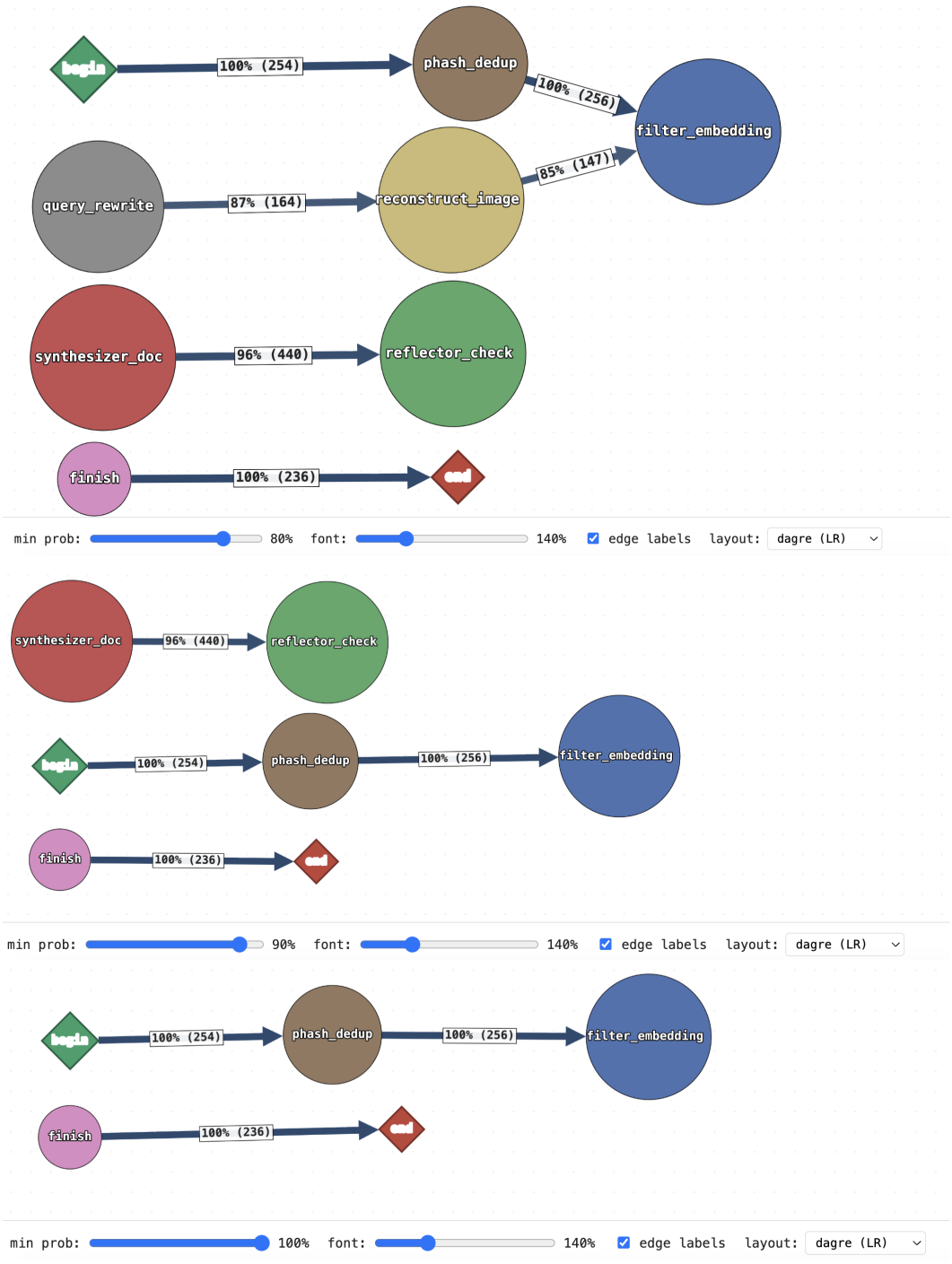


Figure 4: Progressive pruning of the execution graph on MineDoc-Public (continued). From top to bottom, the pruning level increases from 80 to 100 in increments of 10. Beyond the critical pruning level, transitions belonging to the dominant procedural backbone are progressively removed, eventually disrupting complete connectivity between `Begin` and `End`.

G TRACE-DERIVED CANDIDATE PROCEDURAL PRIORS

When reliable procedural priors are not explicitly available, execution traces can provide candidate procedural structure. We aggregate autonomous execution traces into an execution graph and progressively retain stable, frequently traversed transitions to identify a candidate procedural backbone. This procedure is intended as a practical prior-proposal mechanism rather than a fully automatic prior-validation method.

