ComfySearch: Autonomous Exploration and Reasoning for ComfyUI Workflows

Jinwei Su[1,*], **Qizhen Lan**[2,*], **Zeyu Wang**[3], **Yinghui Xia**[5], **Hairu Wen**[5]
Yiqun Duan[5], **Xi Xiao**[4], **Yang Jingsong**[5,†], **Lewei He**[1,†], **Tianyu Shi**[6,†]
[1]South China Normal University, [2]The University of Texas Health Science Center
[3]University of California, Los Angeles, [4]University of Alabama at Birmingham
[5]Autoagent.AI, [6]McGill University
edward.yang@agentspro.cn, 2024024579@m.scnu.edu.cn, tianyu.shi3@mcgill.ca

Abstract

AI-generated content has progressed from monolithic models to modular workflows, especially on platforms like ComfyUI, allowing users to customize complex creative pipelines. However, the large number of components in ComfyUI and the difficulty of maintaining long-horizon structural consistency under strict graph constraints frequently lead to low pass rates and workflows of limited quality. To tackle these limitations, we present ComfySearch, an agentic framework that can effectively explore the component space and generate functional ComfyUI pipelines via validation-guided workflow construction. Experiments demonstrate that ComfySearch substantially outperforms existing methods on complex and creative tasks, achieving higher executability (pass) rates, higher solution rates, and stronger generalization.Our code is open-sourced at https://github.com/SHITIANYU-hue/comfysearch.git

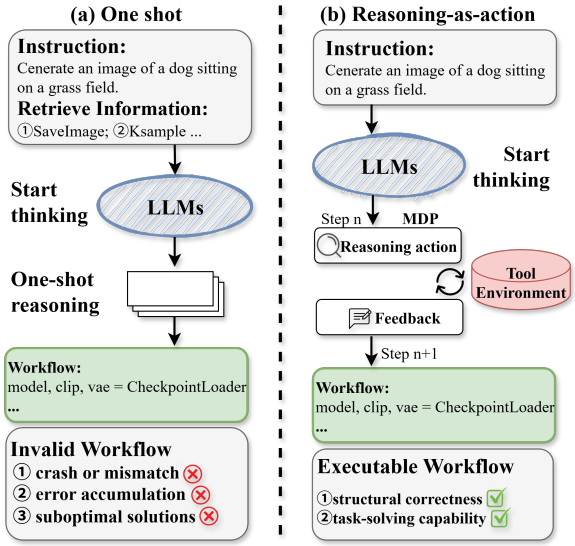

Figure 1: One-shot workflow planning generates the entire graph at once and is brittle, whereas reasoning-as-action generation (ComfySearch) incrementally builds and validates each edit, ensuring robust and executable workflows.

1 Introduction

Generative vision models increasingly power text-to-image synthesis (Podell et al., 2023; Esser et al., 2024; Labs, 2024), image editing (Brooks et al., 2023; Ai, 2024; Huang et al., 2024), and video generation (HaCohen et al., 2024; Tencent, 2024; Wang et al., 2025). While large general-purpose models (Ge et al., 2024; Wu et al., 2024; OpenAI, 2025) can deliver strong quality across tasks, they are expensive to customize and deploy. This has spurred compositional workflow systems that expose intermediate structure for controllable design (Gal et al., 2024). A representative platform is ComfyUI (ComfyUI Contributors, 2023), which represents a generation pipeline as a typed node graph that users compose into reusable, task-specific workflows.

Despite its flexibility, ComfyUI authoring is brittle. A workflow must satisfy strict type/interface constraints and global graph invariants; a single mismatched edge or missing adapter can render the graph non-executable. As a result, users often rely on expertise and repeated trial-and-error. Recent LLM-based systems aim to automate workflow construction through explicit planning and composition (e.g., ComfyAgent (Xue et al., 2025), ComfyMind (Guo et al., 2025)) or by training generators with reasoning traces (e.g., ComfyUI-R1 (Xu et al., 2025)). However, long-horizon construction remains fragile: decisions that are locally plausible may later violate downstream compatibility, producing workflows that appear coherent but crash at execution time or deviate from the user's intent.

The core difficulty is that workflow generation is not purely retrieval. Retrieval can identify relevant nodes and documentation, but it does not ensure

*Equal contribution. †Corresponding authors.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 734–748
October 24-29, 2026 ©2026 Association for Computational Linguistics

state-dependent composability: whether an edit is valid depends on the current partial graph and how future branches will constrain interfaces. This motivates an interactive, execution-based paradigm in which each proposed graph edit is immediately validated for *executability* (typing/schema / invariants) and repaired using diagnostic feedback, rather than relying on one-shot plans(Figure 1).

We distill two challenges. (***C1***) *Online structural correctness.* Deferring validation allows early wiring mistakes to propagate in typed graphs, leading to compounding errors in long-horizon generation (Arora et al., 2022). A practical generator should therefore validate each edit against the current state and repair violations on the fly. (***C2***) *Long-horizon branching under uncertainty.* Multiple edits may be locally admissible yet induce sharply different downstream feasibility; moreover, validator feedback changes the effective state distribution during construction. Thus, the generator should *selectively branch* at high-uncertainty decision points instead of committing to a single trajectory (Yao et al., 2023; Haarnoja et al., 2018).

These observations suggest a *reasoning-as-action* view: the model incrementally edits the workflow graph, queries a validator for state-aware feedback, and revises decisions when incompatibilities arise. We view this process as an episodic Markov Decision Process (MDP), where the state is the current workflow prefix, an action is a node-level graph edit, and transitions return the updated graph with validation feedback.

To address (***C1***) and (***C2***), we propose **ComfySearch**, an reasoning-as-action framework for ComfyUI workflow generation. ComfySearch enforces executability via node-level, state-aware validation and in-place repair (***C1***). It further introduces *entropy-adaptive branching*, using policy uncertainty to allocate a limited rollout budget toward ambiguous decision points, improving robustness and efficiency without exhaustive branching (***C2***) (Wang et al., 2026; Dong et al., 2025b). We also construct consistency-filtered query–workflow supervision by executing candidate workflows and canonicalizing multiple valid designs within each query cluster, reducing noisy node-level labels. Finally, we train ComfySearch with tool-mediated feedback under a reinforcement learning objective, enabling the model to learn which nodes to add, when to validate intermediate edits, and when to branch. Our contributions are:

- Reasoning-as-action workflow generation: we formulate ComfyUI construction as reasoning-as-action with state-aware online validation and in-place repair.
- Entropy-adaptive branching: we selectively explore high-ambiguity decisions using policy uncertainty, improving robustness and efficiency under limited rollout budgets.
- Execution-verified supervision and RL training: we curate consistency-filtered data and train with tool-mediated feedback; evaluations on ComfyBench and GenEval improve executability, task success, and efficiency.

