MANET: TOWARDS NETWORK TASK AUTOMATION WITH MULTI-AGENT INTELLIGENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language agents have demonstrated remarkable versatility in personal computing scenarios and are increasingly expected to operate critical infrastructure. *Network Task Automation* (NTA), the focus of this paper, aims to compile a network operator's natural-language intent into an executable workflow over a proprietary capability catalog, which confronts a small on-premise model with heterogeneous decisions beyond the coverage of pretraining. Existing approaches typically rely on hand-crafted workflows or prompts over frozen models and thus struggle to incorporate execution feedback, while the lack of relevant public data also hinders further advancement. In response to these challenges, we present NETBENCH, the first NTA benchmark that pairs thousands of realistic operator intents with corresponding reference workflows over the network capability catalog. Beyond this dedicated dataset, we further introduce a multi-agent framework (MANET) for NTA, which assigns the heterogeneous network decisions to specialized agent roles and jointly optimizes these roles via reinforcement learning from execution feedback. Empirically, MANET exhibits consistent superiority on NETBENCH, as well as on the open-source benchmark TaskBench for general task automation. The analysis reveals that specialized agents improve steadily on complex intents where a shared policy plateaus, showcasing the effectiveness of multi-agent intelligence in NTA.[1]

1 INTRODUCTION

Recent advances in large language agents (Yao et al., 2023a; Schick et al., 2023; Qin et al., 2023; Shen et al., 2023) have ushered in a new era of autonomous task execution across everyday computing, spanning information retrieval, public service invocation, and code execution with minimal human oversight. Such autonomy, however, rests on a favorable operating environment, wherein tool documentation is publicly available, user requests remain routine, and errors are readily recoverable. These agents are increasingly entrusted with critical infrastructure, yet a dedicated production network inherently violates all three conditions (Manias et al., 2024), giving rise to a critical question of whether such agents can remain reliable once this favorable environment is withdrawn.

This question arises acutely in *Network Task Automation* (NTA), which requires an agent to compile a network operator's natural-language intent into a deployable workflow over a proprietary capability catalog (Tong et al., 2025). Such compilation inherently entails two coupled decisions, the selection of the capabilities that an intent requires and their arrangement into an executable order (Figure 1). Neither decision admits a unique solution, as several distinct workflows typically fulfill the same intent, and a single reference chain is therefore an insufficient criterion of correctness. The admissibility of an order further compounds this difficulty, since the capabilities are bound by dependencies whose violation renders the entire workflow unexecutable. The deployment setting aggravates both difficulties, as privacy and latency requirements confine the agent to a small on-premise model, while an erroneous workflow misconfigures a live network rather than a testbed.

A straightforward approach to NTA applies prompt engineering to frozen models, organized either as a single agent or as a manually designed pipeline of collaborating agents (Wu et al., 2023; Qian et al., 2023; Hong et al., 2024; Guo et al., 2024; Li et al., 2026a). Such prompting typically yields suboptimal workflows, since the dependencies among capabilities are specific to the operator's catalog and seldom represented in pretraining corpora. Although state-of-the-art proprietary models

[1] Code is available at https://anonymous.4open.science/r/MaNet-DA58/.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

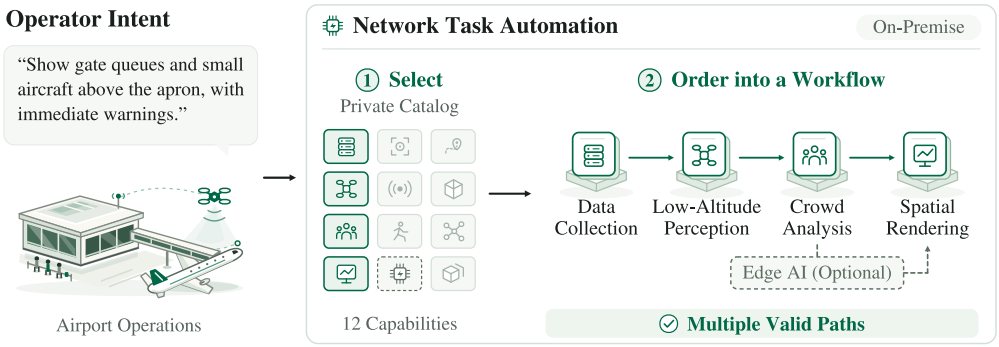

Figure 1: Illustration of Network Task Automation. A network operator's natural-language intent is compiled into an executable workflow over a proprietary capability catalog.

(OpenAI, 2026; Anthropic, 2026) can select the required capabilities, the orders they produce often violate these dependencies and thus fail to execute. A natural remedy lies in training the agents with execution feedback, which nevertheless raises two fundamental challenges. On the one hand, the scarcity of public benchmarks that pair realistic operator intents with executable reference workflows impedes the training and evaluation of NTA agents. On the other hand, the coupled decisions of NTA call for several specialized agents to be jointly optimized towards valid workflows, whereas supervised fine-tuning on reference workflows (Zhang et al., 2023b) tends to fit a single recorded path, which may suppress equally valid alternatives and degrade capability selection.

To address these challenges, we first present NETBENCH, a benchmark built on a proprietary catalog of twelve sensing and computing capabilities with explicit ordering dependencies, wherein thousands of realistic operator intents of varying complexity are each paired with an executable reference workflow. Since a single intent can often be fulfilled by several workflows, NETBENCH also annotates the admissible capability alternatives of each intent to accommodate multiple valid solutions. Beyond this dedicated benchmark, we further introduce MANET, a multi-agent framework that assigns the compilation of an intent to three specialized roles. Specifically, a planner first decomposes the intent into operational steps, an assembler then grounds these steps in catalog capabilities, and a workflower finally arranges them into an executable order. Notably, instead of imitating reference workflows, MANET jointly optimizes these roles through multi-agent reinforcement learning (Shao et al., 2024; Zhao et al., 2026; Feng et al., 2026), in which multi-path verifiable rewards, derived from execution feedback, credit valid decisions beyond the single recorded path.

Empirically, we evaluate MANET on NETBENCH for network operations and on TaskBench (Shen et al., 2024) for general task automation. Experimental results demonstrate that MANET (i) achieves higher workflow executability on NETBENCH than prompt-based pipelines, supervised fine-tuning, and a single agent under the same rewards, (ii) surpasses state-of-the-art proprietary models on complex intents, which suggests that the main difficulty of NTA lies in the ordering of capabilities rather than their selection, and (iii) owes its advantage primarily to the multi-path verifiable rewards instead of elaborate agent arrangements (Zhuge et al., 2024; Zhang et al., 2025b; Li et al., 2026b).

The main contributions of this paper are summarized as follows.

- **NETBENCH**, a dedicated NTA benchmark that pairs thousands of realistic operator intents of varying complexity with executable reference workflows over a proprietary catalog of network capabilities governed by explicit ordering dependencies, and further annotates admissible capability alternatives for each intent to accommodate multiple valid solutions.
- **MANET**, a multi-agent framework that employs three specialized agents to decompose the user intent into steps, select matching capabilities from the proprietary catalog, and arrange them into an executable workflow that satisfies their ordering dependencies, while jointly optimizing these agents via reinforcement learning with multi-path verifiable rewards.
- **Extensive experiments** on NETBENCH and TaskBench, which demonstrate that MANET outperforms prompt-based pipelines, supervised fine-tuning, and single-agent training in workflow executability on NETBENCH, attribute this advantage primarily to multi-path verifiable rewards, and delineate the conditions under which decomposition is beneficial.

2 Related Work

Language agents for tool use and task automation. Prevailing tool-use agents equip a language model with external tools and cast task automation as planning over a tool catalog (Yao et al., 2023a; Schick et al., 2023; Qin et al., 2023; Shen et al., 2023; Ruan et al., 2023; Kong et al., 2023), a formulation that TaskBench (Shen et al., 2024) evaluates as the recovery of a tool graph. A complementary strand replaces prompting with training on self-generated solutions (Zelikman et al., 2022; Yuan et al., 2023; Singh et al., 2024) and reflections on failures (Shinn et al., 2023; Yao et al., 2023b) or against verifiable rewards (Shao et al., 2024; Guo et al., 2025). These approaches, however, rest on a single policy that observes the entire task and is evaluated against a single reference, while benchmarks of role-specialized teams (Yu et al., 2026) overlook operator intents over proprietary catalogs.

