OPENAGENCY: SELF-EVOLVING MULTI-AGENT SYSTEM VIA GOVERNED COLLABORATION PROTOCOLS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents have demonstrated strong capabilities in solving complex tasks, motivating the development of multi-agent systems (MAS) that coordinate agents with specialized roles. However, existing multi-agent collaboration paradigms are often static, with collaboration patterns determined a priori and remaining unchanged during execution. Enabling continuously improving collaboration requires protocols that balance flexibility and controllability: flexibility allows agents to explore diverse coordination strategies at runtime, while controllability ensures that such decisions can be used to improve collaboration. Existing collaboration protocols typically fall into two extremes: they are either rigidly predefined, limiting adaptation, or fully free-form, sacrificing oversight. In this paper, we introduce OpenAgency, a self-evolving multi-agent system built upon governed collaboration protocols. These protocols enable agents to dynamically select their next collaborators within a predefined yet extensible space at runtime, transforming the collaboration pattern itself into an evolvable object. To evolve collaboration protocols, OpenAgency maintains execution traces from agent interactions and introduces a collaboration-driven evolving agent that distills these experiences into a bounded number of atomic protocol improvements. Through this mechanism, OpenAgency allows multi-agent systems to continuously refine their collaboration strategies and adapt to diverse complex tasks while preserving control over the evolution process. Experiments across fourteen benchmarks spanning general question answering, long-horizon reasoning tasks, and production-oriented workloads demonstrate that OpenAgency achieves improved collaboration performance and continues to enhance its effectiveness through self-evolution.[1]

1 INTRODUCTION

Large language models (LLMs) (OpenAI, 2022) have shown strong capabilities on increasingly complex tasks, including software engineering (Yang et al., 2024b; Wang et al., 2025; Zhang et al., 2024a), data research (Liu et al., 2026; Li et al., 2024), and long-horizon planning (Zhang et al., 2026; Xu et al., 2024). Building upon these capabilities, LLM-based multi-agent systems further improve task-solving ability by assigning agents complementary roles and enabling them to exchange intermediate information through communication (Wang et al., 2024; Xi et al., 2023; Hong et al., 2024; Qian et al., 2024; Wu et al., 2023). However, as tasks become longer, more diverse, and increasingly open-ended, the collaboration process itself becomes a new source of failure, since ineffective coordination, unnecessary interactions, and inappropriate delegation accumulate errors throughout execution. A multi-agent system therefore has to keep improving the way its agents collaborate, and not only the way each agent reasons.

However, evolving collaboration is non-trivial as it asks for two properties that pull against each other. Flexibility lets a system explore different coordination patterns and compare what they produce, which is how a better way to cooperate is found. Controllability lets the system identify which coordination decision contributed to success or failure and verify that a revised collaboration remains valid. Without flexibility, evolution degenerates into selecting among a fixed set of predefined

[1] Code is available in the supplementary material.

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.

strategies. Without controllability, improvement becomes blind optimization guided only by final task rewards, with little account of how or why a collaboration pattern changes.

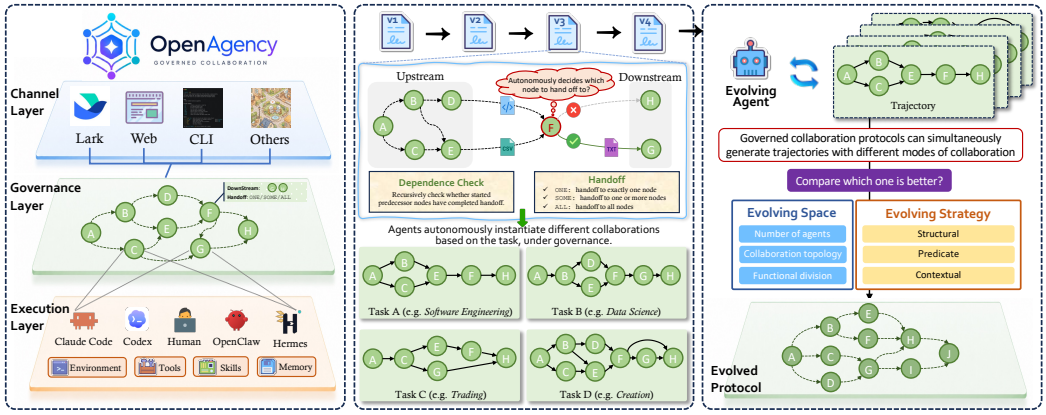
Existing collaboration protocols typically provide only one of these properties, which limits how far they can be evolved. *Blueprint-driven* approaches encode collaboration as graphs, workflows, or standard operating procedures (Hong et al., 2024; Wu et al., 2023; Fourney et al., 2024), and later work automates such designs through workflow search or generated agent programs (Zhang et al., 2025b; Hu et al., 2025; Zhang et al., 2025a). Because the coordination structure is written down, these approaches are controllable, yet once a workflow is instantiated the execution follows a predetermined route and the collaboration cannot be adapted locally when an unexpected situation arises. *Agent-driven* approaches let communication patterns, debates, or agent connectivity emerge at runtime (Li et al., 2023; Du et al., 2023; Liu et al., 2024; Park et al., 2023), and learned orchestrators adjust agent activation orders from feedback (Dang et al., 2025). These approaches are flexible, yet the coordination they realize stays implicit and hard to attribute or revise across runs. In essence, neither line offers a collaboration protocol that is adaptable during execution and revisable afterwards.