2 Related work

Workflow Generation. LLM-based workflow generation translates natural-language requests into structured plans or executable programs (Hu et al., 2024; Zhang et al., 2024, 2025; Ye et al., 2025; Su et al., 2026). Recent work extends this paradigm to ComfyUI, where a workflow is a typed node graph with strict interface constraints. Prior systems explore multi-agent decomposition (e.g., ComfyAgent (Xue et al., 2025) and ComfyGPT (Huang et al., 2025)), modular and hierarchical planning (e.g., ComfyMind (Guo et al., 2025)), or long-form reasoning with documentation retrieval (e.g., ComfyUI-R1 (Xu et al., 2025)). While effective in proposing plausible graphs, these approaches are typically not *execution-grounded*. They lack node-level, *state-aware validation* of each graph edit and do not *selectively branch* at high-uncertainty decision points, so locally admissible edits can later violate downstream compatibility and fail at execution time.

Agentic Reinforcement Learning. Agentic RL has recently advanced long-horizon tool-use and structured decision-making by allocating exploration to uncertain steps (Lan et al., 2026; Li et al., 2026; Xiao et al., 2026). GRPO (Shao et al., 2024) provides a PPO-style optimization scheme that improves training efficiency, while ARPO (Dong et al., 2025b) and AEPO (Dong et al., 2025a) use entropy-aware rollouts/updates to encourage exploration and stabilize learning in multi-turn settings. Complementary lines of work study uncertainty-triggered information seeking (e.g., Search-o1 (Li et al., 2025)) and show that high-entropy minority tokens function as critical "forks" in RLVR for LLM reasoning (Wang

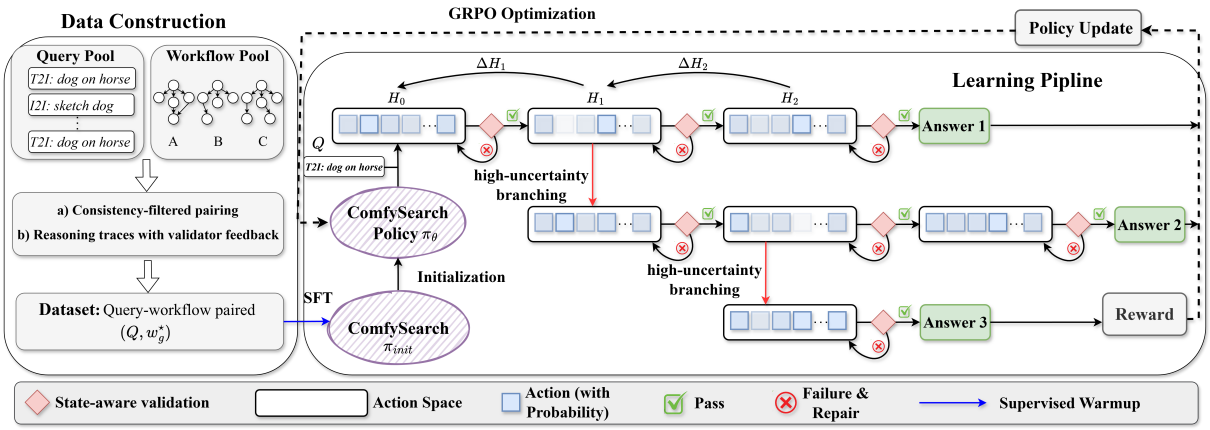

Figure 2: Overview of the ComfySearch framework. We formulate workflow generation as a Markov Decision Process(§3.2). (a) State-aware Validation(§3.3): Implements real-time verification and repair to ensure online structural correctness (***C1***). (b) Entropy Rollout Branching(§3.4): Employs selective exploration at high-uncertainty points to navigate long-horizon decision branching (***C2***).

et al., 2026). Our ComfySearch instantiates these insights at the *graph-edit action* level by coupling entropy-adaptive branching with state-aware validation feedback, enabling robust workflow construction under execution grounding.

3 Method

ComfySearch constructs an executable ComfyUI workflow by iteratively proposing atomic *graph edits* and grounding each edit with *node-level, state-aware validation feedback*. This design addresses two challenges from §1: (i) *online structural correctness* via validation and in-place repair (**C1**), and (ii) *long-horizon branching under uncertainty* via entropy-adaptive selective branching (**C2**). We learn a policy that decides *what* to edit, *how* to repair after violations, and *when* to branch from validated prefixes. Figure 2 illustrates the overall pipeline of our method. For clarity, we summarize the full notations in Appendix C.

3.1 Dataset Construction

Community-shared ComfyUI workflows rarely come with aligned natural-language queries, and synthetic pairing can introduce query–workflow mismatches that are especially harmful for node-level learning. We therefore build training data using (i) **consistency-filtered pairing** and (ii) **validator-grounded reasoning traces**. Detailed data statistics are provided in Appendix A.

Consistency-filtered pairing. We collect a workflow pool and a query pool, and only match within the same modality family (e.g., T2I/I2I/T2V/V2V/I2V). For each query Q, we execute candidate workflows and obtain a binary correctness score $s(w, Q) \in \{0, 1\}$ using an LLM judge (GPT-4o; details in Appendix E). We keep only workflows with $s(w, Q) = 1$. To reduce ambiguity from multiple valid designs, we cluster semantically similar queries into groups $\{\mathcal{Q}_g\}$ and select a canonical workflow per group:

$$w_g^\star = \arg\max_{w \in \mathcal{W}_g} \sum_{Q \in \mathcal{Q}_g} s(w, Q), \qquad (1)$$

where $\mathcal{W}_g$ denotes workflows that succeed for at least one query in $\mathcal{Q}_g$. This yields stable supervision by mapping each query group to a single consistently successful workflow.

Reasoning traces with validator feedback. For each paired $(Q, w_g^\star)$, we prompt a stronger teacher model to produce stepwise traces(details in Appendix G). Each step proposes one atomic edit in `<node>` tags, followed by a `<result>` block that records validator feedback (accept/reject plus diagnostics). These traces supervise both *what to add* and *how to react* to tool feedback, which is critical for long-horizon construction.

3.2 Reasoning-As-Action Graph-Editing MDP

Workflow representation. A ComfyUI workflow is a typed directed graph $G = (\mathcal{V}, \mathcal{E})$, where each node is an operator with typed input/output ports, and edges connect type-compatible ports. Let $G^\star$ denote the canonical target workflow paired with Q. We denote the current partial workflow at

interaction step t by G_t with $G_0 = \varnothing$; edits are committed only when accepted by the validator.

State and actions. We model construction as an episodic MDP with state

$$s_t \triangleq (Q, G_t, h_t), \tag{2}$$

where h_t summarizes recent diagnostics and accumulated constraints (making the process Markov in our tool-mediated environment). The policy emits a textual action a_t, deterministically parsed into an atomic graph edit Δ_t via Φ:

$$\Delta_t = \Phi(a_t). \tag{3}$$

Edits include adding a node (with parameters), adding/removing an edge between typed ports, and updating parameters. A special `STOP` action triggers termination if the current workflow passes a final executability check.

Tool-mediated transition. The environment applies an edit only if it passes *state-aware* validation:

$$(G_{t+1}, y_t, m_t) = \textsc{Validate}(G_t, \Delta_t), \tag{4}$$

where $y_t \in \{0, 1\}$ indicates acceptance and m_t is a diagnostic message (empty if $y_t = 1$). If $y_t = 0$, we keep $G_{t+1} = G_t$ and the policy may propose repairs conditioned on m_t (§3.3). We update history by $h_{t+1} \leftarrow \textsc{Update}(h_t, a_t, y_t, m_t)$.