Multi-agent systems built from language models. Beyond a single agent, multi-agent systems distribute a task over model instances that communicate in natural language. The mainstream paradigm encodes operating procedures into agent roles and organizes their exchanges as programmable conversations (Wu et al., 2023; Qian et al., 2023; Chen et al., 2023a; Hong et al., 2024), a design since extended to embodied control (Chen et al., 2023b; Mandi et al., 2023; Zhang et al., 2023a) and to debate among peers, where disagreement drives refinement (Du et al., 2024; Liang et al., 2024). More adaptive designs route messages by declared intent instead of along a fixed topology (Li et al., 2026a). As surveys document (Guo et al., 2024; Yan et al., 2025), however, such systems typically keep their participants frozen, so that the team improves through orchestration rather than training.

Optimization of multi-agent systems. Another body of work optimizes the multi-agent system itself. At the system level, methods adapt the agent-graph topology (Zhuge et al., 2024; Zhang et al., 2025b; Li et al., 2026b), prune redundant messages (Zhang et al., 2025a), or refine the orchestration specification behind both (Chen et al., 2026), while latent communication exchanges hidden states instead of text (Pham et al., 2023; Zou et al., 2025; Du et al., 2025; Zheng et al., 2025). At the agent level, reinforcement fine-tuning of role-structured teams (Liao et al., 2025), grounded in classical multi-agent credit assignment (Foerster et al., 2018), has been extended to group rollouts by role and turn (Zhao et al., 2026), normalize advantages per role (Feng et al., 2026), and merge roles into one policy (Zeng et al., 2026), yet these advances are validated mainly on general reasoning benchmarks.

Agents for network and infrastructure operations. Agentic intelligence for operational networks has been articulated mainly at the level of architecture and requirements. A-Core (Tong et al., 2025) envisions agentic coordination for the 6G core network, and intent-based studies convert natural-language requests into controller-ready policies (Manias et al., 2024; Dinh et al., 2025; Hossain & Aljoby, 2026). Closer to execution, tool-augmented agents perform root cause analysis by prompting a frozen model (Wang et al., 2023). Although these studies call for closed-loop optimization, their evidence remains largely confined to proposals and case studies, and public data for compiling operator intents into executable workflows is scarce. NetBench provides such data with multi-path annotations, and MANet brings multi-agent reinforcement learning to this closed loop.

3 NetBench: Benchmarking Network Task Automation

3.1 Task and capability catalog

Formally, NTA maps an operator intent x to a workflow $w = (S, \sigma)$ that fulfills it, where $S \subseteq \mathcal{C}$ is a set of capabilities drawn from a closed catalog $\mathcal{C}$ and σ is an execution order over S. Each intent is phrased as an operator would state it, for instance a request to follow a delivery truck driving erratically through an industrial park and to predict whether it will collide with anything. NetBench instantiates $\mathcal{C}$ with a proprietary service catalog in production use, comprising $|\mathcal{C}| = 12$ sensing and computing capabilities that range from data collection and target detection through crowd-flow analysis and behavior recognition to digital-twin simulation and visualization, each accompanied by a brief description of its function and scope. A precedence relation $\prec \subseteq \mathcal{C} \times \mathcal{C}$ encodes the dependencies among these capabilities, which arise mainly from data flow, since analysis consumes what sensing produces and simulation relies on the model that digitization builds. A workflow is therefore executable only when S fulfills the intent and σ respects every applicable pair of $\prec$.

The design of NetBench is shaped by two inherent properties of NTA. On the one hand, an intent rarely admits a unique capability set, since auxiliary steps such as data collection or final render-

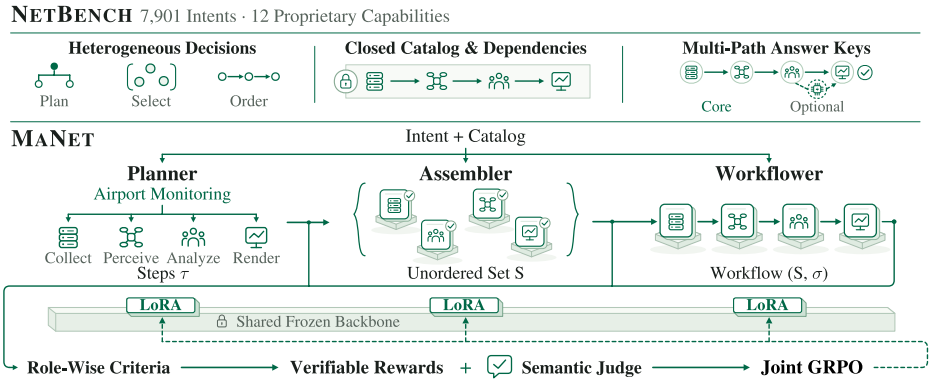

Figure 2: Overview of NETBENCH, which pairs operator intents with multi-path answer keys, and MANET, which jointly optimizes a planner, an assembler, and a workflower with verifiable rewards.

ing may be stated explicitly or left implicit without altering what the workflow accomplishes, and an adjudication of the disagreements between a frontier model and the reference labels finds both answers acceptable in 43% of the cases (Appendix D.3). A benchmark scored against a single reference chain would thus penalize equally valid workflows and favor the imitation of annotator habits. On the other hand, an order is executable only when it satisfies $\prec$, whereas this ordering knowledge is absent from the capability descriptions and, given the proprietary catalog, largely beyond the coverage of pretraining. In contrast to prior tool-use benchmarks that rest on public APIs with a single reference per request (Shen et al., 2024), NETBENCH therefore evaluates both whether a selection is admissible under a multi-path answer key and whether its order respects the precedence relation.

3.2 DATASET CHARACTERISTICS

NETBENCH comprises 7,901 operator intents, each paired with an executable reference workflow $w^\star = (S^\star, \sigma^\star)$ and a multi-path answer key. These intents arise from operational scenes in which the core network orchestrates sensing and computing services, such as transport hubs, industrial sites, crowded venues, and low-altitude inspection. A lexical gate further ensures that no intent names a capability or an operation verb from the catalog, so that the required capabilities need to be inferred from the semantics of the scene rather than recovered by surface matching. Although natural operator requests rarely involve more than four capabilities, the reference workflows extend to seven, as longer intents are additionally constructed to capture the compound demands that arise when several services interact within one scene. The training split contains 6,701 intents, while the validation and test splits contain 200 and 1,000 intents, respectively. The test split is evenly divided between short intents with $|S^\star| \leq 4$ and long intents with $|S^\star| \geq 5$, and results are reported separately for the two halves. Figure 2 and Appendix D summarize the catalog, the length distribution, and the splits.

3.3 MULTI-PATH ANSWER KEY AND EVALUATION PROTOCOL

Rather than a single reference, NETBENCH equips each intent x with a multi-path answer key that partitions the catalog into three parts, wherein a core set $\mathcal{K}_{\mathrm{c}}(x)$ contains the capabilities that every valid workflow requires, an optional set $\mathcal{K}_{\mathrm{o}}(x)$ contains dispensable additions, and the remaining capabilities are regarded as invalid. The key is determined by a majority vote among LLM annotators given the intent, the catalog, and one known valid workflow, such that the reference chain remains valid under the resulting key while alternative workflows beyond it are credited as well. Optional capabilities account for roughly a third of a typical reference workflow, which reflects the considerable latitude in composing a valid workflow and hence the need for such a key.

On the basis of this answer key, the selection score and the end-to-end success of a selected set $S \subseteq \mathcal{C}$ under an order σ over S are defined as

$$\mathrm{Sel}(S) \;=\; \frac{|S \cap \mathcal{K}_{\mathrm{c}}|}{|\mathcal{K}_{\mathrm{c}}|} \cdot \frac{|S \cap (\mathcal{K}_{\mathrm{c}} \cup \mathcal{K}_{\mathrm{o}})|}{|S|}, \qquad \mathrm{Succ}(S, \sigma) \;=\; \mathbf{1}[\mathrm{Sel}(S) = 1] \cdot \mathbf{1}[\sigma \models \prec], \quad (1)$$

where $\sigma \models \prec$ indicates that σ satisfies every precedence pair applicable within S, and the argument x of both keys is omitted for brevity. The selection score multiplies the recall of the core set by the precision of the selection, with optional capabilities counted as correct, so that a workflow receives

full credit, namely $\mathrm{Sel}(S) = 1$, whenever it covers the core set without invalid additions. The end-to-end success Succ further requires an admissible order and thus couples selection with ordering as an actual deployment does. For comparability with prior tool-use benchmarks (Shen et al., 2024), we additionally report node-level and edge-level F1 (n-F1 and e-F1) against the reference chain, although both metrics undervalue valid alternatives. An LLM-based judge (Gu et al., 2024) that assesses whether the workflow fulfills the intent also serves as an independent semantic check.

