DiG-Plan: Mitigating Early Commitment for Tool-Graph Planning via Diffusion Guidance

Yansi Li , Zhuosheng Zhang*
School of Computer Science, Shanghai Jiao Tong University
yansi_li@sjtu.edu.cn, zhangzs@sjtu.edu.cn

Abstract

Generating executable tool plans requires selecting appropriate subsets from tool libraries, a combinatorial search problem with an exponentially large solution space. However, we identify a critical misalignment in predominant approaches: standard autoregressive (AR) decoding suffers from *early commitment*, where initial token choices rigidly constrain the search trajectory. A controlled study shows that masked denoising raises Pass@10 solution coverage from 0.320 to 0.943 over AR sampling under matched compute. Motivated by this, we propose **DiG-Plan**, a framework that decouples combinatorial exploration from structural refinement. DiG-Plan employs a diffusion-based proposer to generate diverse tool sets via iterative refinement, followed by an AR refiner for dependency prediction. On TaskBench, DiG-Plan improves over AR baselines by a 10% relative margin, with the largest gains on complex compositional tasks; API-Bank results show that the propose-refine-select design remains effective across domains. Code is available at https://github.com/puddingyeah/DiG-Plan.

1 Introduction

Tool-augmented large language models (LLMs) have evolved from solving simple queries to addressing complex, multi-step problems that require composing explicit executable plans [Yao *et al.*, 2023b; Qin *et al.*, 2024]. Unlike single-tool invocation, real-world planning involves selecting an appropriate subset of tools from large libraries and determining their execution order [Shen *et al.*, 2024]. This problem is fundamentally combinatorial: with a library of N tools, the subset-hypothesis space contains 2^N possible tool sets. We do not assume exhaustive enumeration; the challenge is effective search in this large discrete space as libraries scale [Shen *et al.*, 2024; Li *et al.*, 2023; Liu *et al.*, 2024].

*Corresponding author. This work was supported by the National Key R&D Program of China (No. 2024YFC3306500), the National Natural Science Foundation of China (62406188), and the Shanghai Municipal Special Program for Basic Research on General AI Foundation Models (2025SHZDZX025G08).

Despite the combinatorial nature of this task, predominant approaches treat tool planning as a sequential text generation problem, relying on standard autoregressive (AR) decoding [Yao *et al.*, 2023b; Yao *et al.*, 2023a]. While AR models have proven highly effective for text generation, we identify a critical misalignment when applied to combinatorial search: *early commitment*. Because AR generation proceeds strictly left-to-right, early token choices establish a prefix that constrains all subsequent decisions. Once the model commits to an initial tool selection, the probability distribution over remaining tools becomes conditioned on this prefix, limiting the model's ability to explore alternative tool combinations even under stochastic sampling.

This prefix constraint creates a form of *path dependency* in the search process. Early choices determine the trajectory through the search space, effectively pruning large regions of the combinatorial space before they can be explored. For instance, we observe cases where AR models initially select generic vision tools for document-understanding tasks, failing to discover plans requiring more specialized document or visual question-answering tools even across multiple samples with varied sampling strategies.

Crucially, we find that this limitation is resistant to simply increasing sampling diversity. Even with extensive stochastic sampling, AR models fail to recover the missed plans, indicating that the constraint is intrinsic to the sequential decoding mechanism itself rather than a lack of randomness.

We designed a synthetic tool-selection task to strictly disentangle the decoding mechanism from other variables, such as model architecture and pretraining data. By fixing the output length and matching model capacity, we find that under identical compute budgets, masked denoising raises Pass@10 solution coverage from 0.320 to 0.943 compared with AR sampling. This provides empirical evidence that the limitation is intrinsic to the sequential decoding strategy itself.

Motivated by this insight, we propose **DiG-Plan** (**Di**ffusion-**G**uided **Plan**ning), a framework that addresses early commitment by *decoupling* combinatorial exploration from structural refinement. DiG-Plan operates in three stages: first, a diffusion-based proposer generates diverse tool sets via iterative refinement, leveraging global context to revise early decisions; second, a shared autoregressive refiner predicts dependency structures conditioned on each proposed tool set, eliminating the combinatorial search problem in joint gener-

ation; finally, an inference-time judge-free value function selects the best candidate using only deployable features, without calling an external LLM judge.

On TaskBench, DiG-Plan improves over AR baselines by a 10% relative margin, with the largest gains on complex compositional tasks; API-Bank results show that the propose-refine-select design remains effective across domains. Diagnostic analyses show that diffusion proposers improve best-of-K quality from 0.735 to 0.787 and union precision from 0.575 to 0.692 compared with AR proposers under matched compute budgets. Importantly, systematic sweeps over AR sampling parameters confirm that this advantage cannot be replicated by simply increasing sampling randomness, validating that the benefit stems from the iterative refinement mechanism rather than increased stochasticity.