Algorithm 1 summarizes this procedure.

Algorithm 1 Progressive Extraction of a Trace-Derived Candidate Procedural Backbone

Require: Autonomous execution traces $\mathcal{H}$, pruning step $\Delta\tau$
Ensure: Candidate procedural backbone G_{prior}
Aggregate traces $\mathcal{H}$ into a directed execution graph $G = (V, E)$
Normalize traversal frequency $f(e) \in [0, 1]$ for each transition $e \in E$
Initialize $G_{\text{prior}} \leftarrow G$
Initialize pruning threshold $\tau \leftarrow 0$
while $\tau \leq 1$ **do**
    $\tau \leftarrow \tau + \Delta\tau$
    $G' \leftarrow G_{\text{prior}}$
    for each transition $e \in E(G')$ with $f(e) < \tau$ **do**
        Remove e from G'
    end for
    if G' preserves a path from Begin to End **then**
        $G_{\text{prior}} \leftarrow G'$
    else
        break
    end if
end while
return G_{prior}

H DETAILED RESULTS WITH CONVENTIONAL EVALUATION METRICS

Table 4 reports detailed results on the iNaturalist and Landmarks test sets of MineDoc using four conventional evaluation metrics: precision, recall, F1, and accuracy. Results are presented at both the query and document levels, complementing the document-mining reliability results in the main text.

Table 4: Conventional evaluation results on MineDoc at the query and document levels. All values are percentages, and higher values are better for all metrics. Bold indicates the best result for each metric within each test set and evaluation level, including ties.

	Query Level				Document Level			
Method	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy
iNaturalist 2021								
`fixed_workflow`	58.14	29.41	39.06	68.80	49.75	25.25	33.50	97.68
`pgaa`	**68.61**	**55.29**	**61.24**	**76.20**	60.84	**25.59**	**36.02**	**98.30**
`react`	49.39	47.65	48.50	65.60	61.19	12.06	20.15	98.00
`react_prior`	59.86	51.76	55.52	71.80	**69.29**	12.94	21.81	98.12
Google Landmarks Dataset v2								
`fixed_workflow`	**100.00**	34.62	51.43	79.60	**100.00**	**27.03**	**42.56**	97.75
`pgaa`	**100.00**	**45.51**	**62.56**	**83.00**	**100.00**	25.29	40.37	**98.78**
`react`	74.73	43.59	55.06	77.80	**100.00**	11.73	20.99	98.45
`react_prior`	40.1%	39.1%	39.6%	62.8%	100.00%	11.1%	19.9%	98.1%

I REACT PLANNER PROMPTS AND MODEL OUTPUTS

A detailed prompt example is provided below. Non-English content has been translated into English. URLs, identifiers, and source-specific candidate data have been replaced with "...", while the original tool-call structure and action sequence have been preserved.

I.1 STEP 1: PHASH_DEDUP

Prompt

```
## Current context
request_id      : ...
query_text      : What model is the motorcycle in the image?
query_image_url : ...

## Progress so far
tool_calls_used     : 0 / 30
pool_units_total    : 26
filtered_units      : 0
ranked_units        : 0
selected_units      : 0
docs_generated      : 0
docs_passed         : 0
phash_done          : False

## Last observation
(none | first turn)

## Recent scratchpad (most recent last)
(empty)

Now emit the next JSON action.
```

Model raw output

```
{
  "thought": "This is the first turn. The goal is to identify the motorcycle model in the
query image. I have a pool of 26 candidate image-text units. To ensure efficiency and avoid
processing duplicates, the protocol recommends calling phash_dedup first before any
embedding-based retrieval. This will clean the candidate pool.",
  "action": "phash_dedup",
  "args": {
    "threshold": 5,
    "workers": 16,
    "timeout": 15
  }
}
```

I.2 STEP 2: FILTER_EMBEDDING

Prompt

```
## Current context
request_id      : ...
query_text      : What model is the motorcycle in the image?
query_image_url : ...

## Progress so far
tool_calls_used     : 1 / 30
pool_units_total    : 24
filtered_units      : 0
ranked_units        : 0
selected_units      : 0
docs_generated      : 0
docs_passed         : 0
phash_done          : True

## Last observation
{
  "ok": true,
  "n_input": 26,
  "n_output": 24,
```

```
  "n_removed": 2,
  "elapsed_ms": 768
}

## Recent scratchpad (most recent last)
[step 1] action=phash_dedup args={"threshold": 5, "workers": 16, "timeout": 15}
observation={"ok": true, "n_input": 26, "n_output": 24, "n_removed": 2, "elapsed_ms": 768}

Now emit the next JSON action.
```