4 MANET: A MULTI-AGENT FRAMEWORK FOR NTA

4.1 PROBLEM FORMULATION

Let x denote an operator intent, $\mathcal{C}$ the capability catalog, and $\prec$ the precedence relation over it. The compilation of x into an executable workflow entails three decisions of distinct nature, wherein a plan τ decomposes the intent into a sequence of natural-language steps, a selection $S \subseteq \mathcal{C}$ identifies the capabilities that these steps require, and an order σ arranges S into a workflow $w = (S, \sigma)$. MANET entrusts these decisions, in that order, to three roles $k \in \mathcal{R} = \{\mathrm{P}, \mathrm{A}, \mathrm{W}\}$, each conditioned on the intent and, where applicable, on the artifact produced immediately upstream, so that the joint policy factorizes as

$$\pi_\theta(\tau, S, \sigma \mid x) \;=\; \pi_{\theta_\mathrm{P}}(\tau \mid x)\, \pi_{\theta_\mathrm{A}}(S \mid x, \tau)\, \pi_{\theta_\mathrm{W}}(\sigma \mid x, S), \tag{2}$$

with $\theta = (\theta_\mathrm{P}, \theta_\mathrm{A}, \theta_\mathrm{W})$ collecting the parameters of the three roles. Since the answer key of Section 3.3 credits the workflows that an intent legitimately admits, the learning objective is to maximize the expected end-to-end success

$$J(\theta) \;=\; \mathbb{E}_x\, \mathbb{E}_{(\tau,S,\sigma)\sim\pi_\theta(\cdot|x)}\big[\mathrm{Succ}(S, \sigma)\big], \tag{3}$$

which treats admissible workflows alike and assigns no value to inadmissible ones. MANET renders this objective more amenable to optimization through a decomposition into specialized roles together with role-level rewards derived from the answer key rather than from a single reference path.

4.2 FRAMEWORK DESIGN

MANET instantiates the factorization of Equation (2) with three specialized agents (Figure 2), wherein a *planner* (P) decomposes the intent into steps, an *assembler* (A) grounds these steps in a capability set, and a *workflower* (W) arranges this set into an executable order. This division follows the structure of the task rather than a convenience of implementation, since the three decisions draw on the operator's intention, the content of the catalog, and the dependencies among capabilities, respectively. Each decision also fails in its own manner, through a plan too coarse to ground, a selection that misses or exceeds what the intent requires, or an order that violates the dependencies. A monolithic policy that emits an entire workflow receives a single verdict on intertwined decisions and can hardly attribute credit to any one of them, whereas separate roles keep the verdicts apart.

The roles communicate through structured artifacts that serve as contracts between them, such that a plan is a list of steps, a selection is an unordered set of catalog identifiers, and a workflow is an order over that set. Each artifact can therefore be validated before it travels downstream and provides a natural surface on which the reward of its author is defined. The allocation of capacity across roles constitutes a design dimension orthogonal to this decomposition, as the roles may either share a single set of parameters, which confines deployment to the footprint of one model, or hold parameters of their own at a proportional cost in memory. Since neither the decomposition, the contracts, nor the rewards depend on this choice, MANET accommodates both configurations without modification.

4.3 JOINT REINFORCEMENT LEARNING WITH MULTI-PATH VERIFIABLE REWARDS

Reward design is central to this framework, since a policy trained with reinforcement learning tends to acquire what its reward certifies. The end-to-end success of Equation (1) decomposes into a selection term and an ordering term, which mirror the assembler and workflower factors of Equation (2) and thus suggest deriving the reward of each role from the term it governs. For a selection S and an order σ produced for intent x,

$$r_\mathrm{A}(S) = \mathrm{Sel}(S) + \gamma\, \mathbf{1}[\mathrm{Sel}(S) = 1], \quad r_\mathrm{W}(\sigma \mid S) = \mathbf{1}[\sigma \in \mathrm{Perm}(S)]\Big(\omega + (1-\omega)\,\frac{\mathrm{sat}(\sigma)}{\mathrm{app}(S)}\Big), \tag{4}$$

Table 1: Main results on NETBENCH and TaskBench. NETBENCH is reported separately for short intents with $|S^\star| \leq 4$ and long intents with $|S^\star| \geq 5$. All open-weight systems share the same backbone. Best results are in **bold** and second best are underlined.

Method	NETBENCH ($\|S^\star\| \leq 4$)					NETBENCH ($\|S^\star\| \geq 5$)					TaskBench			
	n-F1	e-F1	Sel	Succ	Judge	n-F1	e-F1	Sel	Succ	Judge	n-F1	e-F1	Succ	Judge
GPT-5.5	0.699	0.334	0.807	0.518	0.746	0.681	0.198	0.796	0.312	0.576	0.823	0.493	0.486	0.706
Claude Opus 4.6	0.682	0.320	0.800	0.502	0.728	0.664	0.189	0.788	0.268	0.552	0.818	0.491	0.470	0.692
Single agent	0.629	0.223	0.763	0.360	0.652	0.602	0.196	0.739	0.182	0.412	0.730	0.426	0.212	0.440
ReAct	0.632	0.231	0.772	0.382	0.668	0.607	0.203	0.751	0.200	0.426	0.742	0.432	0.226	0.456
Multi-agent pipeline	0.634	0.236	0.773	0.394	0.676	0.611	0.207	0.748	0.214	0.440	0.744	0.435	0.234	0.464
Single-agent SFT	0.653	0.368	0.683	0.430	0.538	0.637	0.339	0.662	0.334	0.392	0.854	0.640	0.640	0.814
Per-role SFT	0.665	0.376	0.692	0.454	0.558	0.648	0.347	0.671	0.366	0.414	0.865	0.654	0.662	0.826
Single-agent GRPO	0.709	0.349	0.843	0.622	0.816	0.646	0.284	0.781	0.388	0.602	0.834	0.575	0.584	0.748
MARFT	0.704	0.367	0.810	0.612	0.792	0.659	0.318	0.774	0.504	0.618	0.830	0.604	0.614	0.786
MANET	**0.724**	**0.392**	**0.855**	**0.696**	**0.854**	**0.697**	**0.353**	**0.826**	**0.650**	**0.752**	**0.866**	**0.655**	**0.676**	**0.834**

where γ is a bonus granted when $\mathrm{Sel}(S) = 1$, $\mathrm{Perm}(S)$ denotes the set of permutations of S, ω credits a valid permutation, and $\mathrm{app}(S)$ and $\mathrm{sat}(\sigma)$ count the precedence pairs applicable within S and satisfied by σ, respectively, with the ratio taken as one when no pair applies. The ordering reward thus relaxes the indicator $\mathbf{1}[\sigma \models \prec]$ of Equation (1) into the fraction of satisfied pairs, which provides partial credit throughout training while reserving the maximal value for an order that execution would accept. Since the plan corresponds to no term of Succ, its reward combines a term $r_{\mathrm{fmt}}(\tau)$ for well-formedness and granularity with an optional probe of the selection that it induces downstream,

$$r_{\mathrm{P}}(\tau) \;=\; (1-\lambda)\, r_{\mathrm{fmt}}(\tau) + \lambda\, \mathrm{Sel}\big(\hat{S}(\tau)\big), \tag{5}$$

where $\hat{S}(\tau)$ is the selection that the assembler derives from τ. In contrast to rewards anchored to a single reference path, which certify the imitation of that path, these rewards certify the answer key itself and thereby credit admissible alternatives on equal terms.

The three roles are optimized jointly with group-relative policy optimization (Shao et al., 2024; Guo et al., 2025; Zhao et al., 2026; Feng et al., 2026) over a rollout that branches by role, in which each role samples a group of G candidates conditioned on the same input, such that the candidates within a group differ only in that role's decision. The group-standardized advantage $\hat{A}_i = (r_i - \bar{r})/\mathrm{std}(r)$, with the mean and standard deviation taken over the group, therefore isolates the contribution of each role without a learned critic, and every role $k \in \mathcal{R}$ maximizes the clipped surrogate

$$\mathcal{J}_k(\theta_k) \;=\; \mathbb{E}_x\Big[\frac{1}{G}\sum_{i=1}^{G} \min\big(\rho_i \hat{A}_i,\; \mathrm{clip}(\rho_i, 1-\epsilon, 1+\epsilon)\hat{A}_i\big)\Big] \;-\; \beta\, \mathrm{KL}\big(\pi_{\theta_k} \,\|\, \pi_k^{\mathrm{ref}}\big), \tag{6}$$

in which r_i is the reward of candidate i for role k, ρ_i is its importance ratio, ϵ bounds the trust region, and β weights a KL penalty towards the initial policy π_k^{ref} of the role. All roles are updated within one step, so the pipeline trains as a single policy under role-specific gradients.