Our main contributions are as follows:

- We identify early commitment in autoregressive tool planning as a key barrier to combinatorial tool-set exploration, and validate this effect with a controlled fixed-length study showing that masked denoising covers substantially more valid tool-set hypotheses under matched compute.
- We introduce DiG-Plan, a propose-refine-select framework that separates tool-set exploration from dependency refinement: diffusion proposal searches over candidate tool subsets, autoregressive refinement predicts edges conditioned on each subset, and an inference-time judge-free value function selects among candidate graphs using deployable features.
- Through experiments on TaskBench, we show that DiG-Plan improves tool-set prediction over AR baselines by 10%, while API-Bank results indicate cross-domain applicability. Candidate-pool diagnostics, AR sampling sweeps, and AR-beam comparisons confirm that the gains come from stronger proposal quality rather than simply increased stochasticity.

2 Related Work

2.1 Tool-Augmented LLMs

Tool-augmented systems are typically formulated as reasoning-and-acting pipelines or tool-invocation learners [Yao *et al.*, 2023b; Qin *et al.*, 2024], often employing planner-centric orchestration for error correction [Wei *et al.*, 2025; Ma *et al.*, 2025]. Recent benchmarks like MetaTool and Toolink further diagnose tool choice behaviors [Huang *et al.*, 2024; Qian *et al.*, 2024]. However, these methods predominantly rely on autoregressive decoding, which we show suffers from early commitment when the task requires exploring a combinatorial space of tool subsets.

2.2 Planning Benchmarks

Current benchmarks, including TaskBench, API-Bank, and AgentBench [Shen *et al.*, 2024; Li *et al.*, 2023; Liu *et al.*, 2024], have evolved to evaluate end-to-end planning capabilities. Since these metrics primarily prioritize final execution accuracy, the specific impact of decoding strategies on candidate diversity remains underexplored. We complement these benchmarks by focusing on their compositional subsets, conducting controlled studies to analyze how different decoding mechanisms influence exploration.

2.3 Multi-Sample Reasoning

Search-based prompting methods (e.g., Tree of Thoughts, Self-Consistency) and graph-structured reasoning [Yao *et al.*, 2023a; Wang *et al.*, 2023; Besta *et al.*, 2024; Pandey *et al.*, 2025] explore candidate sets to improve reasoning. These approaches effectively operate as search strategies atop autoregressive models. However, our experiments suggest that their performance is naturally bounded by the underlying proposal distribution. If prefix dependencies limit diversity, increasing the sampling budget offers limited gains. DiG-Plan addresses this limitation at the proposal stage by adopting diffusion models for non-autoregressive candidate generation.

2.4 Diffusion and Denoising Language Models

Diffusion models have introduced iterative refinement to text generation and discrete sequence modeling [Austin *et al.*, 2021; Li *et al.*, 2022; Nie *et al.*, 2025; Ye *et al.*, 2025]. Most prior work has applied this capability to controllable generation or infilling tasks. Here, we extend it to combinatorial tool-set exploration. The global visibility of diffusion models facilitates non-sequential revisions, which is particularly helpful for handling complex inter-tool dependencies that are often challenging for strictly left-to-right generation.

3 Problem Setup

We now formalize the tool-graph planning problem and establish the evaluation framework.

3.1 Tool-Graph Planning

A tool library $\mathcal{T}$ contains tools where each tool $t \in \mathcal{T}$ has a name and a short description. Given a natural-language instruction x, the goal is to predict a tool graph $\hat{G} = (\hat{V}, \hat{E})$, where $\hat{V} \subseteq \mathcal{T}$ is the selected tool set and $\hat{E} \subseteq \hat{V} \times \hat{V}$ is a set of directed dependency edges. A directed edge $(u \to v) \in \hat{E}$ indicates that tool v should be executed after tool u.

Evaluation compares predictions against a ground-truth tool graph $G^\star = (V^\star, E^\star)$ provided by the dataset [Shen *et al.*, 2024]. Three factors make this problem challenging: (i) the tool library can be large, with a combinatorial search space of $2^{|\mathcal{T}|}$ possible subsets; (ii) instructions are often underspecified; and (iii) multiple valid plans may exist, but evaluation uses a single ground-truth reference.

3.2 Metrics

Tool-Set F1

We denote this metric as ToolF1. Given predicted tool set $\hat{V}$ (canonicalized against $\mathcal{T}$) and ground-truth $V^\star$:

$$P_V = \frac{|\hat{V} \cap V^\star|}{\max(|\hat{V}|, 1)}, \quad R_V = \frac{|\hat{V} \cap V^\star|}{\max(|V^\star|, 1)},$$
$$\text{ToolF1}(\hat{V}, V^\star) = \frac{2P_V R_V}{\max(P_V + R_V, \epsilon)}, \tag{1}$$