Model raw output

```
{
  "thought": "Phash dedup is complete, reducing the pool from 26 to 24 units. The next
logical step is to retrieve relevant candidates using embedding-based recall. Since the
query is about identifying a motorcycle model from an image, using the 'qwen_vl' path with
the query text enabled is appropriate as it leverages multimodal understanding to match the
visual content with the specific text query.",
  "action": "filter_embedding",
  "args": {
    "path": "qwen_vl",
    "topk": 10,
    "min_score": 0.7,
    "use_text": true
  }
}
```

I.3 STEP 3: VERIFIER_RANK

Prompt

```
## Current context
request_id      : ...
query_text      : What model is the motorcycle in the image?
query_image_url : ...

## Progress so far
tool_calls_used     : 2 / 30
pool_units_total    : 24
filtered_units      : 10
ranked_units        : 0
selected_units      : 0
docs_generated      : 0
docs_passed         : 0
phash_done          : True

## Last observation
{
  "ok": true,
  "path": "qwen_vl",
  "kept": 10,
  "cumulative_filtered": 10,
  "top_units": [...]
}

## Recent scratchpad (most recent last)
[step 1] action=phash_dedup args={"threshold": 5, "workers": 16, "timeout": 15}
observation=...
[step 2] action=filter_embedding args={"path": "qwen_vl", "topk": 10, "min_score": 0.7,
"use_text": true} observation=...

Now emit the next JSON action.
```

Model raw output

```
{
  "thought": "I have successfully deduplicated the pool and filtered 10 candidate units
using the qwen_vl embedding path. The next step in the pipeline is to score these filtered
units for caption usefulness and relevance using the verifier_rank tool. This will allow me
to rank the candidates before selecting which ones to generate full docs for.",
  "action": "verifier_rank",
  "args": {}
}
```

I.4 STEP 4: SYNTHESIZER_SELECT

Prompt

```
## Current context
request_id      : ...
query_text      : What model is the motorcycle in the image?
query_image_url : ...

## Progress so far
tool_calls_used     : 3 / 30
pool_units_total    : 24
filtered_units      : 10
ranked_units        : 10
selected_units      : 0
docs_generated      : 0
docs_passed         : 0
phash_done          : True

## Last observation
{
  "ok": true,
  "n_scored": 10,
  "top": [
    {
      "unit_id": "...",
      "source": "...",
      "doc_score": 3.0,
      "is_badcase": false,
      "reason": "..."
    },
    ...
  ]
}

## Recent scratchpad (most recent last)
[step 1] action=phash_dedup args={"threshold": 5, "workers": 16, "timeout": 15}
observation=...
[step 2] action=filter_embedding args={"path": "qwen_vl", "topk": 10, "min_score": 0.7,
"use_text": true} observation=...
[step 3] action=verifier_rank args={} observation=...

Now emit the next JSON action.
```

Model raw output

```
{
  "thought": "The verifier_rank tool has identified 5 top units with a perfect score of
3.0, indicating they are highly likely to contain accurate information about the motorcycle
model. The next step in the pipeline is to select which of these ranked units are worth
generating a full ideal doc for. I will call synthesizer_select to pick the best candidates
from the current ranked list.",
  "action": "synthesizer_select",
  "args": {}
}
```

I.5 STEP 5: SYNTHESIZER_DOC

Prompt

```
## Current context
request_id      : ...
query_text      : What model is the motorcycle in the image?
query_image_url : ...

## Progress so far
tool_calls_used     : 4 / 30
pool_units_total    : 24
filtered_units      : 10
ranked_units        : 10
selected_units      : 1
docs_generated      : 0
docs_passed         : 0
phash_done          : True
```

```
## Last observation
{
  "ok": true,
  "n_selected": 1,
  "note": "...",
  "selected": [
    {
      "unit_id": "...",
      "source": "...",
      "image_url": "...",
      "embed_score": 0.791540016181292,
      "embed_score_dino": null,
      "embed_score_vl": 0.791540016181292,
      "target_image_caption": "..."
    }
  ]
}

## Recent scratchpad (most recent last)
[step 1] action=phash_dedup args={"threshold": 5, "workers": 16, "timeout": 15}
observation=...
[step 2] action=filter_embedding args={"path": "qwen_vl", "topk": 10, "min_score": 0.7,
"use_text": true} observation=...
[step 3] action=verifier_rank args={} observation=...
[step 4] action=synthesizer_select args={} observation=...

Now emit the next JSON action.
```