Since each downstream role consumes the output of its predecessor, its training inputs can be drawn either from reference artifacts or from the outputs of the preceding role. The former offer a clean interface yet differ from the imperfect inputs received at deployment, whereas the latter match deployment conditions yet carry transient errors into training, a tension that parallels the exposure bias addressed by scheduled sampling (Bengio et al., 2015). MANET therefore trains the workflower on a mixture of both, presenting the reference set $S^\star$ with probability p and the assembler's own selection $\hat{S}$ otherwise, so that $p = 1$ recovers teacher forcing and $p = 0$ fully on-policy interaction, while the same mixture applies analogously to the interface between the planner and the assembler.

5 EXPERIMENTS

In this section, we conduct comprehensive experiments to validate the efficacy of MANET in network task automation. We first compare MANET with proprietary, prompting-based, and training-based alternatives on NETBENCH and TaskBench (Section 5.2), then dissect the contributions of its multi-path rewards, role decomposition, and training interface through ablation studies (Section 5.3), and finally examine its sensitivity to key hyperparameters and to workflow length (Section 5.4).

Table 2: Ablation study of MANET. Each variant removes or replaces a single component of the full framework, grouped by the design dimension that it probes.

Variant	NETBENCH						TaskBench		
	$\lvert S^\star\rvert \le 4$			$\lvert S^\star\rvert \ge 5$					
	Sel	Succ	Judge	Sel	Succ	Judge	n-F1	Succ	Judge
MANET (full)	0.855	0.696	0.854	0.826	0.650	0.752	0.866	0.676	0.834
Reward design									
single-reference reward	0.794	0.558	0.702	0.788	0.540	0.604	0.861	0.670	0.826
w/o partial ordering credit	0.850	0.650	0.848	0.823	0.588	0.738	0.862	0.646	0.828
w/o selection bonus ($\gamma = 0$)	0.845	0.676	0.838	0.809	0.632	0.746	0.858	0.664	0.832
Role decomposition									
w/o planner	0.854	0.672	0.836	0.815	0.620	0.702	0.853	0.642	0.806
w/o role decomposition	0.843	0.622	0.816	0.781	0.388	0.602	0.834	0.584	0.748
Parameter sharing									
w/o per-role LoRA	0.849	0.668	0.830	0.803	0.566	0.706	0.857	0.660	0.822
full-parameter training per role	0.858	0.702	0.856	0.831	0.658	0.758	0.868	0.680	0.838
Training interface									
on-policy inputs only ($p = 0$)	0.801	0.628	0.770	0.798	0.598	0.692	0.859	0.640	0.812
reference inputs only ($p = 1$)	0.851	0.662	0.844	0.819	0.628	0.736	0.850	0.624	0.790

5.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate MANET on the proposed NETBENCH and the widely adopted TaskBench (Shen et al., 2024), which assesses the generalization beyond the network domain. On NETBENCH, we report the selection score Sel, the end-to-end success Succ, and the LLM-based judge of Section 3.3 separately for short intents with $|S^\star| \le 4$ and long intents with $|S^\star| \ge 5$, complemented by node-level and edge-level F1 scores (n-F1 and e-F1) against the reference chain. On TaskBench, we pool its three domains, hold out 500 stratified requests, and report n-F1, e-F1, the judge, and a strict Succ that requires an exact match of both the tools and their dependencies.

Baselines. We compare MANET with representative baselines, which fall into three families according to how they handle capabilities unseen in pretraining. The first family relies on state-of-the-art proprietary LLMs, namely GPT-5.5 (OpenAI, 2026) and Claude Opus 4.6 (Anthropic, 2026), which map the catalog and the intent to a workflow in a zero-shot manner. The second family imposes inference-time structure on the same backbone, including a single agent that sequentially plans, assembles, and orders the workflow, its ReAct (Yao et al., 2023a) variant that further reasons before each action, and an untrained multi-agent pipeline that instantiates the three roles of MANET following MetaGPT (Hong et al., 2024). The third family adapts the backbone through training, including supervised fine-tuning of the single agent and of each role (Zhang et al., 2023b), the single agent optimized by GRPO (Shao et al., 2024) under the same multi-path rewards as MANET, and MARFT (Liao et al., 2025), which trains role-structured teams with multi-agent reinforcement learning.

Implementation Details. All open-weight systems are instantiated on Qwen3-4B-Instruct-2507 (Yang et al., 2025) and evaluated with greedy decoding. By default, MANET builds the three roles upon a single shared backbone, each with a role-specific LoRA adapter (Hu et al., 2022), while an alternative that independently optimizes three models with full parameters is explored in Section 5.3. Joint optimization runs for 280 steps with a cosine-decayed learning rate of 5×10^{-5}, where each role samples $G = 8$ candidates per group under a KL penalty towards its initial policy. Unless otherwise specified, we set $\gamma = 0.25$, $\omega = 0.3$, $\lambda = 0.7$, and $p = 0.5$, and mix the verifiable reward with the judge reward under weight $\eta = 0.5$. Appendix B provides further implementation details.

5.2 MAIN RESULTS

Superiority on NETBENCH. As shown in Table 1, MANET attains the highest Sel, Succ, and judge score on both splits of NETBENCH, ahead of all open-weight alternatives and the state-of-the-art proprietary LLMs. MANET also degrades the least from short to long intents, so that its margin over the strongest baseline roughly doubles on long intents. Prompting-based methods gain only marginally from staged reasoning or instantiated roles, which indicates that inference-time structure remains insufficient for a proprietary catalog beyond the coverage of pretraining. Supervised fine-tuning attains an e-F1 close to that of MANET by reproducing the reference chain, yet falls markedly

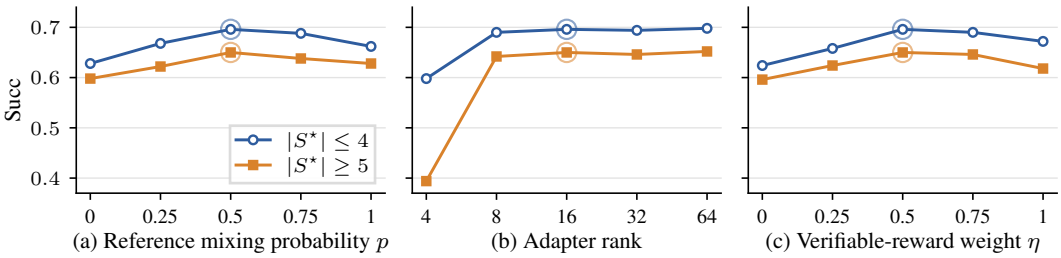

Figure 3: Sensitivity of MANET on NETBENCH to the reference mixing probability p, the adapter rank, and the verifiable-reward weight η, where circled points mark the adopted configuration.

behind in Sel and Succ, since the imitation of a single path suppresses the equally valid alternatives in the answer key. Under identical multi-path rewards, the single agent remains competitive on short intents but deteriorates sharply on long ones, a controlled contrast that attributes the remaining advantage of MANET to the decomposition of heterogeneous decisions across specialized roles.

Dependencies as the principal bottleneck of NTA. Although the proprietary LLMs identify capabilities nearly as accurately as MANET on long intents, their Succ remains below half that of MANET. Figure 4 attributes this discrepancy primarily to order violations, which account for most failures of GPT-5.5, whereas missing core capabilities dominate the errors of per-role supervised fine-tuning and the untrained multi-agent pipeline introduces the most invalid capabilities. MANET reduces all three types of error and most markedly suppresses the order violations that prevail among the untrained systems. This observation indicates that once capabilities are selected accurately, the principal difficulty of NTA lies in their dependencies.

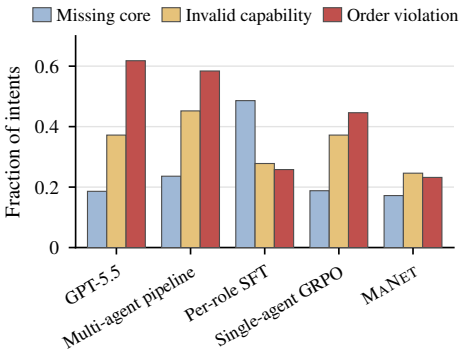

Figure 4: Fractions of long intents with each error type, where these types may co-occur.

Generalization to TaskBench. On TaskBench, whose requests admit essentially a single reference and involve short dependency chains, MANET remains the strongest on all metrics, although its margin over per-role supervised fine-tuning narrows markedly and, unlike on NETBENCH, both supervised variants surpass the reinforcement learning baselines. This reversal suggests that imitation already captures most of the task when the reference is unique, whereas role decomposition and multi-path rewards exhibit their full advantage on long intents that admit several valid workflows.