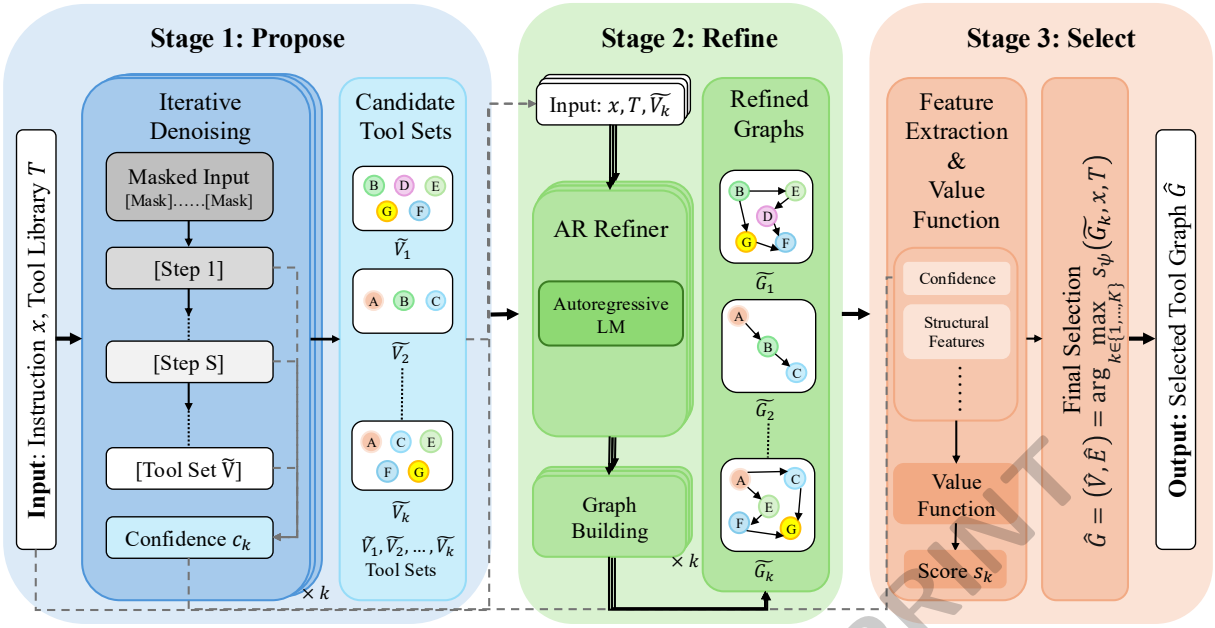


Figure 1: DiG-Plan overview. (1) Propose: A diffusion-based model generates diverse candidate tool sets via iterative refinement; (2) Refine: A shared AR model predicts dependency edges for each fixed tool set; and (3) Select: An inference-time judge-free value function identifies the optimal plan using only deployable features, avoiding external LLM judging at deployment.

where $\epsilon = 10^{-8}$ ensures numerical stability.

Edge Recall

We denote this metric as EdgeRec. Given predicted edge set $\hat{E}$ (restricted to endpoints in $\hat{V}$) and ground-truth $E^\star$: when $|E^\star| = 0$, EdgeRec is 1 if $|\hat{E}| = 0$ and 0 otherwise; when $|E^\star| > 0$,

$$\text{EdgeRec}(\hat{E}, E^\star) = \frac{|\hat{E} \cap E^\star|}{|E^\star|}. \tag{2}$$

Candidate-Pool Metrics

To diagnose candidate diversity and upper-bound quality, we use three complementary metrics. Given a pool of K candidates $\{(\tilde{V}_k, \tilde{E}_k)\}_{k=1}^K$:

$$\text{Pass@K} = \text{UnionRecall@K} = \frac{\left|\left(\cup_{k=1}^K \tilde{V}_k\right) \cap V^\star\right|}{\max(|V^\star|, 1)}, \tag{3}$$

$$\text{UnionPrec@K} = \frac{\left|\left(\cup_{k=1}^K \tilde{V}_k\right) \cap V^\star\right|}{\max\left(\left|\cup_{k=1}^K \tilde{V}_k\right|, 1\right)}, \tag{4}$$

$$\text{Oracle@K} = \max_{1 \le k \le K} \text{ToolF1}(\tilde{V}_k, V^\star). \tag{5}$$

Pass@K quantifies the aggregate *coverage* of the generated pool by computing the union recall against the ground truth. A high value confirms that the model has effectively explored the relevant search space. Oracle@K establishes the performance *upper bound* (best-of-K) by selecting the optimal candidate via ground-truth labels. The difference between these two metrics isolates the error source: low Pass@K implies poor exploration, while a large gap to the *upper bound* indicates poor selection.

Executability

We define a lightweight predicate, $\text{ExecOK}(\hat{G}) \in \{0, 1\}$, to enforce structural validity using the predicted graph. It returns 1 only if three conditions are met: (i) all edge endpoints lie in the predicted node set, with no self-loops or duplicate edges; (ii) for chain instances, edges form a single path visiting all nodes; and (iii) for DAG tasks, the graph contains at least one sink and no isolated nodes. We apply this strictly as an optional filter during selection to ensure deployability.

3.3 Motivating Evidence: A Controlled Study

To validate the early commitment hypothesis, we construct a controlled tool-set prediction task that isolates the combinatorial search component. We define a tool universe $\mathcal{T} = \{t_1, \ldots, t_{23}\}$ and represent tool subsets as fixed-length 23-bit vectors. Each sample provides a symbolic graph specification (sources, sinks, and a noisy depth hint), and the model predicts which tools are needed. This task has the same combinatorial structure: identifying a sparse subset from 2^{23} possible combinations.