In this paper, we make the collaboration protocol itself carry both properties, so that the collaboration pattern becomes an evolvable object. To this end, we introduce *OpenAgency*, a self-evolving multi-agent system built upon *governed collaboration protocols*. As illustrated in Figure 1, a protocol specifies the agent roles and the handoffs admitted between them, fixing a set of valid coordination choices without committing to one execution path. An agent selects its next collaborator at runtime from inside that set, which supplies the flexibility, while the set rules out every transition the protocol does not admit, which supplies the controllability. Both properties therefore live in the same protocol, where each transition is checkable before it is taken and the collaboration pattern becomes something a later run can inspect and revise. On top of this protocol, OpenAgency adds an evolution loop that keeps improving the deployed collaboration. Each run leaves a trace that records accepted handoffs, rejected handoffs, completion events, and failures, so the coordination evidence prior systems discard stays available afterwards. A collaboration-driven evolving agent reads that trace and proposes at most four atomic edits over the protocol. A validity operator filters malformed topologies and the same agent scores the coordination each surviving candidate realizes, so every deployed round stays admissible and traceable to the events that motivated it. To the best of our knowledge, OpenAgency is the first multi-agent system that treats the collaboration pattern itself as an evolvable object, improving it from execution traces while runtime validity and interpretability are preserved. We summarize our contributions as follows.

- **Evolvable Collaboration Protocol:** We make the collaboration pattern itself a governed and evolvable object, because the admissible handoff set is checkable before a transition is taken and revisable after the run that produced it.
- **Collaboration-driven Evolution:** We propose an evolving agent that learns from the coordination evidence prior systems discard. A validity operator and a collaboration quality check keep every deployed round admissible and free of regression.
- **Strong Performance:** Experiments across fourteen benchmarks spanning general question answering, long-horizon reasoning, and production-oriented tasks show that OpenAgency reaches the best primary metric on every benchmark, improving over unstructured multi-agent initialization by an average of 20.7 points.

2 Related Work

Multi-agent Collaboration. Existing systems expose collaboration with different degrees of structure. Blueprint-driven approaches encode graphs, workflows, or standard operating procedures (Hong et al., 2024; Qian et al., 2024; Wu et al., 2023). These systems make coordination explicit but typically fix the selected topology before execution. Agent-driven systems allow collaboration to emerge from debate (Du et al., 2023; Liang et al., 2023), role-conditioned dialogue (Li et al., 2023; Chan et al., 2023), or dynamic connectivity (Liu et al., 2024; Zhang et al., 2024b). Collaboration then adapts during execution, though the realized structure remains difficult to inspect and replay. Recent methods also optimize communication for accuracy or efficiency (Sun et al., 2023; Chen et al., 2024a;b). Across both families, the collaboration pattern is treated as a fixed design choice or as an emergent byproduct, and neither case leaves a protocol that later runs can update.



(a) Multi-agent Architecture (b) Governed Collaboration Protocols (c) Collaboration-driven Protocol Evolution

Figure 1: Overview of OpenAgency. (a) Three layers keep the collaboration protocol independent of the backend that realizes each role. (b) Under one governed protocol the agents choose their own handoffs, so different tasks instantiate different collaborations. (c) The evolving agent reads the collected trajectories and returns the next protocol, moving inside the evolving space under one of three edit strategies.

Agent Self-evolution. Agent systems can improve through reasoning feedback, memory, prompt optimization, or program search. Reflexion and Self-Refine use rollout feedback to revise subsequent reasoning (Shinn et al., 2023; Madaan et al., 2023), while AgentGym and experiential co-learning accumulate experience across interactions (Xi et al., 2024; Qian et al., 2023). DSPy, TextGrad, and OPRO optimize language-model programs or prompt strings (Khattab et al., 2024; Yuksekgonul et al., 2024; Yang et al., 2024a). More recently, AFlow, automated agent design, and MaaS search over workflow graphs or generated agent programs (Zhang et al., 2025b; Hu et al., 2025; Zhang et al., 2025a), while evolving orchestration trains a central controller to reorder agent activations (Dang et al., 2025). These methods show that collaboration structure can be optimized, but what they return is an executable topology or a controller policy, neither of which exposes a boundary that a later round can check a candidate revision against.