5.3 ABLATION STUDY

Reward design. As reported in Table 2, substituting a single-reference reward for the multi-path one incurs the largest degradation among the reward variants on both splits of NETBENCH, yet barely affects TaskBench, whose requests admit essentially one reference. This contrast establishes the multi-path verifiable reward as the primary source of the advantage on intents with several valid workflows. Without the partial ordering credit, the degradation is sharper on long intents than on short ones, since the ordering signal degenerates into a sparse binary outcome as applicable precedence pairs multiply. The selection bonus yields a smaller yet consistent gain on both benchmarks, as it merely sharpens a selection signal that core recall and valid precision already convey.

Role decomposition and parameter sharing. The variant without role decomposition suffers the largest degradation on long intents among all variants, and the variant without the planner declines consistently, which confirms that separating heterogeneous decisions is most beneficial where errors compound along the workflow. For parameter sharing, a single adapter shared by all roles degrades long intents far more than short ones, which indicates that heterogeneous decisions compete for shared capacity. At the other extreme, full-parameter training of separate models yields only a marginal gain over role-specific adapters on a shared backbone. The default configuration thus retains nearly all the benefit of independent roles at a fraction of their parameter and memory cost.

Training interface. Both extremes of the mixing probability underperform the intermediate mixture, since fully on-policy inputs at $p = 0$ propagate upstream errors into training and thus depress

the selection score. Reference inputs alone at $p = 1$, in contrast, fail to prepare the downstream roles for the imperfect inputs encountered at deployment and thus incur the larger loss on TaskBench.

5.4 Analysis

Sensitivity. Figure 3 shows that MANET remains robust across a broad range of settings. Its Succ peaks at an intermediate mixing probability p, which corroborates the training-interface ablation, and saturates once the adapter rank reaches eight, below which long intents deteriorate sharply. The best results arise with a verifiable-reward weight η between 0.5 and 0.75, which suggests that the judge's semantic feedback complements verification against the answer key without replacing it.

Effect of workflow length. Figure 5 shows that the success of every method declines as the number of required capabilities increases, yet MANET degrades far more gently, so that its advantage widens steadily as workflows lengthen. On intents with one or two capabilities, MANET performs on par with the single agent trained under identical rewards, whereas on intents with seven capabilities its Succ more than doubles that of this agent. This trend indicates that the decomposition into specialized roles grows more valuable as errors compound along the longer chains of interdependent decisions that such intents entail.

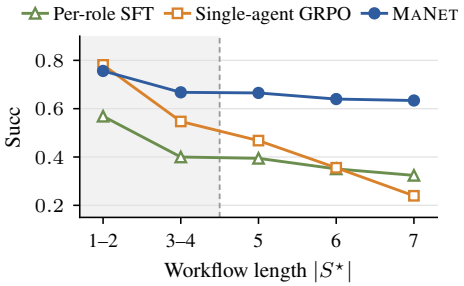

Figure 5: Succ of each method on NETBENCH under different workflow lengths.

Case study. Figure 6 contrasts MANET with two baselines on long intents that exhibit an incomplete selection and an invalid order. On the first intent, per-role supervised fine-tuning produces a plausible chain that omits the low-altitude perception required by the drone and thus fails to fulfill the intent. On the second intent, GPT-5.5 selects every required capability but places crowd analysis before the low-altitude perception on which it depends, which renders the workflow non-executable despite a perfect selection score. MANET resolves both intents with complete selections in valid orders.

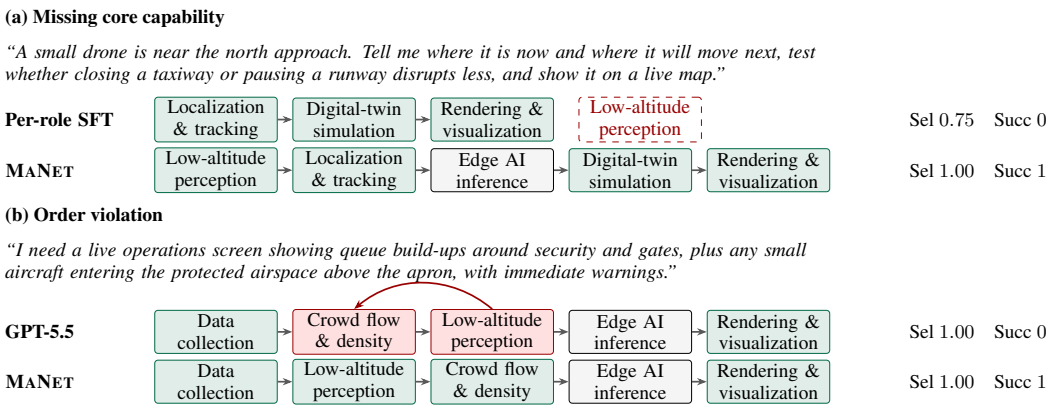

Figure 6: Two abridged long intents from NETBENCH that exhibit the selection and ordering errors of Figure 4 under the baselines most prone to them. Core capabilities are green and errors red.

6 Conclusion

This paper explores LLM agents for network task automation (NTA), which compiles operator intents into executable workflows and empowers the next-generation core network. On the one hand, we construct NETBENCH, the first NTA benchmark, which features multi-path answer keys and an order-aware metric. On the other hand, we propose MANET, a multi-agent framework that entrusts the heterogeneous decisions of NTA to specialized roles and jointly optimizes them with verifiable rewards. Extensive experiments showcase that MANET consistently outperforms proprietary LLMs and open-weight alternatives, which paves the way towards closed-loop agentic automation of real-world networks. Appendix H discusses the limitations of this work and directions for future research.

AI USE STATEMENT

In this work, generative AI tools were used to aid and polish the writing of the manuscript, such as improving its grammar, wording, and overall readability, and all AI-assisted content has been carefully reviewed and revised by the authors to ensure that it is accurate and faithfully reflects our work. Beyond writing assistance, generative AI is also involved in the evaluation protocol of NETBENCH, wherein an LLM-based judge provides an independent semantic assessment of the generated workflows. The authors take responsibility for the content produced with the aid of generative AI tools.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we release the anonymized source code at https://anonymous.4open.science/r/MaNet-DA58/, including the data splits of NETBENCH together with the scripts for training and evaluation. The task formulation, the multi-path answer keys, and the evaluation protocol are specified in Section 3, while the factorized policy of MANET, its role-level rewards, and the joint optimization are formalized in Section 4. The backbone, the role-specific adapters, the optimization hyperparameters, and the baseline configurations are further detailed in Section 5.1 and Appendix B, and the complete prompts of all three roles are provided in Appendix G.

REFERENCES

Anthropic. System card: Claude Opus 4.6. https://www.anthropic.com/system-cards, 2026.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2015.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chen Qian, Chi-Min Chan, Yujia Qin, Yaxi Lu, Ruobing Xie, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. *arXiv preprint arXiv:2308.10848*, 2023a.

Xudong Chen, Yixin Liu, Hua Wei, and Kaize Ding. Lemon: Learning executable multi-agent orchestration via counterfactual reinforcement learning. *arXiv preprint arXiv:2605.14483*, 2026.

Yongchao Chen, Jacob Arkin, Yang Zhang, Nicholas Roy, and Chuchu Fan. Scalable multi-robot collaboration with large language models: Centralized or decentralized systems? *arXiv preprint arXiv:2309.15943*, 2023b.

Lam Dinh, Sihem Cherrared, Xiaofeng Huang, and Fabrice Guillemin. Towards end-to-end network intent management with large language models. *arXiv preprint arXiv:2504.13589*, 2025.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*, 2024.

Zhuoyun Du, Runze Wang, Huiyu Bai, Zouying Cao, Xiaoyong Zhu, Yu Cheng, Bo Zheng, Wei Chen, and Haochao Ying. Enabling agents to communicate entirely in latent space. *arXiv preprint arXiv:2511.09149*, 2025.

Lang Feng, Longtao Zheng, Shuo He, Fuxiang Zhang, and Bo An. Dr. mas: Stable reinforcement learning for multi-agent llm systems. *arXiv preprint arXiv:2602.08847*, 2026.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*, 2024.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024.

Md Kamrul Hossain and Walid Aljoby. Ai-driven intent-based networking approach for self-configuration of next generation networks. *arXiv preprint arXiv:2603.23772*, 2026.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Yilun Kong, Jingqing Ruan, Yihong Chen, Bin Zhang, Tianpeng Bao, Shiwei Shi, Guoqing Du, Xiaoru Hu, Hangyu Mao, Ziyue Li, et al. Tptu-v2: Boosting task planning and tool usage of large language model-based agents in real-world systems. *arXiv preprint arXiv:2311.11315*, 2023.