3.3 State-Aware Validation and In-Place Repair (C1)

State-aware validator. Local plausibility is insufficient in typed workflow graphs. We define $\textsc{Validate}(G_t, \Delta_t) \to (G_{t+1}, y_t, m_t)$ by composing two checks:

$$y_t \;=\; \textsc{Int}(\Delta_t) \;\wedge\; \textsc{Comp}(G_t, \Delta_t), \tag{5}$$

where $\textsc{Int}(\Delta_t)$ verifies intrinsic validity (operator existence, required fields, parameter domains, schema validity), and $\textsc{Comp}(G_t, \Delta_t)$ verifies composability w.r.t. the current graph (typed port compatibility, required adapters/casts, and graph-level invariants such as acyclicity and conditioning-branch constraints). Importantly, this execution grounding is implemented as *state-aware executability checking* rather than running full generative inference after every edit.

In-place repair with diagnostics. If $y_t = 0$, the diagnostic message m_t specifies the violated constraint (e.g., type mismatch, missing adapter, illegal topology). The policy proposes a repaired edit conditioned on m_t:

$$a_t^{\text{rep}} \sim \pi_\theta(\cdot \mid Q, G_t, h_t, m_t), \qquad \Delta_t^{\text{rep}} = \Phi(a_t^{\text{rep}}), \tag{6}$$

which is re-validated via Eq. (4). This validate–repair loop enforces online structural correctness and prevents early wiring errors from compounding over long horizons (**C1**).

Validated-prefix property. Because only accepted edits update the graph, every prefix along a rollout is executable by construction. Entropy-adaptive branching (§3.4) expands additional children only from such validated prefixes.

3.4 Entropy-Adaptive Branching Under Uncertainty (C2)

Even under state-aware validation, multiple edits can be admissible yet lead to sharply different downstream feasibility and task utility. Motivated by entropy-guided exploration in long-horizon reasoning and tool use (Wang et al., 2026; Dong et al., 2025b), we *selectively branch* at high-uncertainty decision points to explore diverse, feedback-consistent continuations under finite budgets.

Step-level uncertainty. Let $\tilde{\mathcal{A}}_t$ be a constrained candidate set (e.g., top-K actions under constrained decoding), and $p_t(a) = \pi_\theta(a \mid s_t)$ for $a \in \tilde{\mathcal{A}}_t$. We use normalized entropy:

$$H_t \triangleq -\frac{1}{\log |\tilde{\mathcal{A}}_t|} \sum_{a \in \tilde{\mathcal{A}}_t} p_t(a) \log p_t(a) \in [0, 1]. \tag{7}$$

We trigger branching using an entropy increase criterion $\Delta H_t \triangleq H_t - H_{t-1}$ (with $H_{-1} = 0$).

Branching rule. We map uncertainty to a branching probability:

$$P_t^{\text{branch}} \triangleq \sigma(\alpha + \beta \Delta H_t), \tag{8}$$

where $\sigma(\cdot)$ is the sigmoid function and α, β are scalars. If $P_t^{\text{branch}} > \tau_b$, we spawn an additional branch from the same validated prefix by sampling an alternative action (e.g., top-K resampling), while continuing the current branch unchanged. All branches remain subject to Eq. (4), so branching never compromises online executability (**C2**).

Method	Model	Vanilla		Complex		Creative		Total	
		%Pass	%Resolve	%Pass	%Resolve	%Pass	%Resolve	%Pass	%Resolve
Zero-shot	GPT-4o	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Few-shot	GPT-4o	32.0	27.0	16.7	8.3	7.5	0.0	22.5	16.0
CoT	GPT-4o	44.0	29.0	11.7	8.3	12.5	0.0	28.0	17.0
RAG	Claude-3.5-Sonnet	27.0	13.0	23.0	6.7	7.5	0.0	22.0	8.5
	LLaMA-3.1-70B	58.0	32.0	23.0	10.0	15.0	5.0	39.0	20.0
	Qwen2.5-70B	60.0	33.0	25.0	13.0	15.0	5.0	40.5	21.5
	GPT-4o	62.0	41.0	45.0	21.7	40.0	7.5	52.0	23.0
	o1-preview	70.0	46.0	48.3	23.3	30.0	12.5	55.5	32.5
ComfyAgent	Qwen2.5-72B	63.0	38.0	25.0	18.3	20.0	5.0	43.0	25.5
SFT+GRPO	ComfyUI-R1	-	-	-	-	-	-	67.0	-
ComfyMind	Qwen2.5-72B	-	80.0	-	63.0	-	25.0	-	64.0
Ours	**ComfySearch-7B**	**92.0**	**80.0**	**91.6**	**76.6**	**95.0**	**42.5**	**92.5**	**71.5**

Table 1: ComfyBench (Xue et al., 2025) evaluation of autonomous ComfyUI workflow construction. %Pass measures whether the generated workflow executes successfully in ComfyUI, and %Resolve measures whether the executed output satisfies all task requirements. Results are reported by difficulty subset and overall; "-" denotes metrics not reported by the corresponding method or not comparable. Higher is better.

3.5 Learning with Supervised Warmup and GRPO

We train the policy π_θ in two stages: supervised warmup followed by tool-feedback reinforcement learning.

Supervised warmup. Given training tuples $\mathcal{D} = \{(Q, T, G^\star)\}$, where T is a teacher trace and $G^\star$ is the target workflow, we maximize:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(Q,T,G^\star)\sim\mathcal{D}}\Big[\log p_\theta(T, G^\star \mid Q)\Big]. \tag{9}$$

Tool-feedback RL with GRPO. We adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024). For each query, we generate a group of rollouts under the tool-mediated environment(including entropy-adaptive branching under a fixed budget), compute terminal rewards, and update the policy with GRPO (full objective in Appendix B).

3.6 Hierarchical Terminal Reward

We use a hierarchical terminal reward that evaluates (i) output format, (ii) trace–workflow consistency, and (iii) structural accuracy against $G^\star$. We provide a detailed discussion of this design choice in Appendix I.

R_f: format validity. We require a well-formed `<trace>` block (containing `<node>` and `<result>` steps) and a parseable `<workflow>` block. If the format is invalid, we set $R_f = -1$.

R_c: trace–workflow consistency. We further require that each node type appearing in the final workflow is mentioned in the trace; otherwise $R_c = -1$.

Structural term. Let $\textsc{Types}(G)$ be the set of node types in graph G. We use node-type recall:

$$\widehat{R}_s(G_T, G^\star) \triangleq \frac{|\textsc{Types}(G_T) \cap \textsc{Types}(G^\star)|}{|\textsc{Types}(G^\star)|} - 1. \tag{10}$$

Final reward. We apply veto-style gates and a graded structural score:

$$R(\tau) = \begin{cases} -1, & R_f = -1 \text{ or } R_c = -1, \\ \frac{3+\widehat{R}_s(G_T,G^\star)}{3}, & \text{otherwise.} \end{cases} \tag{11}$$

This provides informative signals for valid trajectories while sharply penalizing invalid outputs.