We control three key factors: (i) *model capacity* by using identical small Transformer backbones with 2 layers, 128 hidden dimensions, and 4 attention heads for both AR and denoising models; (ii) *training data* by generating a shared synthetic corpus from the same distribution; and (iii) *output format* by fixing the output length to 23 bits, removing any advantage from variable-length generation. The only manipulated factor is the decoding family: prefix AR through

Model	k=1 ToolF1	Pass@10	Oracle@10
Greedy AR	0.355±0.00	0.320±0.00	0.355±0.00
Masked denoising	0.349±0.04	0.943±0.02	0.566±0.01

Table 1: Controlled tool-set playground with 23-bit outputs and $K=10$ candidates. Values are mean±std over 6 seeds.

next-token prediction versus masked denoising, which randomly masks 30% of bits and predicts them conditioned on unmasked context.

We generate $K=10$ candidates per input using temperature sampling for AR and stochastic denoising for masked models, then evaluate Pass@K and Oracle@K.

Table 1 shows that masked denoising raises Pass@10 coverage from 0.320 to 0.943 and Oracle@10 from 0.355 to 0.566, while single-candidate quality remains comparable. With 10 candidates, masked denoising covers 94% of the ground-truth tools on average, while AR covers only 32%. Since model capacity, training data, and output format are controlled, this result provides direct evidence that iterative denoising explores a broader region of the tool-set space than prefix AR decoding.

Implications. This controlled study provides empirical evidence that AR's sequential generation limits exploration in combinatorial search spaces. Motivated by this finding, we propose DiG-Plan (Section 4), which leverages diffusion language models for diverse candidate proposal while using AR for structural refinement.

4 Method: DiG-Plan

4.1 Probabilistic Decomposition

The controlled study in Section 3.3 demonstrates that diffusion-based decoding achieves higher exploration diversity than AR in combinatorial search spaces. Motivated by this finding, DiG-Plan follows a propose-refine-select paradigm that separates tool selection from dependency prediction. Given an instruction x and a tool library $\mathcal{T}$, the framework operates in three stages (Figure 1). First, generate K diverse candidate tool sets $\{\tilde{V}_k\}_{k=1}^{K}$ through diffusion-based proposal. Second, refine each candidate into a directed tool graph $\tilde{G}_k = (\tilde{V}_k, \tilde{E}_k)$ by predicting dependency edges with autoregressive decoding. Third, select a single plan $\hat{G}$ using a deployable value function.

DiG-Plan splits tool-graph planning into two stages:

$$\text{(Tool Set)} \quad \tilde{V} \sim p_\phi(V \mid x, \mathcal{T}), \tag{6}$$

$$\text{(Dependency)} \quad \tilde{E} \sim p_\theta(E \mid x, \mathcal{T}, \tilde{V}), \tag{7}$$

where ϕ denotes the proposer parameters and θ denotes the refiner parameters. The first stage explores diverse tool combinations using diffusion language models. The second stage predicts dependencies using autoregressive decoding, which naturally captures sequential structure. The first stage focuses purely on exploring diverse tool combinations, while the second stage focuses on structured prediction given a fixed tool set. This separation allows using diffusion language models for the first stage and autoregressive decoding for the second stage.

4.2 Diffusion Proposer

As established in Section 1, autoregressive decoding suffers from early commitment in combinatorial search spaces. Diffusion language models [Austin *et al.*, 2021; Li *et al.*, 2022; Nie *et al.*, 2025; Ye *et al.*, 2025] offer an alternative: they iteratively refine a complete output through multiple denoising steps, allowing the model to revise early decisions.

The proposer defines a distribution over tool subsets, instantiated with a diffusion language model. Stochastic sampling draws K candidates $\{\tilde{V}_1, \ldots, \tilde{V}_K\}$ under fixed compute budget. Our controlled experiments (Section 3.3) show that diffusion models maintain higher hypothesis diversity than AR decoding under matched budgets. We compare against AR proposers with temperature and nucleus sampling sweeps in Section 5. Even with top-p=1.0, AR does not match diffusion's exploration quality.

For diffusion proposers exposing per-step token entropies, we derive plan-level confidence by aggregating entropy over the diffusion trajectory. Let $H^{(s)}$ denote mean entropy over masked positions at step s. We discard the first 3 burn-in steps and define $\bar{H} = \frac{1}{S-S_0}\sum_{s=S_0+1}^{S} H^{(s)}$ and $c = \frac{1}{1+\bar{H}} \in (0, 1]$ as input to the value function, where $S_0 = 3$.

4.3 Autoregressive Refiner

While diffusion language models excel at exploring discrete combinatorial spaces, autoregressive decoding naturally captures sequential dependencies through its left-to-right generation process. By conditioning on a fixed tool set $\tilde{V}_k$, we eliminate the combinatorial search problem that causes early commitment.