Rui Li, Zeyu Zhang, Xiaohe Bo, Quanyu Dai, Chaozhuo Li, Feng Wen, and Xu Chen. Towards adaptive, scalable, and robust coordination of llm agents: A dynamic ad-hoc networking perspective. *arXiv preprint arXiv:2602.08009*, 2026a.

Shiyuan Li, Yixin Liu, Qingsong Wen, Chengqi Zhang, and Shirui Pan. Assemble your crew: Automatic multi-agent communication topology design via autoregressive graph generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 23142–23150, 2026b.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 17889–17904, 2024.

Junwei Liao, Muning Wen, Jun Wang, and Weinan Zhang. Marft: Multi-agent reinforcement fine-tuning. *arXiv preprint arXiv:2504.16129*, 2025.

Zhao Mandi, Shreeya Jain, and Shuran Song. Roco: Dialectic multi-robot collaboration with large language models. *arXiv preprint arXiv:2307.04738*, 2023.

Dimitrios Michael Manias, Ali Chouman, and Abdallah Shami. Towards intent-based network management: Large language models for intent extraction in 5g core networks. In *2024 20th International Conference on the Design of Reliable Communication Networks (DRCN)*, pp. 1–6. IEEE, 2024.

OpenAI. GPT-5.5 system card. https://deploymentsafety.openai.com/gpt-5-5, 2026.

Chau Pham, Boyi Liu, Yingxiang Yang, Zhengyu Chen, Tianyi Liu, Jianbo Yuan, Bryan A Plummer, Zhaoran Wang, and Hongxia Yang. Let models speak ciphers: Multiagent debate through embeddings. *arXiv preprint arXiv:2310.06272*, 2023.

Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 2023.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023.

Jingqing Ruan, Yihong Chen, Bin Zhang, Zhiwei Xu, Tianpeng Bao, Hangyu Mao, Ziyue Li, Xingyu Zeng, Rui Zhao, et al. Tptu: Task planning and tool usage of large language model-based ai agents. In *NeurIPS 2023 foundation models for decision making workshop*, 2023.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180, 2023.

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. *Advances in Neural Information Processing Systems*, 37:4540–4574, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron T. Parisi, Abhishek Kumar, Alexander A. Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Fathy Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura Culp, Lechao Xiao, Maxwell L. Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yundi Qian, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research*, 2024.

Wen Tong, Wei Huo, Thierry Lejkin, Joel Penhoat, Chenghui Peng, Carlos Pereira, Fei Wang, Shaoyun Wu, Lu Yang, and Yuanming Shi. A-core: A novel framework of agentic ai in the 6g core network. In *2025 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1104–1109. IEEE, 2025.

Zefan Wang, Zichuan Liu, Yingying Zhang, Aoxiao Zhong, Lunting Fan, Lingfei Wu, and Qingsong Wen. Rcagent: Cloud root cause analysis by autonomous agents with tool-augmented large language models. *arXiv preprint arXiv:2310.16340*, 2023.

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

Bingyu Yan, Zhibo Zhou, Litian Zhang, Lian Zhang, Ziyi Zhou, Dezhuang Miao, Zhoujun Li, Chaozhuo Li, and Xiaoming Zhang. Beyond self-talk: A communication-centric survey of llm-based multi-agent systems. *arXiv preprint arXiv:2502.14321*, 2025.

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

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*, 2023a.

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, et al. Retroformer: Retrospective large language agents with policy gradient optimization. *arXiv preprint arXiv:2308.02151*, 2023b.

Tao Yu, Hao Wang, Changyu Li, Shenghua Chai, Minghui Zhang, Zhongtian Luo, Yuxuan Zhou, Haopeng Jin, Zhaolu Kang, Jiabing Yang, et al. Beyond the all-in-one agent: Benchmarking role-specialized multi-agent collaboration in enterprise workflows. *arXiv preprint arXiv:2605.08761*, 2026.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022.

Zhichen Zeng, Huiyuan Chen, Ju Ming Cheng, Ying Chen, Xiyuan Yang, Chaosheng Dong, Haiyang Zhang, and Hanghang Tong. One model, many minds: Unlocking multi-agent synergy in a single agent via mixture of roles. *arXiv preprint arXiv:2608.27338*, 2026.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for LLM-based multi-agent systems. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topologies via graph neural networks. In *Forty-second International Conference on Machine Learning*, 2025b.

Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models. *arXiv preprint arXiv:2307.02485*, 2023a.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*, 2023b.

Yujie Zhao, Lanxiang Hu, Yang Wang, Minmin Hou, Hao Zhang, Ke Ding, and Jishen Zhao. Stronger-mas: Multi-agent reinforcement learning for collaborative llms. In *International Conference on Learning Representations*, volume 2026, pp. 150619–150651, 2026.

Yujia Zheng, Zhuokai Zhao, Zijian Li, Yaqi Xie, Mingze Gao, Lizhu Zhang, and Kun Zhang. Thought communication in multiagent collaboration. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *Forty-first International Conference on Machine Learning*, 2024.

Jiaru Zou, Xiyuan Yang, Ruizhong Qiu, Gaotang Li, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, et al. Latent collaboration in multi-agent systems. *arXiv preprint arXiv:2511.20639*, 2025.

A GLOSSARY OF SYMBOLS

Table 3 lists the principal symbols of this paper, grouped by the component to which they belong.

Table 3: Glossary of variables and symbols used in this paper.

Symbol	Type	Description
Task formulation		
x	Text	Operator intent expressed in natural language
$\mathcal{C}$	Set	Closed catalog of network capabilities
$\prec$	Relation	Precedence relation over $\mathcal{C}$
$w = (S, \sigma)$	Tuple	Workflow with capability set $S \subseteq \mathcal{C}$ and execution order σ
$w^\star = (S^\star, \sigma^\star)$	Tuple	Reference workflow of an intent
$\tau^\star$	Sequence	Reference plan of an intent
Evaluation		
$\mathcal{K}_{\mathrm{c}}(x)$	Set	Core capabilities required by every valid workflow
$\mathcal{K}_{\mathrm{o}}(x)$	Set	Optional capabilities admissible as additions
$\mathrm{Sel}(S)$	$[0, 1]$	Selection score, the product of core recall and valid precision
$\mathrm{Succ}(S, \sigma)$	$\{0, 1\}$	End-to-end success of a workflow
Framework		
$\mathcal{R}$	Set	Roles $\{\mathrm{P}, \mathrm{A}, \mathrm{W}\}$, namely planner, assembler, and workflower
τ	Sequence	Plan of natural-language steps produced by the planner
π_{θ_k}	Function	Policy of role $k \in \mathcal{R}$ with parameters θ_k
$r_{\mathrm{P}}, r_{\mathrm{A}}, r_{\mathrm{W}}$	$\mathbb{R}$	Rewards of the planner, the assembler, and the workflower
r_k^{J}	$\mathbb{R}$	Reward of the LLM-based judge for role k
Hyperparameters		
γ	$\mathbb{R}_+$	Bonus granted to the assembler when $\mathrm{Sel}(S) = 1$
ω	$[0, 1]$	Credit granted to the workflower for a valid permutation
λ	$[0, 1]$	Weight of the downstream selection term in the planner reward
G	$\mathbb{N}$	Number of candidates sampled per group for each role
p	$[0, 1]$	Probability of presenting reference inputs to the downstream roles
η	$[0, 1]$	Weight of the verifiable reward against the judge reward

B IMPLEMENTATION DETAILS

Backbone and adapters. All open-weight systems, including every trained baseline, are instantiated on Qwen3-4B-Instruct-2507 and decoded greedily at evaluation. By default, MANET keeps this backbone frozen and equips each of the three roles with a LoRA adapter of rank 16, so that the roles share the backbone while their trainable parameters remain disjoint. Two alternative allocations of capacity complete the spectrum examined in Section 5.3, the first of which shares a single adapter across all three roles, whereas the second optimizes three separate models with full parameters.

Optimization. The three roles are optimized jointly with group-relative policy optimization for 280 steps under a cosine-decayed learning rate of 5×10^{-5}, wherein each role samples $G = 8$ candidates per group and a KL penalty towards the initial policy of the role regularizes every update. During training, the inputs of the assembler and the workflower are drawn from the reference artifacts with probability $p = 0.5$ and from the outputs of the preceding role otherwise, whereas at evaluation every downstream role consumes the greedy output of its upstream role.