4 Experiments

4.1 Experiment Settings

Dataset and Benchmarks. We build our training data from Flow-Dataset (Huang et al., 2025), which contains community-shared ComfyUI workflows. Since such workflows are rarely accompanied by aligned natural-language queries, we follow the data construction pipeline in Section 3.1 to obtain query–workflow supervision, including consistency-filtered pairing and validator-grounded reasoning traces. We evaluate ComfySearch on two benchmarks: ComfyBench (Xue et al., 2025)

Method	Overall	Single Obj.	Two Obj.	Counting	Colors	Position	Attri. Binding
Frozen Text Encoder Mapping Methods							
SDv1.5 (Rombach et al., 2022)	0.43	0.97	0.38	0.35	0.76	0.04	0.06
SDv2.1 (Rombach et al., 2022)	0.50	0.98	0.51	0.44	0.85	0.07	0.17
SD-XL (Podell et al., 2023)	0.55	0.98	0.74	0.39	0.85	0.15	0.23
DALLE-2 (Ramesh et al., 2022)	0.52	0.94	0.66	0.49	0.77	0.10	0.19
SD3-Medium (Esser et al., 2024)	0.74	0.99	0.94	0.72	0.89	0.33	0.60
LLMs/MLLMs Enhanced Methods							
LlamaGen (Sun et al., 2024)	0.32	0.71	0.34	0.21	0.58	0.07	0.04
Chameleon (Team, 2024)	0.39	-	-	-	-	-	-
LWM (Liu et al., 2025)	0.47	0.93	0.41	0.46	0.79	0.09	0.15
SEED-X (Ge et al., 2024)	0.49	0.97	0.58	0.26	0.80	0.19	0.14
Emu3-Gen (Wang et al., 2024)	0.54	0.98	0.71	0.34	0.81	0.17	0.21
Janus (Wu et al., 2024)	0.61	0.97	0.68	0.30	0.84	0.46	0.42
JanusFlow (Ma et al., 2024)	0.63	0.97	0.59	0.45	0.83	0.53	0.42
Janus-Pro-7B (Chen et al., 2025)	0.80	0.99	0.89	0.59	**0.90**	**0.79**	0.66
GoT (Fang et al., 2025)	0.64	0.99	0.69	0.67	0.85	0.34	0.27
Workflow Generation Systems							
ComfyAgent (Xue et al., 2025)	0.32	0.69	0.30	0.33	0.50	0.04	0.04
ComfyMind (Guo et al., 2025)	0.80	0.95	0.88	**0.83**	0.89	0.58	0.60
ComfySearch	**0.82**	**1.00**	**0.91**	0.79	0.86	0.65	**0.75**

Table 2: GenEval (Ghosh et al., 2023) text-to-image compositional accuracy. For workflow-generation systems, we execute the generated ComfyUI workflows to synthesize images and then evaluate them with GenEval. Obj.: object; Attr.: attribute. Higher is better.

measures autonomous workflow construction under strict executability constraints, and GenEval (Ghosh et al., 2023) evaluates downstream text-to-image compositional accuracy.

Implementation Details. We use Qwen2.5-Coder-7B-Instruct as the backbone of ComfySearch. The SFT stage is trained on 8 NVIDIA A100 (80GB) GPUs for 1 epoch with learning rate 2×10^{-5} and per-GPU batch size 1. During RL training, we use 4 NVIDIA A100 (80GB) GPUs for 600 steps with learning rate 5×10^{-7} and total batch size 16. In Eq. 8, we set $\beta = 0.2$, $\alpha = 0.5$, and $\tau_b = 0.5$. Top-K = 10(Section 3.4). We set 5 rollouts in GRPO, with each rollout splitting into at most two sub-branches. Additional sensitivity analysis is provided in the Appendix H.

4.2 Autonomous Workflow Construction

We evaluate autonomous workflow construction on ComfyBench, which contains 200 generation and editing tasks spanning image and video modalities with three difficulty subsets (Vanilla, Complex, and Creative). For each task, the model must synthesize a ComfyUI workflow that (i) executes successfully and (ii) satisfies the user instruction. ComfyBench reports two metrics: executability (%Pass), indicating whether the workflow runs without execution errors, and task success (%Resolve), indicating whether the executed output meets all task requirements.

As shown in Table 1, ComfySearch achieves strong overall performance with 92.5 %Pass and 71.5 %Resolve, substantially outperforming zero-/few-shot prompting and retrieval-augmented baselines. Compared to the learning-based workflow generator ComfyUI-R1, ComfySearch improves the overall %Pass from 67.0 to 92.5, indicating markedly more reliable executability. Relative to ComfyAgent, ComfySearch improves both %Pass and %Resolve across all difficulty levels. Compared to the state-of-the-art ComfyMind, ComfySearch matches its %Resolve on Vanilla (80.0) while achieving large gains on Complex (76.6 vs. 63.0) and Creative (42.5 vs. 25.0), resulting in a higher overall %Resolve (71.5 vs. 64.0). We attribute these gains to reasoning-as-action workflow construction: state-aware validation with in-place repair prevents early structural errors from compounding, and entropy-adaptive branching enables targeted exploration at high-uncertainty decision points.

4.3 Text-to-Image Generation

We report downstream text-to-image performance on GenEval. GenEval evaluates compositional alignment across six dimensions, including single/-multiple objects, counting, color accuracy, spatial positioning, and attribute binding. For workflow-generation systems, we execute the generated ComfyUI workflows to synthesize images and then com-

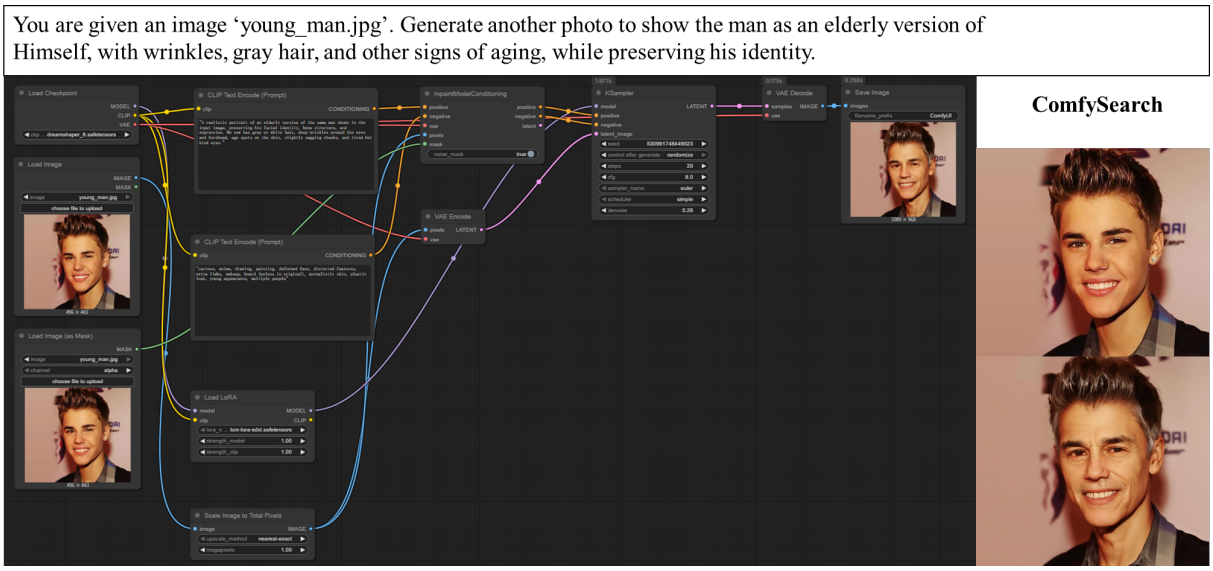

Figure 3: Examples generated by ComfySearch.

pute GenEval scores.