For each candidate $\tilde{V}_k$, an autoregressive refiner predicts dependency structure $\tilde{E}_k$ conditioned on $(x, \mathcal{T}, \tilde{V}_k)$. The refiner is shared across all proposers to isolate tool-set exploration effects. It takes as input the instruction x, the tool library $\mathcal{T}$, and the proposed tool set $\tilde{V}_k$, then generates a structured output specifying dependencies between tools. The output is parsed into tool graph $\tilde{G}_k = (\tilde{V}_k, \tilde{E}_k)$ after canonicalizing tool names and filtering edges.

4.4 Judge-Free Selection via a Value Function

Best-of-K gains require a selector to choose among candidates. Oracle selection is not deployable, as it requires access to ground-truth labels at inference time. External LLM judges introduce both cost and evaluation confounds. We train a lightweight value function $s_\psi(\tilde{G}, x, \mathcal{T})$ that scores candidate graphs using only deployable, candidate-observable features.

At inference time, we select the highest-scoring candidate:

$$\hat{G} = \arg\max_{k \in \{1,\ldots,K\}} s_\psi(\tilde{G}_k, x, \mathcal{T}). \tag{8}$$

Training objective. On a training split where ground-truth graphs are available, the value function is trained to predict a combined target that trades off tool-set accuracy and edge coverage:

$$y(\tilde{G}, G^\star) = \alpha \cdot \text{ToolF1}(\tilde{V}, V^\star) + (1-\alpha) \cdot \text{EdgeRec}(\tilde{E}, E^\star), \tag{9}$$

Proposer	Refiner	ToolF1 ↑	EdgeRec ↑
Autoregressive and retrieval baselines			
AR-only	–	0.629±0.30	0.200±0.36
AR proposer	AR refiner	0.661±0.32	0.242±0.38
Retriever (top-k=5)	AR refiner	0.569	0.127
Retriever (top-k=10)	AR refiner	0.638	0.156
Retriever (top-k=15)	AR refiner	0.660	0.185
Diffusion proposers			
Dream DLM-only	–	0.128±0.28	0.335±0.47
LLaDA proposer	AR refiner	0.734	0.258
Dream proposer	**AR refiner**	**0.729±0.28**	**0.268±0.39**

Table 2: End-to-end tool-graph planning on TaskBench-23 with N=501 and K=1. We separate the *proposer*, the tool-set generator, from the *refiner*, the edge predictor. Numbers are means over instances, with standard deviations shown when available.

with $\alpha = 0.7$ by default. This weighting prioritizes tool-set accuracy: correct tool selection is required for accurate edge prediction. The predictor s_ψ is fit by regression to y using only candidate-observable features.

Feature design. The input to s_ψ concatenates several feature groups:

- *Proposer confidence* c, derived from diffusion entropy to capture uncertainty.
- *Structural heuristic score* combining three components with weights 0.3/0.3/0.4: structural validity, coverage, and coherence.
- *Predicted counts* (number of tools and edges).
- *DAG-validity flag* computed by topological sorting.
- *Candidate index* k to capture position bias.
- *Multi-hot tool vector* over predicted tool names to capture tool-specific patterns.

In the `graph` feature set, we additionally append a directed edge multi-hot vector over tool pairs. Importantly, no ground-truth-derived quantities are used at inference time, so the selector is fully deployable without oracle access.

4.5 Executability Constraints

When high-scoring candidates are close, exec-filtered selection chooses the highest-scoring candidate satisfying $\mathrm{ExecOK}(\tilde{G}_k)$=1 (as defined in Section 3), falling back to unconstrained top-1 otherwise.

5 Experiments

We evaluate on TaskBench-23 [Shen *et al.*, 2024] and API-Bank [Li *et al.*, 2023], reporting end-to-end tool-graph metrics (ToolF1 and EdgeRec) and candidate-pool diagnostics (Pass@K, Oracle@K).

5.1 Benchmarks

TaskBench-23. TaskBench [Shen *et al.*, 2024] provides a tool-graph planning benchmark with diverse dependency structures. We use N=501 instances, with 167 each for single-tool, chain, and DAG tasks. For candidate-pool analyses, we focus on the compositional subset of chain and DAG instances, yielding N=334 instances.

API-Bank. We convert API-Bank [Li *et al.*, 2023] to TaskBench format with tool nodes and dependency edges, serving as an out-of-domain check.

5.2 Baselines

We compare DiG-Plan against several baselines: **AR-only**, **AR two-stage**, **AR-beam+AR**, **Retriever+AR** (dense retriever with $k \in \{5, 10, 15\}$ combined with AR refiner), and **Diffusion+AR**. The retriever baseline uses a pre-trained dense retriever (all-MiniLM-L6-v2) without task-specific finetuning, computing cosine similarity between instruction embeddings and tool description embeddings to select top-k tools. All two-stage methods share the same AR refiner to isolate the proposer's effect.