Rewards. On NETBENCH, the verifiable rewards follow Equations (4) and (5) under the settings $\gamma = 0.25$, $\omega = 0.3$, and $\lambda = 0.7$, where the term $r_{\mathrm{fmt}}(\tau)$ credits a plan that forms a non-empty list of

steps at an appropriate granularity. The verifiable reward of each role is then mixed with the reward of the LLM-based judge under weight $\eta = 0.5$. On TaskBench, whose requests admit essentially a single reference each, the same rewards are computed against this reference as a degenerate answer key, and all remaining settings are kept identical to those on NETBENCH.

Baselines. GPT-5.5 and Claude Opus 4.6 receive the catalog and the intent and emit a workflow in a single zero-shot pass. The single agent executes planning, assembly, and ordering in sequence with one instance of the backbone, and its ReAct variant additionally reasons before each action. The multi-agent pipeline instantiates the three roles of MANET with the prompts of Appendix G but without any training. Among the trained baselines, supervised fine-tuning adapts either the single agent or each individual role to the reference workflows. Single-agent GRPO optimizes the single agent under the multi-path rewards aggregated over its three stages, whereas MARFT trains the three roles with the multi-agent reinforcement fine-tuning procedure of Liao et al. (2025).

Ablation variants. Each variant in Table 2 departs from the default configuration in a single respect. The single-reference reward scores the selection and the order against the reference chain alone, and the variant without partial ordering credit replaces the fraction of satisfied precedence pairs with the indicator $\mathbf{1}[\sigma \models \prec]$. The variant without the selection bonus sets $\gamma = 0$, whereas the variant without the planner conditions the assembler on the intent alone. The variant without role decomposition coincides with single-agent GRPO, the two variants of parameter sharing correspond to the allocations described above, and the variants of the training interface fix p at zero or one.

C STATISTICAL SIGNIFICANCE

Table 4 reports paired-bootstrap 95% confidence intervals of the differences in Succ, obtained from 1,000 resamples of the 500 intents in each test split, for the key contrasts of Section 5. On long intents, the advantages of MANET over GPT-5.5, over the single agent trained under identical rewards, and over MARFT exclude zero by a wide margin, as do the gains of the multi-path reward over the single-reference reward and of role-specific adapters over a shared one. By contrast, the intervals of full-parameter training against role-specific adapters contain zero on both splits, which confirms that the default configuration forgoes no significant benefit of independent roles.

Table 4: Paired-bootstrap 95% confidence intervals of the differences in Succ for key contrasts.

Benchmark	**Contrast**	ΔSucc	**95% CI**
NETBENCH ($\lvert S^\star\rvert \geq 5$)	MANET − GPT-5.5	$+0.338$	$[+0.280, +0.396]$
	MANET − single-agent GRPO	$+0.262$	$[+0.210, +0.314]$
	MANET − MARFT	$+0.146$	$[+0.098, +0.194]$
	multi-path − single-reference reward	$+0.110$	$[+0.068, +0.150]$
	role-specific − shared adapter	$+0.084$	$[+0.050, +0.120]$
	full-parameter − role-specific adapters	$+0.008$	$[-0.014, +0.030]$
NETBENCH ($\lvert S^\star\rvert \leq 4$)	MANET − single-agent GRPO	$+0.074$	$[+0.034, +0.116]$
	full-parameter − role-specific adapters	$+0.006$	$[-0.012, +0.024]$
TaskBench	MANET − single-agent GRPO	$+0.092$	$[+0.054, +0.132]$

D NETBENCH DETAILS

D.1 CAPABILITY CATALOG AND PRECEDENCE RELATION

The catalog comprises twelve sensing and computing capabilities drawn from a proprietary service catalog in production use, which Table 5 groups by their function. The precedence relation $\prec$ follows the flow of data through these groups. Data collection precedes every other capability, and spatial rendering and visualization succeeds every capability whose output it presents. Between these two ends, the perception capabilities precede the analysis capabilities that consume their outputs, and both precede edge AI inference and digital-twin simulation, whereas model training and adaptation

precedes edge AI inference, which in turn precedes digital-twin simulation. A workflow is therefore executable only when its order satisfies every pair of $\prec$ that applies within its selected set.

Table 5: Capability catalog of NETBENCH, grouped by function.

Group	Capabilities
Acquisition	data collection
Perception	target detection, target localization and tracking, low-altitude perception, ambient physical awareness, spatial environment digitization
Analysis	behavior and action analysis, crowd flow and density analysis
Computation	edge AI inference, model training and adaptation, digital-twin simulation
Presentation	spatial rendering and visualization

D.2 DATASET STATISTICS

Table 6 summarizes the principal statistics of NETBENCH, and Figure 7 further depicts the distribution of workflow lengths on the test split, which is evenly divided between short and long intents as described in Section 3.2, so that Section 5 can evaluate the two halves separately.

Table 6: Principal statistics of NETBENCH.

Statistic	Value
Capabilities in the catalog	12
Workflow length $\lvert S^\star \rvert$	1–7
Training intents	6,701
Validation intents	200
Test intents	1,000
Total intents	7,901

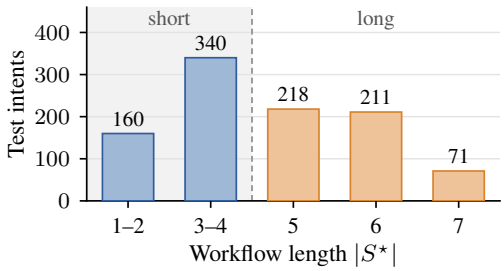

Figure 7: Workflow lengths of the test intents, where the shaded region marks short intents.

D.3 ANSWER KEY ANNOTATION

Given the intent, the catalog, and one known valid workflow, each annotator labels every capability of the catalog as core, optional, or invalid. The labels are aggregated by majority vote under a rule that preserves the validity of the reference chain, whereby a capability of the reference chain is core when a majority labels it as such and optional otherwise, whereas a capability outside the reference chain is optional only when every annotator accepts it. The need for such a key is confirmed by an adjudication of the intents on which a frontier model disagrees with the reference labels, which finds both answers acceptable in 43% of the adjudicated cases.

E DIVERSE AGENT BACKBONES

To examine whether the advantage of MANET depends on a particular backbone, we instantiate MANET together with five representative baselines on Qwen3-8B, Qwen3-14B, and Llama-3.1-8B-Instruct, while retaining every remaining setting of Appendix B. Figure 8 reports the selection score, the end-to-end success, and the judge score of each system on both splits of NETBENCH.

MANET attains the highest value on every metric and split under all three backbones, as in the main experiments, and its margin over the single agent trained under identical rewards is consistently larger on long intents than on short ones. Scaling the backbone from Qwen3-8B to Qwen3-14B improves every method, yet benefits the prompting-based systems most, as a stronger pretrained model partially compensates for the absence of training. The weaker Llama-3.1-8B-Instruct lowers every method relative to Qwen3-8B without altering their ranking on any metric, which indicates that the benefit of MANET stems from the framework rather than from a particular model family.

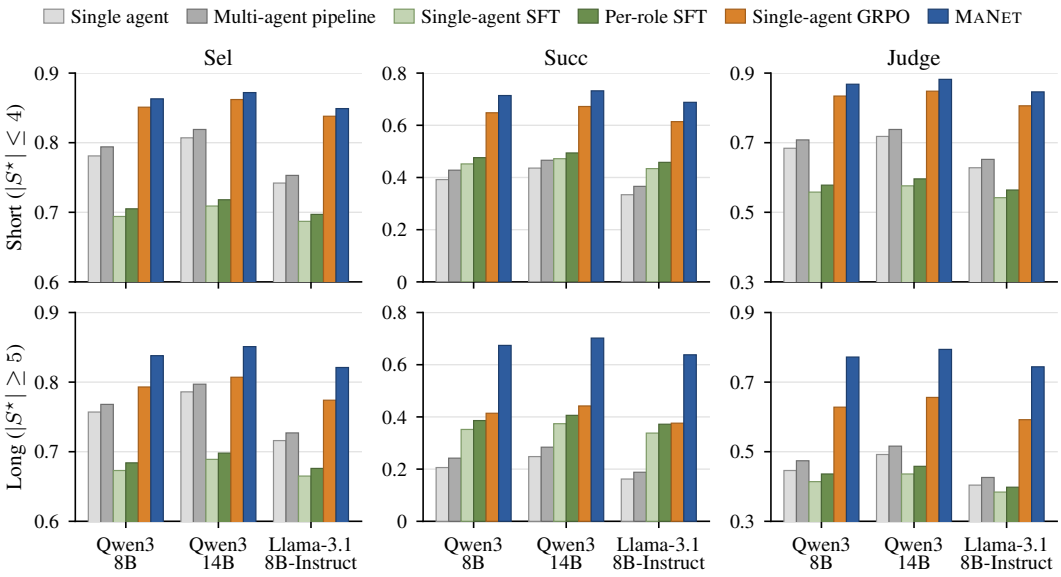

Figure 8: Sel, Succ, and judge score on NETBENCH under three backbones, with short intents in the upper row and long intents in the lower row.