In this paper, we aim to make the collaboration protocol itself an evolvable object. OpenAgency defines a governed protocol that fixes admissible handoffs and completion conditions without committing to one route, so agents keep local choices during execution while every decision remains checkable, which is what makes the protocol revisable from traces across runs. Unlike workflow search and orchestrator learning, which optimize the executed route or the policy that produces it, OpenAgency evolves the governance boundary that admits those runtime decisions.

3 OpenAgency

OpenAgency treats collaboration as a first-class object separated from the executors that realize each role. It combines a layered architecture (Section 3.1), a governed protocol that makes coordination inspectable (Section 3.2), and bounded protocol evolution driven by execution traces (Section 3.3), as Figure 1 illustrates.

3.1 Multi-agent Architecture

To apply one collaboration protocol across applications with heterogeneous executors, OpenAgency separates multi-agent collaboration into three layers that keep the protocol independent of backend-specific prompts and actions.

Channel Layer. The channel layer exposes the collaboration to its users through an enterprise chat workspace, a web console, or a command line, and keeps instructions, agent messages, human feedback, and coordination events on one shared timeline.

Governance Layer. The governance layer hosts the collaboration protocol and validates each proposed handoff against the current protocol state. It records both accepted and rejected events in a trace that supplies the evidence for later protocol evolution.

Execution Layer. The execution layer binds each role to a concrete executor. An executor can be an LLM agent, a coding harness, a domain-specific tool, or a human reviewer, and it sits below the governance layer, so the protocol constrains how roles hand off whichever backend realizes a role.

We formalize a running OpenAgency instance at step t as $\mathcal{I}^{(t)} = (\mathcal{X}^{(t)}, \mathcal{P}, \mathcal{E}, s^{(t)})$, where $\mathcal{X}^{(t)}$ is the shared context, $\mathcal{P}$ is the governed protocol, $\mathcal{E}$ is the set of available executors, and $s^{(t)}$ is the current protocol state. The map $\rho : \mathcal{R} \to \mathcal{E}$ binds each role in the role set $\mathcal{R}$ to an executor and stays separate from the protocol logic. The governance layer renders the shared state into a role-specific execution context $u^{(t)} = f_{\mathrm{ctx}}(\mathcal{X}^{(t)}, s^{(t)}, n, r, \mathcal{P})$ for the active node n with assigned role r, since an LLM executor may require tool instructions while a human executor may require a concise protocol description. Adapters return messages, handoffs, finishes, and failures in a common event format, so the executor bound to a role can change without changing the handoff rules, trace format, or completion predicates.

3.2 Governed Collaboration Protocols

To govern collaboration without fixing a complete execution route, OpenAgency represents collaboration as a declarative tuple that exposes enough structure to validate each handoff while preserving the agent's choice among admissible targets.

Protocol Definition. OpenAgency represents a governed collaboration protocol as

$$\mathcal{P} = (\mathcal{R},\, \mathcal{N},\, \alpha,\, \mathcal{H},\, \mu,\, \Omega,\, \mathcal{C}), \tag{1}$$

where $\mathcal{R}$ is the set of roles, $\mathcal{N}$ the set of collaboration nodes, $\alpha\colon \mathcal{N} \to \mathcal{R}$ assigns each node to a role, $\mathcal{H} \subseteq \mathcal{N} \times \mathcal{N}$ defines admissible handoff edges, $\mu(n) \in \{\texttt{ONE}, \texttt{SOME}, \texttt{ALL}\}$ is the handoff mode, Ω_n is the completion predicate, and $\mathcal{C}$ contains the shared `rule` and `culture` that every role sees.

Admissible Handoff Set. For a node n under state s, the admissible handoff set is $\Gamma(n, s) = \{\, n' \in \mathcal{N} \mid (n, n') \in \mathcal{H}$ and the transition is valid under $s \,\}$. The handoff mode $\mu(n)$ then determines how the active agent selects targets from $\Gamma(n, s)$. Under `ONE` the agent picks exactly one admissible target, under `SOME` a nonempty subset, and under `ALL` every admissible target. A node with no admissible target completes without a handoff, and the local choice within $\Gamma(n, s)$ is left to the agent.

Handoff Validation and Completion. When the agent proposes a handoff (n, B, m) with target set B and message m, the protocol validates B under $\mu(n)$. An admissible handoff commits B as pending targets and updates the state, and an inadmissible one leaves the state unchanged and returns structured feedback so the agent may propose another target. Committing a target set does not activate it. A finish event (n, y) reports the output y that the source node produced, and the governance layer activates the committed targets only when the completion predicate $\Omega_n(y)$ holds on that output, so downstream execution waits for the completion predicate even after an admissible target set is committed. Activation runs one further dependence check, which walks back from each committed target and confirms that every started predecessor of that target has already handed off, so a node begins only once the collaborators it depends on have delivered.