As shown in Table 2, ComfySearch achieves an overall GenEval score of 0.82, outperforming ComfyAgent (0.32) by a large margin and improving over SD3-Medium (0.74) among frozen text-encoder mapping methods. ComfySearch also slightly improves over ComfyMind (0.80) and is competitive with strong multimodal generation systems such as Janus-Pro-7B (0.80), achieving particularly strong performance on composition-sensitive dimensions (e.g., attribute binding).

4.4 Qualitative Results

Figure 3 shows a representative example workflow generated by ComfySearch from a user query, illustrating that ComfySearch can interpret the instruction, construct a valid executable workflow, and complete the task successfully. Additional qualitative comparisons, including execution outputs and side-by-side results against ComfyMind, are provided in Figure 4 and Appendix D.

Methods	Prompt Tokens	Completion Tokens	Wall-clock Time (s)	Resolve (%)
ComfySearch	9,332	24,895	145	100
ComfyMind	145,532	11,400	185	100

Table 3: Inference efficiency comparison between ComfySearch and ComfyMind on 20 ComfyBench instances filtered to those solved by both methods. We report total prompt tokens, completion tokens, and end-to-end wall-clock time aggregated over the 20 instances; Resolve denotes the task success rate on this filtered subset.

4.5 Efficiency Analysis

We randomly sample 20 ComfyBench instances and compare inference efficiency between ComfySearch and ComfyMind. To focus on efficiency rather than failure cases, we restrict the analysis to instances that are successfully solved by both methods; consequently, both methods achieve 100% %Resolve on this filtered subset (Table 3). Under this controlled setting, ComfySearch uses substantially fewer total tokens and achieves lower end-to-end wall-clock time than ComfyMind. This advantage is consistent with ComfyMind's iterative tree-based composition and replanning, which can incur large prompt overhead, whereas our ComfySearch performs reasoning-as-action editing with immediate validation/repair signals and selective exploration. Full-dataset efficiency results are reported in Appendix F.

Methods	Vanilla		Complex		Creative	
	%Pass	%Resolve	%Pass	%Resolve	%Pass	%Resolve
ComfySearch	**92.0**	**80.0**	**91.6**	**76.6**	**95.0**	**42.5**
SFT	79.0	60.0	60.0	30.0	72.5	25.0
SFT+GRPO	80.0	65.0	66.6	38.0	87.5	30.0
SFT+GRPO+VAL	88.0	74.0	88.3	73.0	92.5	37.5
RAG+SFT+GRPO	86.0	68.0	70.0	55.0	87.5	35.0

Table 4: Ablation results of our ComfySearch on ComfyBench. VAL denotes the state-aware validation and in-place repair module (C1). ComfySearch(SFT+GRPO+VAL+Entropy-Adaptive Branching). Higher is better.

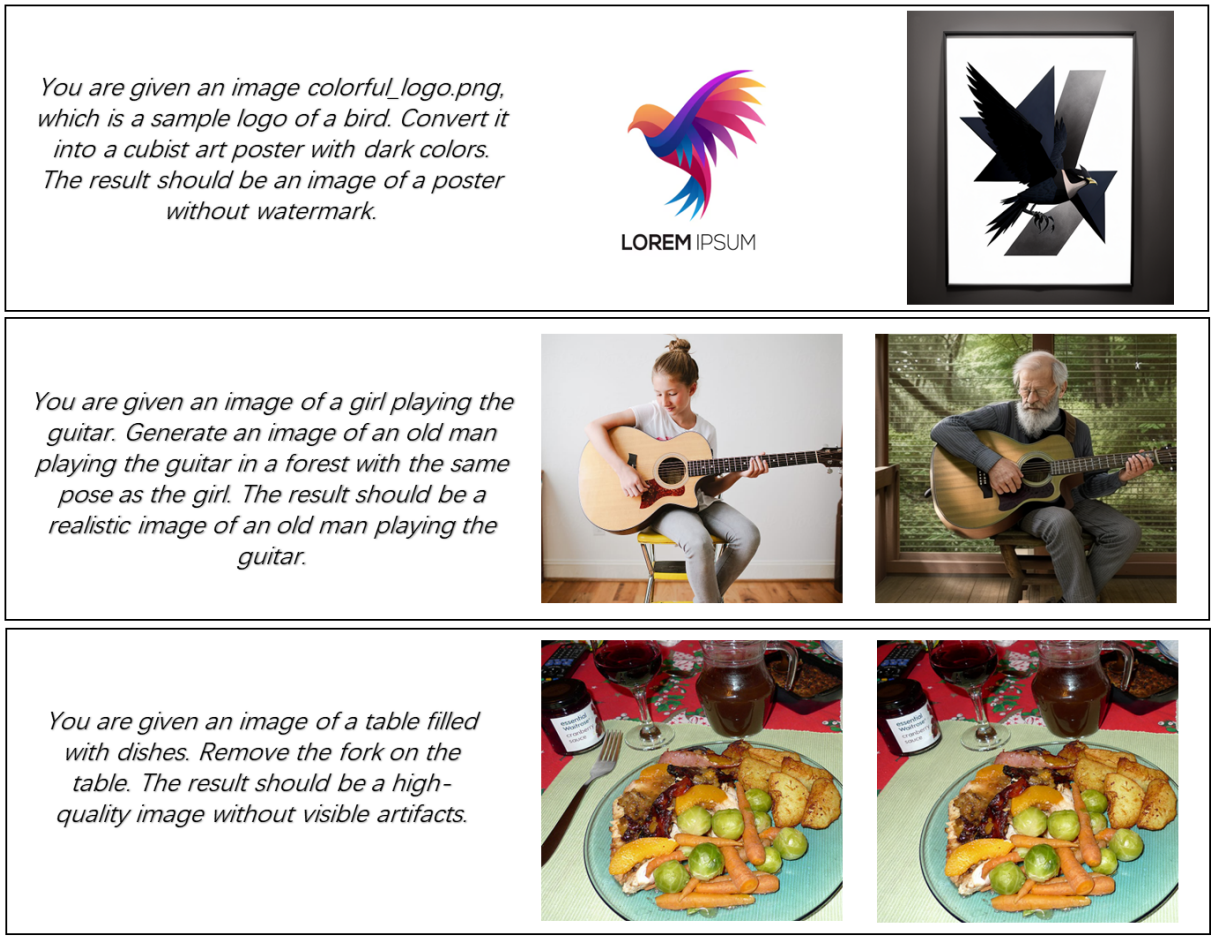

Figure 4: Qualitative results showing the performance of our method on various tasks.