5.3 Implementation Details

Model selection. All two-stage methods share the same AR refiner, differing only in the proposer. For end-to-end evaluation we report K=1 in Table 2, while for candidate-pool analyses we generate $K \in \{5, 10\}$ candidates per input. The value function is trained on a disjoint training split with supervision $y = \alpha \cdot \mathrm{ToolF1} + (1-\alpha) \cdot \mathrm{EdgeRec}$ where $\alpha = 0.7$.

Proposer models. Diffusion proposers use Dream 7B [Ye *et al.*, 2025], LLaDA-8B-Instruct [Nie *et al.*, 2025], or LLaDA2.0-mini-preview [Bie *et al.*, 2025]. AR components use Qwen2.5-7B-Instruct [Yang *et al.*, 2024]. The retriever baseline uses all-MiniLM-L6-v2 [Reimers and Gurevych, 2019].

Value function. The value function uses a GradientBoostingRegressor [Pedregosa *et al.*, 2011] with 300 trees, learning rate 0.05, and max depth 4, trained on candidate pools from the training split.

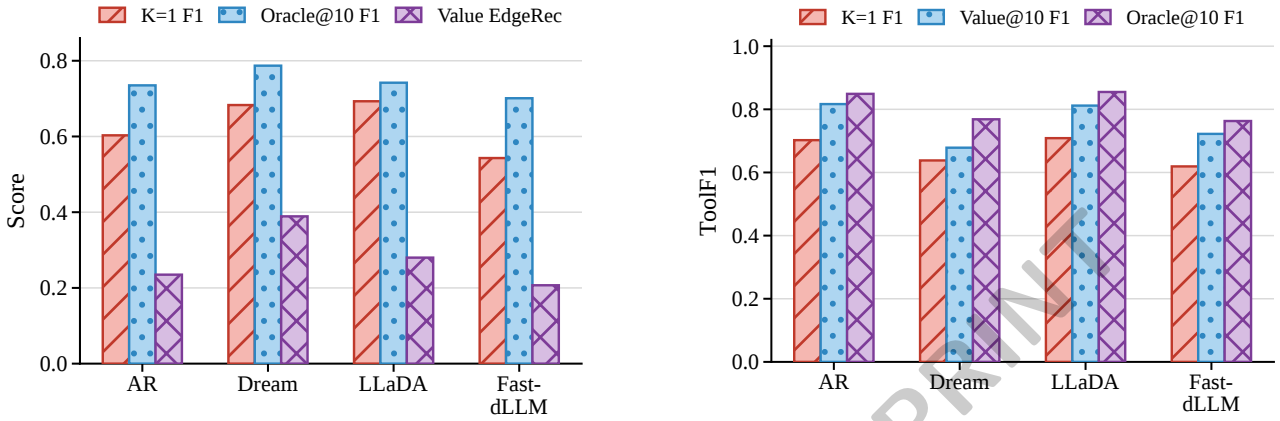
Table 3 presents candidate-pool diagnostics on the held-out chain+DAG subset with N=334 and K=10. Compared with an AR proposer under comparable budgets, Dream raises UnionPrec@10 from 0.575 to 0.692 and Oracle@10 from 0.735 to 0.787, indicating both stronger best-of-K quality and higher-precision coverage.

5.4 Disentangling Proposal Quality from Selection

Table 2 reports end-to-end results on TaskBench-23 with N=501 and K=1. DiG-Plan with Dream proposer achieves

Proposer family	k=1 ToolF1	Oracle@10	UnionRec@10	UnionPrec@10
AR proposer (T=1.3)	0.603	0.735	0.728	0.575
Dream proposer	0.683	0.787	0.759	0.692
LLaDA2 proposer	0.535	0.605	0.547	0.612

Table 3: Candidate-pool diagnostics on the held-out chain+DAG set (N=334) with K=10 candidates per input.



(a) TaskBench-23 proposer comparison

(b) API-Bank cross-domain results

Figure 2: Proposer family comparison. (a) On TaskBench-23 compositional instances with N=334 and K=10, Dream improves over AR on K=1 ToolF1, Oracle@10, and EdgeRec. (b) On API-Bank, LLaDA reaches the strongest Oracle@10 and competitive value-selected ToolF1, averaged over 4 seeds.

Selector	ToolF1 ↑	EdgeRec ↑
K=1	0.685	0.244
Heuristic	0.690	0.311
Value function	**0.716**	**0.309**
Oracle	0.768	0.283

Table 4: Selection on the held-out compositional subset (N=334) with K=5 candidates.

ToolF1 of 0.729 and EdgeRec of 0.268, outperforming the AR two-stage baseline with ToolF1 of 0.661 and EdgeRec of 0.242, corresponding to a 10% relative improvement. Since single-tool instances do not require combinatorial search, performance differences mainly arise on chain and DAG instances; we therefore focus additional analyses on the compositional subset of N=334 instances.