F ALGORITHM

Algorithm 1 summarizes the joint training of MANET, in which each role samples a group of candidates under the training interface of Section 4.3, receives the role-level rewards of Equations (4) and (5) mixed with the reward of the LLM-based judge, and is updated through the clipped surrogate of Equation (6). Algorithm 2 depicts the greedy inference used at evaluation and deployment, wherein every downstream role consumes the output of its upstream role.

Algorithm 1 Joint multi-agent reinforcement learning of MANET

Input: intents $\mathcal{D}$ with answer keys $(\mathcal{K}_c, \mathcal{K}_o)$ and reference artifacts $(\tau^\star, S^\star)$, catalog $\mathcal{C}$ with precedence relation $\prec$, initial policies $\{\pi_{\theta_k}\}_{k\in\mathcal{R}}$, group size G, mixing probability p, reward hyperparameters $\gamma, \omega, \lambda, \eta$, and number of steps T

Output: trained role policies $\{\pi_{\theta_k}\}_{k\in\mathcal{R}}$

for $t = 1, \dots, T$ **do**
  sample a batch $\mathcal{B} \subset \mathcal{D}$ of intents
  for all $x \in \mathcal{B}$ **do**
    $\{\tau_i\}_{i=1}^{G} \sim \pi_{\theta_P}(\cdot \mid x)$ ▷ planner group
    $r_{P,i} \leftarrow (1-\lambda)\, r_{\text{fmt}}(\tau_i) + \lambda\, \text{Sel}\big(\hat{S}(\tau_i)\big)$ ▷ Eq. (5)
    $\bar{\tau} \leftarrow \tau^\star$ with probability p, and the greedy plan of π_{θ_P} otherwise ▷ training interface
    $\{S_i\}_{i=1}^{G} \sim \pi_{\theta_A}(\cdot \mid x, \bar{\tau})$ ▷ assembler group
    $r_{A,i} \leftarrow \text{Sel}(S_i) + \gamma\, \mathbf{1}[\text{Sel}(S_i) = 1]$ ▷ Eq. (4)
    $\bar{S} \leftarrow S^\star$ with probability p, and the greedy selection $\hat{S}$ of π_{θ_A} otherwise ▷ training interface
    $\{\sigma_i\}_{i=1}^{G} \sim \pi_{\theta_W}(\cdot \mid x, \bar{S})$ ▷ workflower group
    $r_{W,i} \leftarrow r_W(\sigma_i \mid \bar{S})$ ▷ Eq. (4)
    for all $k \in \mathcal{R}$ and $i \in \{1, \dots, G\}$ **do**
      $r_{k,i} \leftarrow \eta\, r_{k,i} + (1-\eta)\, r^{J}_{k,i}$ ▷ judge mixing
      $\hat{A}_{k,i} \leftarrow \big(r_{k,i} - \text{mean}_j\, r_{k,j}\big)/\text{std}_j\, r_{k,j}$ ▷ group standardization
    end for
  end for
  for all $k \in \mathcal{R}$ **do**
    update θ_k by ascending $\mathcal{J}_k(\theta_k)$ in Eq. (6) with advantages $\hat{A}_{k,i}$ ▷ joint update
  end for
end for

Algorithm 2 Greedy inference of MANET

Input: operator intent x, catalog $\mathcal{C}$, and trained role policies $\{\pi_{\theta_k}\}_{k\in\mathcal{R}}$
Output: workflow $w = (\hat{S}, \hat{\sigma})$
1: $\hat{\tau} \leftarrow \text{greedy}\left(\pi_{\theta_{\mathrm{P}}}(\cdot \mid x)\right)$ ▷ plan the steps
2: $\hat{S} \leftarrow \text{greedy}\left(\pi_{\theta_{\mathrm{A}}}(\cdot \mid x, \hat{\tau})\right)$ ▷ select the capabilities
3: $\hat{\sigma} \leftarrow \text{greedy}\left(\pi_{\theta_{\mathrm{W}}}(\cdot \mid x, \hat{S})\right)$ ▷ order the selection
4: **return** $w = (\hat{S}, \hat{\sigma})$

G PROMPTS

The three role prompts below are shared by every role-structured system, including the untrained multi-agent pipeline and MANET, and the catalog placeholder in each prompt expands to the twelve capability names together with brief descriptions of their function and scope.

```
Planner. You are a core-network service planning
assistant. Decompose the user intent into sequential
sub-task steps that together fulfill it. Each step
describes one sub-task and corresponds to exactly one
capability from the CAPABILITY CATALOG (sensing or compute).
{capability_catalog} # USER INTENT #: {user_intent} Output
ONLY a valid JSON array of step strings. No markdown, no
explanation.
```

```
Assembler. You are a core-network capability-selection
assistant. Given the user intent and the task steps,
select the set of sensing/compute capabilities needed to
fulfill the intent. Choose only capability names from
the CAPABILITY CATALOG. Output capability names only, no
arguments, no ordering required. {capability_catalog} #
USER INTENT #: {user_intent} # TASK STEPS #: {task_steps}
Output ONLY a valid JSON array of capability-name strings.
```

```
Workflower. You are a core-network workflow-sequencing
assistant. Arrange the given sensing/compute capabilities
into a valid execution order: if one capability consumes
another's output, it must come later. Use only the
capabilities provided. {capability_catalog} # USER
INTENT #: {user_intent} # CAPABILITIES TO ORDER #:
{input_capabilities} Output ONLY a valid JSON array of
capability-name strings in execution order.
```

The LLM-based judge of Section 3.3 returns a single holistic verdict on the final workflow under the prompt below, and its TaskBench counterpart shares the same structure, with the tool list and the reference steps, nodes, and links in place of the catalog and the two workflows.

```
You are an evaluator for core-network task automation.
Decide whether the PREDICTED workflow (an ordered list of
sensing/compute capabilities) can accomplish the user's
intent. {catalog} User intent: {intent} Reference workflow
(one valid solution, for context only): {ref} Predicted
workflow: {pred} The prediction does NOT need to match the
reference, since a different capability set or order that
fulfils the intent is acceptable. Output 0 for critical
errors: a capability the intent truly needs is missing, a
clearly irrelevant capability is included, a capability is
placed before the data it needs exists, or the list is empty
or contains names outside the catalog. Otherwise output 1.
Output ONLY JSON: {"correct": <0 or 1>}
```

H LIMITATIONS AND FUTURE WORK

Limitations. This work instantiates NTA with the catalog of a single operator, whose twelve capabilities and precedence relation, although drawn from production use, cover only a portion of the services that a core network may expose, so that the behavior of MANET on larger and more heterogeneous catalogs remains to be examined. The evaluation further assesses workflows against the answer key and the precedence relation rather than executing them in a live network, which yields a controlled and reproducible testbed but cannot capture runtime conditions such as fluctuating traffic loads, contention for shared computing resources, or partial failures of the deployed services.

Potential. Rather than a new learning algorithm, the core contribution of this work lies in bringing multi-agent intelligence into production networks, a setting in which language agents have so far remained largely confined to proposals and case studies. To this end, NETBENCH makes measurable the demands of NTA that general-purpose benchmarks rarely capture, wherein capabilities lie beyond the coverage of pretraining, dependencies render any violating workflow unexecutable, and intents admit several valid solutions. By showing that specialized agents jointly trained with verifiable rewards can effectively compile operator intents into executable workflows, this work lays a concrete foundation for the agentic operation of the next-generation core network, where the value of automation grows with the scale and complexity of the services that the network orchestrates.

Future work. The most immediate extension of this work incorporates actual execution into the training loop, since the digital-twin simulation contained in the catalog allows the digital twin of the network to serve as a training environment, whose feedback on workflows executed before deployment complements the verifiable rewards with signals that no answer key can provide. The ordering bottleneck identified in Section 5.2 further motivates the induction of precedence relations from execution traces, which would allow the framework to follow evolving catalogs with only the affected role-specific adapters updated as new capabilities emerge. An adaptive role structure, in which the planner instantiates additional roles as the compositional depth of an intent grows, finally offers a promising path towards the closed-loop agentic automation of real-world operator networks.