4.6 Ablation Study

Method Ablation. Table 4 ablates key training and inference components in ComfySearch. Starting from SFT, adding RL (SFT+GRPO) consistently improves both %Pass and %Resolve. Incorporating *state-aware validation* (denoted as *VAL*) on top of SFT+GRPO yields further gains, especially on Complex and Creative tasks, demonstrating the importance of online structural correctness (C1). The full ComfySearch system achieves the best overall performance, suggesting that uncertainty-guided entropy-adaptive branching complements validation-guided editing in long-horizon workflow construction (C2). We also compare to a retrieval-augmented variant (RAG+SFT+GRPO), which underperforms ComfySearch on the harder subsets, indicating that execution-grounded exploration is more effective than relying on static retrieval alone.

Data Construction Ablation. To evaluate the data construction components in §3.1, we remove either consistency-filtered pairing (*w/o Consistency*) or validator-grounded reasoning traces (*w/o Reasoning*). As shown in Table 5, removing either component consistently degrades performance across all difficulty levels. In particular, removing reasoning traces leads to notable drops on Complex and Creative tasks, highlighting the importance of intermediate supervision for robust workflow construction. Furthermore, we ablate teacher model capacity in Appendix G.

Datasets	Vanilla		Complex		Creative	
	%Pass	%Resolve	%Pass	%Resolve	%Pass	%Resolve
Ours	**92.0**	**80.0**	**91.6**	**76.6**	**95.0**	**42.5**
w/o Consistency	85.0	66.0	80.0	63.0	92.5	30.0
w/o Reasoning	85.0	68.0	78.3	60.0	87.5	32.5

Table 5: Ablation of training-data construction (§3.1) on ComfyBench. "w/o Consistency" removes consistency-filtered pairing, and "w/o Reasoning" removes validator-grounded reasoning traces. Higher is better.

5 Conclusion

We present ComfySearch, an reasoning-as-action framework for autonomous ComfyUI workflow generation. We formulate the process as a Markov Decision Process (MDP) with two core innovations: (1) State-aware validation and in-place repair to ensure online structural correctness, preventing error propagation; and (2) Entropy-adaptive branching to navigate long-horizon uncertainty. Optimized via GRPO with hierarchical rewards, ComfySearch enables robust exploration beyond static planning. Experiments on ComfyBench and GenEval demonstrate that our 7B model significantly outperforms baselines in generalization for workflows.

Limitations

ComfySearch relies on explicit validators and curated execution-verified supervision, which may limit performance when encountering entirely unseen components or rapidly evolving node libraries. Although entropy-adaptive branching mitigates exhaustive search, the exploration cost can still grow for highly ambiguous or very large workflow spaces. Moreover, our evaluation focuses on ComfyUI-centric benchmarks, and validating the generality of this execution-grounded framework across other workflow systems and domains remains an important direction for future work.

References

Stability Ai. 2024. Cosxl.

Kushal Arora, Layla El Asri, Hareesh Bahuleyan, and Jackie Chi Kit Cheung. 2022. Why exposure bias matters: An imitation learning perspective of error accumulation in language generation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 700–710.

Tim Brooks, Aleksander Holynski, and Alexei A Efros. 2023. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18392–18402.

Xiaokang Chen, Zhiyu Wu, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, and Chong Ruan. 2025. Janus-pro: Unified multimodal understanding and generation with data and model scaling. *arXiv preprint arXiv:2501.17811*.

ComfyUI Contributors. 2023. ComfyUI: A powerful and modular stable-diffusion gui. https://github.com/comfyanonymous/ComfyUI. Accessed: 2025-05-14.

Guanting Dong, Licheng Bao, Zhongyuan Wang, Kangzhi Zhao, Xiaoxi Li, Jiajie Jin, Jinghan Yang, Hangyu Mao, Fuzheng Zhang, Kun Gai, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. 2025a. Agentic entropy-balanced policy optimization. *ArXiv*, abs/2510.14545.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. 2025b. Agentic reinforced policy optimization. *ArXiv*, abs/2507.19849.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, and 1 others. 2024. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*.

Rongyao Fang, Chengqi Duan, Kun Wang, Linjiang Huang, Hao Li, Shilin Yan, Hao Tian, Xingyu Zeng, Rui Zhao, Jifeng Dai, and 1 others. 2025. Got: Unleashing reasoning capability of multimodal large language model for visual generation and editing. *arXiv preprint arXiv:2503.10639*.

Rinon Gal, Adi Haviv, Yuval Alaluf, Amit H Bermano, Daniel Cohen-Or, and Gal Chechik. 2024. Comfygen: Prompt-adaptive workflows for text-to-image generation. *arXiv preprint arXiv:2410.01731*.

Yuying Ge, Sijie Zhao, Jinguo Zhu, Yixiao Ge, Kun Yi, Lin Song, Chen Li, Xiaohan Ding, and Ying Shan. 2024. Seed-x: Multimodal models with unified multi-granularity comprehension and generation. *arXiv preprint arXiv:2404.14396*.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. 2023. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems*, 36:52132–52152.

Litao Guo, Xinli Xu, Luozhou Wang, Jiantao Lin, Jinsong Zhou, Zixin Zhang, Bolan Su, and Ying-Cong Chen. 2025. Comfymind: Toward general-purpose generation via tree-based planning and reactive feedback. *arXiv preprint arXiv:2505.17908*.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr.

Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, and 1 others. 2024. Ltx-video: Realtime video latent diffusion. *arXiv preprint arXiv:2501.00103*.

Shengran Hu, Cong Lu, and Jeff Clune. 2024. Automated design of agentic systems. *ArXiv*, abs/2408.08435.

Oucheng Huang, Yuhang Ma, Zeng Zhao, Mingrui Wu, Jiayi Ji, Rongsheng Zhang, Zhipeng Hu, Xiaoshuai Sun, and Rongrong Ji. 2025. Comfygpt: A self-optimizing multi-agent system for comprehensive comfyui workflow generation. *Preprint*, arXiv:2503.17671.

Yuzhou Huang, Liangbin Xie, Xintao Wang, Ziyang Yuan, Xiaodong Cun, Yixiao Ge, Jiantao Zhou, Chao Dong, Rui Huang, Ruimao Zhang, and 1 others. 2024. Smartedit: Exploring complex instruction-based image editing with multimodal large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8362–8371.

Black Forest Labs. 2024. Flux.

Qizhen Lan, Xi Xiao, Xiangchen Guan, Mengchen Fan, Moule Lin, Jung Im Choi, and Lijing Zhu. 2026. Trust is not enough: Influence calibration for on-policy self-distillation in agentic rl. *Preprint*, arXiv:2608.14945.

Junjie Li, Xi Xiao, Yunbei Zhang, Chen Liu, Lin Zhao, Xiaoying Liao, Yingrui Ji, Janet Wang, Yingqiang Ge, Weijie Xu, Xi Fang, Xiang Xu, Tianchen Zhao, Youngeun Kim, Jihun Hamm, Tianyang Wang, and Chandan Reddy. 2026. Agent harness engineering: A survey.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025. Search-o1: Agentic search-enhanced large reasoning models. *ArXiv*, abs/2501.05366.

Hao Liu, Wilson Yan, Matei Zaharia, and Pieter Abbeel. 2025. World model on million-length video and language with blockwise ringattention. In *International Conference on Learning Representations*, volume 2025, pages 45953–45977.