Several trends emerge from these results. First, two-stage methods outperform single-stage AR-only baseline, which achieves ToolF1 of 0.629. Second, among two-stage methods, diffusion proposers outperform AR proposers and retrieval-based proposers. Third, the Dream DLM-only baseline achieves high EdgeRec of 0.335 but low ToolF1 of 0.128. However, this aggregate EdgeRec is driven by single-tool instances: DLM-only achieves near-perfect EdgeRec of 0.994 on single tasks but fails on chain with 0.006 and DAG with 0.004 structures. This shows that end-to-end diffusion generation struggles with precise tool-name grounding and complex dependency prediction on compositional tasks, motivating our two-stage design where diffusion handles tool-set exploration and AR handles edge prediction.

Figure 2 visualizes this advantage across multiple metrics. On the compositional subset (Figure 2a), Dream improves over AR on K=1 ToolF1, Oracle@10, and EdgeRec, matching the candidate-pool gains in Table 3. Figure 2b further shows that the propose-refine-select interface transfers to API-Bank, where LLaDA achieves the strongest Oracle@10 and competitive value-selected ToolF1.

We further test whether the gap is due to weak AR decoding by evaluating an AR-beam proposer under the same held-out protocol, using N=334, K=10, and the same execution-constrained selector. AR-beam reaches ToolF1/EdgeRec of 0.625/0.294, still below Dream's 0.708/0.398, showing that stronger AR search alone does not close the gap.

Having established the effectiveness of the diffusion proposer, we next investigate the impact of the selection mechanism. To disentangle the contributions of proposal quality from selection strategy, we fix a single Dream-proposed candidate pool with the shared AR refiner at $K = 5$ and systematically vary only the selection strategy on the held-out compositional test set of $N = 334$ instances. This experimental design isolates the selector's contribution, allowing us to measure selection quality independently of proposal quality.

Table 4 demonstrates that an inference-time judge-free value function achieves ToolF1 of 0.716 compared to 0.690 for heuristic selection. Measured relative to the K=1 baseline of 0.685, this recovers $0.031/0.083 \approx 37\%$ of the oracle gap to the best-of-5 value of 0.768.

Figure 3 shows (a) best-of-K scaling and (b) AR sampling sweep results. The value function consistently outperforms heuristic selection across all K values, and AR cannot match

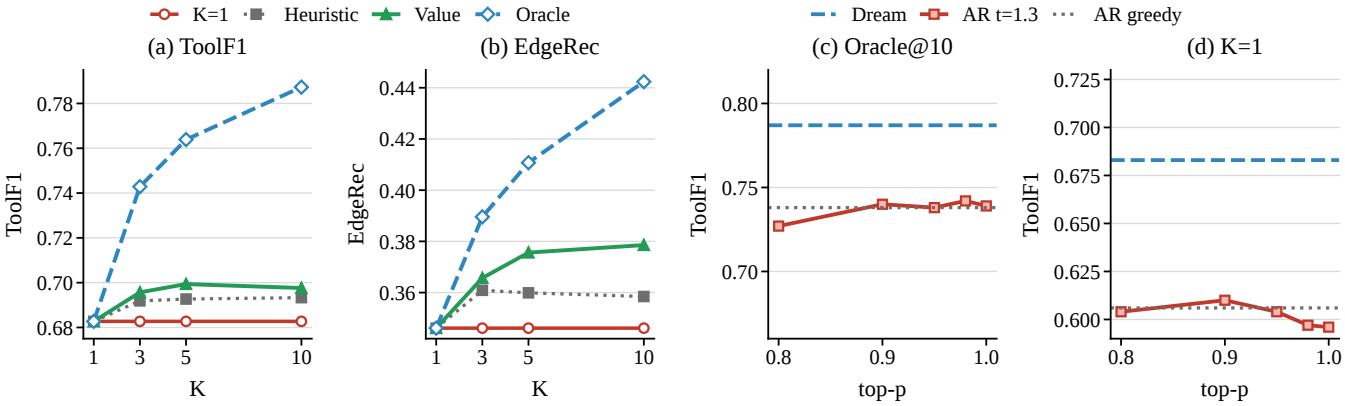


Figure 3: Selection and sampling analysis. (a–b) Performance on the held-out compositional subset with N=334 under a fixed Dream-proposed pool as K varies. (c–d) AR top-p sweep on TaskBench-23 compositional instances with N=334 and K=10.

Type	Task Setup (Instruction & Ground Truth)	Dream (Ours)	AR (Baseline)
Chain	*I have a Spanish quote... translate it, then generate creative text, and finally analyze the document image.* **GT:** Translation, Text Generation, Document QA	Translation, Text Generation, Document QA **(F1: 1.00)**	Translation, Conversational, Image Classification *(F1: 0.33)*
DAG	*I have a long article... summarize it, generate an image, and check whether the image includes a playground.* **GT:** Summarization, Text-to-Image, Visual QA	Summarization, Text-to-Image, Visual QA **(F1: 1.00)**	Image-to-Image, Summarization, Document QA *(F1: 0.33)*

Table 5: Representative case studies comparing Dream and AR. Green text denotes correctly predicted steps matching the Ground Truth (GT), while red text indicates selection errors.

Dream's performance even when higher top-p values increase sampling diversity.