Model raw output

```
{
  "thought": "The synthesizer_select tool has identified one unit (...) as the most
promising candidate to generate an ideal document for. The next step in the pipeline is to
generate the summary/caption for this specific unit using the synthesizer_doc tool.",
  "action": "synthesizer_doc",
  "args": {
    "unit_id": "..."
  }
}
```

I.6 STEP 6: REFLECTOR_CHECK

Prompt

```
## Current context
request_id      : ...
query_text      : What model is the motorcycle in the image?
query_image_url : ...

## Progress so far
tool_calls_used     : 5 / 30
pool_units_total    : 24
filtered_units      : 10
ranked_units        : 10
selected_units      : 1
docs_generated      : 1
docs_passed         : 0
phash_done          : True

## Last observation
{
  "ok": true,
  "unit_id": "...",
  "doc": {
    "unit_id": "...",
    "ideal_image_url": "...",
    "caption": "...",
    "summary": "..."
  }
}

## Recent scratchpad (most recent last)
[step 1] action=phash_dedup args={"threshold": 5, "workers": 16, "timeout": 15}
observation=...
[step 2] action=filter_embedding args={"path": "qwen_vl", "topk": 10, "min_score": 0.7,
"use_text": true} observation=...
```

```
[step 3] action=verifier_rank args={} observation=...
[step 4] action=synthesizer_select args={} observation=...
[step 5] action=synthesizer_doc args={"unit_id": "..."} observation=...

Now emit the next JSON action.
```

Model raw output

```
{
  "thought": "I have generated a doc for unit_id '...'. The caption identifies the bike as
'Kymco CT250', but the query image (based on step 4 observation) shows 'X-TRIP' and 'XONNE'
(likely a Benelli X300 or similar). The synthesizer_select note indicated a potential
mismatch or that other candidates like 'CK250T-11' (step 3) were more accurate. I must
verify this generated doc. If it fails the reflector check (likely due to subject mismatch
Q1), I should proceed to verify the next best candidate from the ranked list (e.g., '...'
which identified 'Changchun CK250T-11' or '...' for 'Sicong RT2') to find a correct
match.",
  "action": "reflector_check",
  "args": {
    "unit_id": "..."
  }
}
```

I.7 STEP 7: FINISH

Prompt

```
## Current context
request_id      : ...
query_text      : What model is the motorcycle in the image?
query_image_url : ...

## Progress so far
tool_calls_used     : 6 / 30
pool_units_total    : 24
filtered_units      : 10
ranked_units        : 10
selected_units      : 1
docs_generated      : 1
docs_passed         : 1
phash_done          : True

## Last observation
{
  "ok": true,
  "unit_id": "...",
  "passed": true,
  "refetch": false,
  "q1_passed": true,
  "q2_passed": true,
  "q3_passed": true,
  "action": "pass",
  "q1_reason": "...",
  "q2_reason": "...",
  "q3_reason": "...",
  "n_passed_total": 1
}

## Recent scratchpad (most recent last)
[step 1] action=phash_dedup args={"threshold": 5, "workers": 16, "timeout": 15}
observation=...
[step 2] action=filter_embedding args={"path": "qwen_vl", "topk": 10, "min_score": 0.7,
"use_text": true} observation=...
[step 3] action=verifier_rank args={} observation=...
[step 4] action=synthesizer_select args={} observation=...
[step 5] action=synthesizer_doc args={"unit_id": "..."} observation=...
[step 6] action=reflector_check args={"unit_id": "..."} observation=...

Now emit the next JSON action.
```

Model raw output

```
{
```

```
  "thought": "I have successfully generated and verified one ideal document (unit_id: ...)
that accurately identifies the motorcycle model as 'Kymco CT250' (Kymco X-Trip/CT250) and
passes all reflection checks (Q1, Q2, Q3). The user's query 'What model is the motorcycle
in the image?' has been directly answered. Given the goal is to produce high-quality ideal
docs and I already have one passing doc, and considering the budget constraints, further
retrieval or generation is unlikely to add significant value unless the user specifically
requested multiple options. The current result is sufficient.",
  "action": "finish",
  "args": {
    "reason": "Successfully generated and verified one high-quality ideal doc identifying
the motorcycle as Kymco CT250 (X-Trip). The query is fully answered."
  }
}
```