Yiyang Ma, Xingchao Liu, Xiaokang Chen, Wen Liu, Chengyue Wu, Zhiyu Wu, Zizheng Pan, Zhenda Xie, Haowei Zhang, Liang Zhao, and 1 others. 2024. Janusflow: Harmonizing autoregression and rectified flow for unified multimodal understanding and generation. *arXiv preprint arXiv:2411.07975*.

OpenAI. 2025. gpt-image-1.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. 2023. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junmei Song, Mingchuan Zhang, Y. K. Li, Yu Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv*, abs/2402.03300.

Jinwei Su, Qizhen Lan, Yinghui Xia, Lifan Sun, Weiyou Tian, Tianyu Shi, and Lewei He. 2026. Difficulty-aware agentic orchestration for query-specific multi-agent workflows. In *Proceedings of the ACM Web Conference 2026*, WWW '26, page 2060–2070, New York, NY, USA. Association for Computing Machinery.

Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. 2024. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*.

Chameleon Team. 2024. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*.

Tencent. 2024. Hunyuanvideo.

Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, and 1 others. 2025. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, and 1 others. 2026. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *Advances in Neural Information Processing Systems*, 38:115452–115486.

Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, and 1 others. 2024. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*.

Chengyue Wu, Xiaokang Chen, Zhiyu Wu, Yiyang Ma, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, Chong Ruan, and 1 others. 2024. Janus: Decoupling visual encoding for unified multimodal understanding and generation. *arXiv preprint arXiv:2410.13848*.

Xi Xiao, Chen Liu, Chih-Ting Liao, Yunbei Zhang, Qizhen Lan, Yuxiang Wei, Lin Zhao, Janet Wang, Jianyang Gu, Muchao Ye, Tianyang Wang, and Hao Xu. 2026. Staying vigilant: Mitigating visual laziness via counterfactual visual alignment in mllms. In *European Conference on Computer Vision*. Springer.

Zhenran Xu, Yiyu Wang, Xue Yang, Longyue Wang, Weihua Luo, Kaifu Zhang, Baotian Hu, and Min Zhang. 2025. Comfyui-r1: Exploring reasoning models for workflow generation. *ArXiv*, abs/2506.09790.

Xiangyuan Xue, Zeyu Lu, Di Huang, Zidong Wang, Wanli Ouyang, and Lei Bai. 2025. Comfybench: Benchmarking llm-based agents in comfyui for autonomously designing collaborative ai systems. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24614–24624. IEEE.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Rui Ye, Shuo Tang, Rui Ge, Yaxin Du, Zhenfei Yin, Siheng Chen, and Jing Shao. 2025. Mas-gpt: Training llms to build llm-based multi-agent systems. *arXiv preprint arXiv:2503.03686*.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. 2025. Multi-agent architecture search via agentic supernet. *arXiv preprint arXiv:2502.04180*.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bangbang Liu, Yuyu Luo, and Chenglin Wu. 2024. Aflow: Automating agentic workflow generation. *ArXiv*, abs/2410.10762.

A An example of SFT data

After our data construction pipeline, we obtained 4,268 curated query–workflow pairs for model training.

sample

```
"instruction": "You are a
    reasoning assistant
    specialized in building valid
    ComfyUI workflows step by
    step. For a given creative or
    multimodal generation task,
    you must first reason which
    nodes are required and how
    they connect to form a valid
    workflow. The top-level
    structure of your output must
    be exactly:
 1. <thinking></thinking>
     enclosing the entire
     reasoning process. Inside
     <thinking>, include all
     reasoning, and for each
     proposed node, wrap its
     single line of valid Python
     code inside <node></node> for
     validation. Each <node> can
     only contain one line of
     code. After a <node>, if a
     <result>...</result> section
     appears, it shows the
     validation feedback. If no
     <result> appears, the node
     has passed validation.
 2. <workflow></workflow>
     enclosing the final workflow.
     This should contain only the
     validated Python code lines
     in the proper order, one line
     per node. Do NOT include any
     <thinking>, <node>, <result>,
     or any other tags inside
     <workflow>.
Always follow this structure: the
    top-level output must start
    with <thinking>...</thinking>
    followed immediately by
    <workflow>...</workflow>, with
    no other wrapping or nested
    top-level elements. Each
    Python line in <workflow> must
    be valid and executable.",

"query": "You are given an image
    `low_res_image.png`, which is
    a low-resolution photo of a
    cityscape at sunset. Enhance
    the resolution of the image
    using the best available
    upscaling method to create a
    high-quality, detailed version
    of the same cityscape. The
    result should be a
    high-resolution image that
    maintains the original colors
    and details.",

"Reasoning":
<thinking>The user's goal is to
    upscale a low-resolution
    cityscape image to
    high-resolution while
    preserving colors and details.
    The key requirements are
    high-quality upscaling, color
    fidelity, and detail
    preservation. The workflow
    uses a Stable Diffusion-based
    approach with text
    conditioning for
    guidance.<node><result>...The
    CheckpointLoaderSimple node is
    required to load the
    foundational model components
    (diffusion model, CLIP
    encoder, and VAE) necessary
    for the upscaling process,
    ensuring compatibility with
    Stable Diffusion-based
    methods.Data flows from
    loading the model and image,
    through text conditioning, to
    the upscaling process, and
    finally to saving the result.
    Each node is critical for
    achieving the desired outcome.
</thinking>


<workflow>
checkpointloadersimple_0_model,
    checkpointloadersimple_0_clip,
    checkpointloadersimple_0_vae =
    CheckpointLoaderSimple()...
   SaveImage(images=
   ultimatesdupscale_0_image)
</workflow>"
```

B GRPO

For each query Q, we sample a group of K trajectories $\{\tau^{(k)}\}_{k=1}^{K}$ (generated by the validate–repair transition and optional branching), and obtain their terminal rewards $\{R(\tau^{(k)})\}$. We compute a group-relative advantage:

$$\hat{A}^{(k)} \triangleq R(\tau^{(k)}) - \frac{1}{K}\sum_{j=1}^{K} R(\tau^{(j)}). \tag{12}$$

The GRPO objective with KL regularization is:

$$\max_{\theta} \mathbb{E}_{k,t}\Big[\hat{A}^{(k)} \log \pi_\theta\big(a_t^{(k)} \mid s_t^{(k)}\big)\Big] - \lambda_{\mathrm{KL}}\, \mathrm{KL}\big(\pi_\theta \,\|\, \pi_{\mathrm{ref}}\big), \tag{13}$$

where π_{ref} is the reference policy (initialized from SFT) and $\lambda_{\mathrm{KL}} > 0$ controls update stability.

C Notation.

We conclude the commonly used notations for reference.