5.5 Error Analysis and Complexity Stratification

We analyze failure patterns using an automatic multi-label taxonomy on the held-out compositional subset of $N = 334$ instances. The taxonomy evaluates the value-function-selected predictions from a Dream proposer pool with $K = 10$ candidates. Error tags are not mutually exclusive, so each predicted graph can receive multiple labels.

Missing edges are the most frequent error tag, affecting 82.9% of samples; missing tools follow at 76.9%. This indicates that coverage remains the primary challenge. Better tool proposals are still useful: Dream raises exact-tool coverage from 0.234 to 0.296 and exact+full-edge coverage from 0.141 to 0.231 over AR. Since the refiner cannot recover edges between missing tools, proposal and refinement quality are complementary. With ground-truth tools fixed, an oracle-refiner diagnostic raises EdgeRec from 0.398 to 0.675, identifying edge refinement as the remaining bottleneck. Extra tools appear in 50.9% of predictions, and overt cyclicity remains rare at 6.9%.

Using $K = 5$ selection from the same Dream pool, value-function selection typically provides gains over heuristic selection, with notable improvements on instances with 3-4 tools where combinatorial search is non-trivial. For example, on chain instances with 3-4 tools, value-function selection achieves ToolF1 of 0.680 and 0.636 respectively, compared to heuristic's 0.661 and 0.632.

5.6 Qualitative Analysis

Table 5 gives representative chain and DAG examples under the same refiner and value-based selector. Dream identifies all required tools in both cases, while AR makes semantically plausible but task-mismatched substitutions such as *Image Classification* for *Document QA* and *Image-to-Image* for *Text-to-Image*. These examples illustrate how early prefix choices can steer AR away from better tool combinations.

6 Conclusion

We presented DiG-Plan, a diffusion-guided propose-refine-select framework for tool-graph planning. The central finding is that prefix autoregressive decoding suffers from early commitment in tool-set search: once early tool choices are sampled, subsequent decisions remain conditioned on that prefix, limiting the diversity of candidate tool sets. DiG-Plan addresses this mismatch by using diffusion for global tool-set proposal, AR decoding for dependency refinement, and an inference-time judge-free value function for candidate selection.

Controlled and benchmark evaluations show stronger candidate coverage and a 10% ToolF1 gain over AR baselines. Candidate-pool diagnostics, AR sweeps, and AR-beam comparisons attribute these gains to proposal quality rather than stochasticity. Future work will strengthen edge refinement.

References

[Austin *et al.*, 2021] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 17981–17993, 2021.

[Besta *et al.*, 2024] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 17682–17690. AAAI Press, 2024.

[Bie *et al.*, 2025] Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Ling Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, Jianfeng Tan, Lanning Wei, Ji-Rong Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Jun Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. Llada2.0: Scaling up diffusion language models to 100b, 2025.

[Huang *et al.*, 2024] Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and Lichao Sun. Metatool benchmark for large language models: Deciding whether to use tools and which to use. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[Li *et al.*, 2022] Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-lm improves controllable text generation. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[Li *et al.*, 2023] Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3102–3116. Association for Computational Linguistics, 2023.

[Liu *et al.*, 2024] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[Ma *et al.*, 2025] Zhiyuan Ma, Jiayu Liu, Xianzhen Luo, Zhenya Huang, Qingfu Zhu, and Wanxiang Che. Advancing tool-augmented large language models via meta-verification and reflection learning. In Luiza Antonie, Jian Pei, Xiaohui Yu, Flavio Chierichetti, Hady W. Lauw, Yizhou Sun, and Srinivasan Parthasarathy, editors, *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, V.2, KDD 2025, Toronto ON, Canada, August 3-7, 2025*, pages 2078–2089. ACM, 2025.

[Nie *et al.*, 2025] Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models, 2025.

[Pandey *et al.*, 2025] Tushar Pandey, Ara Ghukasyan, Oktay Göktas, and Santosh Kumar Radha. Adaptive graph of thoughts: Test-time adaptive reasoning unifying chain, tree, and graph structures. *CoRR*, abs/2502.05078, 2025.

[Pedregosa *et al.*, 2011] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011.

[Qian *et al.*, 2024] Cheng Qian, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. Toolink: Linking toolkit creation and using through chain-of-solving on open-source model. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 831–854. Association for Computational Linguistics, 2024.

[Qin *et al.*, 2024] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[Reimers and Gurevych, 2019] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using

siamese bert-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics, 2019.

[Shen *et al.*, 2024] Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. Taskbench: Benchmarking large language models for task automation. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[Wang *et al.*, 2023] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[Wei *et al.*, 2025] Xiaolong Wei, Yuehu Dong, Xingliang Wang, Xingyu Zhang, Zhejun Zhao, Dongdong Shen, Long Xia, and Dawei Yin. Beyond react: A planner-centric framework for complex tool-augmented LLM reasoning. *CoRR*, abs/2511.10037, 2025.

[Yang *et al.*, 2024] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2024.

[Yao *et al.*, 2023a] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[Yao *et al.*, 2023b] 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, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[Ye *et al.*, 2025] Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models. *CoRR*, abs/2508.15487, 2025.