- $w_g^\star$: The canonical workflow selected for a query group Q_g by maximizing the sum of correctness scores $s(w, Q)$ over workflows W_g that succeed for at least one query in the group.
- s_t: The state in the MDP formulation, consisting of the query Q, current partial workflow G_t, and history h_t of recent diagnostics and constraints.
- Δ_t: An atomic graph edit parsed deterministically from the policy's textual action a_t via the parser Φ.
- G_{t+1}, y_t, m_t: The tool-mediated transition output from VALIDATE, where $y_t \in \{0, 1\}$ indicates acceptance and m_t is the diagnostic message (empty if accepted).
- h_t: Summarizes recent diagnostic information and accumulated constraints. Its presence ensures that the construction process in tool-mediated environments possesses the Markov property.
- y_t: The state-aware validation acceptance indicator, composing intrinsic validity INT and composability COMP.
- m_t: The diagnostic message specifying the violated constraint if validation fails.
- a_t^{rep}: The repaired action proposed by the policy conditioned on the diagnostic m_t if validation fails.
- Δ_t^{rep}: The repaired edit parsed from the repaired action a_t^{rep}.
- $\tilde{A}_t$: The constrained candidate action set at step t.
- $p_t(a)$: The policy probability for action a given state s_t.
- H_t: The normalized entropy of the policy over candidate actions $\tilde{A}_t$, measuring step-level uncertainty.
- ΔH_t: The entropy increase from the previous step, used to trigger branching.
- P_t^{branch}: The branching probability based on entropy increase, using sigmoid with parameters α and β.
- τ_b: The branching threshold; if $P_t^{\text{branch}} > \tau_b$, spawn an additional branch.
- $\mathcal{L}_{\text{SFT}}(\theta)$: The supervised fine-tuning loss over training tuples of queries Q, teacher traces T, and target workflows $G^\star$.
- R_f: The format validity reward component, requiring well-formed trace and workflow blocks.
- R_c: The trace-workflow consistency reward component, ensuring node types in the workflow appear in the trace.
- $R_s(G_T, G^\star)$: The node-type recall for structural accuracy, measuring overlap in node types between generated G_T and target $G^\star$.
- $R(\tau)$: The hierarchical terminal reward, combining format R_f, consistency R_c, and structural R_s with veto penalties for invalid outputs.

D Qualitative Results

Figure 5 shows a qualitative comparison between our method and ComfyMind, illustrating that our method achieves better task completion and higher-quality outputs.

E Evaluation

We follow ComfyBench's evaluation—using GPT-4o for assessment—with text-to-image as an example.

Evaluation prompt

```
'''
You are an expert in image and
    video generation, familiar
    with the latest tasks and
    techniques. You are capable of
    understanding the task
    instruction, analyzing the
    generation result, and
    providing an accurate
    evaluation. Now you are
    evaluating the result of a
    text-to-image generation task.
    You should be tolerant to the
    quality of the generation
    result, and focus on the
    consistency with the
    instruction.
```

Figure 5: Qualitative comparison between our method and ComfyMind

```
The task instruction is described
    as: {instruction}

The given image is the generation
    result, with an actual
    resolution of
    {result_resolution}.

First, analyze whether the
    generation result meets each
    key point in the instruction.
    Enclose your analysis in the
    <analysis> tag. For example:
    <analysis>There is a cat in an
    astronaut suit, which is
    consistent with the
    instruction. The wall is
    white, which is different from
    the "green wall" in the
    instruction.</analysis>.

Then, provide a final judgment of
    whether the generation result
    complies with the instruction.
    The judgment should either be
    "True" or "False". Enclose
    your judgment in the
    <judgment> tag. For example:
    <judgment>False</judgment>.
'''
```

We also performed robustness checks using alternative LLM judges, as shown in Table 6.

Method	GPT-4o	Gemini-2.5-Flash	Qwen2.5-Max
Ours	**71.5**	**73.0**	**71.5**
CA	25.5	27.5	26.5

Table 6: Performance comparison across different evaluators. CA denates ComfyAgent.

Method	Prompt	Completion	Resolve
Ours	$12{,}047 \pm 2\%$	$288{,}450 \pm 2\%$	71.5
CM	$1{,}290{,}201 \pm 2\%$	$112{,}878 \pm 2\%$	64.0

Table 7: Efficiency comparison on ComfyBench (mean $\pm$ std). CM denotes ComfyMind. Token counts are reported for both prompt and completion phases.

F Efficiency

We sampled 20 representative instances to illustrate efficiency trends on tasks successfully solved by both methods. This allows a fair comparison of inference efficiency while avoiding misleading overhead measurements caused by failure cases. We have expanded the cost analysis to include all ComfyBench (CVPR 2025) instances, with results presented in Table 7.

G Model ablation

To evaluate the dependence on the strong teacher model, we conducted ablation experiments using alternative teachers of varying strength. As

Teacher	Resolve (%)
GPT-4o	71.5
Qwen2.5-Max	**73.0**
Qwen2.5-72B	70.5
Qwen2.5-32B	69.5

Table 8: Ablation on teacher model capacity.

Rollout Count	3	5	6	8	10
Resolve (%)	65.0	71.5	72.0	72.5	74.0

Table 9: Sensitivity analysis on the number of rollouts.

shown in Table 8, We observe that performance differences between teacher models are relatively small (within 3.5%). Notably, even when using smaller open-source teachers such as Qwen2.5-32B, the model still achieves competitive performance (69.5%). Importantly, the teacher model is only used to generate intermediate reasoning traces, while the target workflows come from the ground-truth query–workflow pairs. Therefore, the teacher does not define the performance ceiling, but instead helps the model learn structured reasoning patterns.

H Sensitivity Analysis

We conducted experiments to evaluate the impact of varying rollout counts. As shown in Table 9, we observe diminishing returns for rollout ≥ 5, so we chose a rollout budget of 5 for efficiency.

To illustrate the effect of the entropy threshold($\tau_b = 0.5$ in Eq. 8) on performance, we conducted experiments varying this parameter. As shown in Table 10, low thresholds trigger branching too early, before the model has accumulated sufficient uncertainty, resulting in suboptimal exploration. Conversely, high thresholds prevent branching during rollout, reducing the model's ability to explore alternative paths. The range of 0.3–0.5 strikes a balance, yielding the best performance, while thresholds beyond this range lead to a decline in resolve rate.

I Reward Design

Note that our reward focuses on node-type coverage rather than edge-level or parameter-level correctness. This design is intentional because structural validity is already enforced by the state-aware validator during generation (§3.3). Specifically, each graph edit must pass intrinsic validity checks

Threshold (τ_b)	0.2	0.3	0.4	0.5	0.6	0.7
Resolve (%)	67.0	70.0	71.0	**71.5**	69.0	68.0

Table 10: Sensitivity analysis on the entropy threshold (τ_b).

(e.g., operator existence and parameter domains) and composability checks (e.g., port type compatibility and graph invariants) before being committed to the workflow. As a result, any accepted prefix is guaranteed to be executable and structurally consistent. Under this execution-grounded environment, the remaining learning signal mainly concerns whether the correct functional components (node types) are selected for the task. Therefore, node-type recall provides a sufficient and stable structural supervision signal without introducing additional reward complexity.

J Discussion on Long-Horizon Workflow Generation

In ComfyUI workflow construction, even simple pipelines typically contain more than ten nodes with multiple typed dependencies. As a result, the benchmark tasks already represent long-horizon structured generation problems. We report results across difficulty subsets (Vanilla, Complex, Creative), which implicitly correspond to increasing workflow complexity. Moreover, our reasoning-as-action framework is naturally suited for such long-horizon generation: workflows are constructed through sequential graph edits with state-aware validation, which mitigates error accumulation and ensures structural consistency throughout the generation process.