Algorithm 1 states the resulting runtime interpreter. The state advances only on admissible handoffs and valid completions, while the trace keeps every event, including the ones the boundary refused, so later evolution reads both successful and unsuccessful coordination.

3.3 Collaboration-driven Protocol Evolution

Given a protocol whose decisions are explicit and checkable, the remaining question is how to revise it from experience. Revising a protocol takes the shape of an optimization problem, since it needs a signal that says where the current protocol is wrong, a space of admissible moves, and a rule that decides which move to keep. The final answer alone cannot indicate which of a run's many coordination events should change, and regenerating the whole workflow each round would destroy

Algorithm 1 Collaboration under a Governed Protocol

Input: Protocol $\mathcal{P}$, initial context $\mathcal{X}$, executor mapping $\rho\colon \mathcal{R} \to \mathcal{E}$
Output: Collaboration trace τ

$\tau \leftarrow \emptyset$; activate start node n_0; obtain initial state s
while active nodes exist **do**
  for each active node n with role $r = \alpha(n)$ **do**
    $e \leftarrow \rho(r)\big(f_{\mathrm{ctx}}(\mathcal{X}, s, n, r, \mathcal{P})\big)$ ▷ executor proposes an event
    if e is a handoff (n, B, m) and B is admissible **then**
      commit B as pending targets and update s
    else if e is a finish (n, y) and $\Omega_n(y)$ holds **then**
      mark n complete, then activate every committed target whose started predecessors have handed off
    else
      return structured feedback to $\rho(r)$ ▷ refused at the governance boundary
    append (e, s) to τ ▷ accepted and refused events are both retained
return τ

the governance boundary that makes a change inspectable. A single evolving agent therefore drives the loop. It reads a search direction off the collaboration trace, moves inside a bounded space of protocol edits, and keeps a move only when that move survives an admissibility and quality check.

Trace Diagnosis as Search Signal. Each execution produces an ordered trace $\tau = (e_1, s_1), \ldots, (e_T, s_T)$ over the T coordination events of that run and the states they produce, where an event e_t is one of accepted handoff, rejected handoff, finish, message, or failure. Unlike prior systems that keep only the successful trajectory or the final score, the trace retains rejected handoffs and failed completions, which record the transitions the collaboration attempted and could not take. The evolving agent runs an analyzer f_{diag} over $\tau^{(1:v)}$, the traces collected in evolution rounds 1 through v, and returns the coordination sites that account for the observed failures, namely the handoff edges the agents repeatedly attempted and the admissible set refused, together with the completion predicates that fired without a usable output. This signal plays the role a gradient plays in parameter optimization, since it attributes a run-level outcome to individual coordination decisions and points at the part of the protocol a revision should touch.

Factorized Evolving Space. The search space factorizes into three levels. *Number of agents* covers the node count $|\mathcal{N}|$. *Collaboration topology* covers the handoff edge set $\mathcal{H}$ and the handoff modes μ. *Functional division* covers the role assignment α, the completion predicates Ω, and the shared `rule` and `culture` in $\mathcal{C}$, which together fix what each node owes the collaborators downstream of it. A point in this space is not one execution route, because each agent still chooses inside $\Gamma(n, s)$, so a single protocol unfolds into many realized collaboration patterns at inference time, as the four task instantiations of Figure 1 show. The optimizer therefore searches far fewer points than the collaborations they realize. The factorization also makes comparison readable, since a candidate produced by a single-level edit shares the remaining levels with the deployed protocol, so an overloaded role can be told apart from a missing edge.

Bounded Protocol Optimizer. A move in this space is a composite edit of at most k atomic changes drawn from three edit strategies. A *structural* edit adds or removes a node or a handoff edge, a *predicate* edit rewrites a completion or transition condition, and a *contextual* edit rewrites the `rule` and `culture` a role reads. At round v the optimizer f_{evolve} reads the deployed protocol $\mathcal{P}^{(v)}$ together with the analyzed trace signal and returns the edit set $\Delta^{(v)}$ to apply,

$$\Delta^{(v)} = f_{\mathrm{evolve}}\big(\mathcal{P}^{(v)},\ f_{\mathrm{diag}}(\tau^{(1:v)})\big), \qquad \big|\Delta^{(v)}\big| \le k, \tag{2}$$

where $\hat{\mathcal{P}}^{(v+1)}$ denotes the candidate obtained by applying $\Delta^{(v)}$ to $\mathcal{P}^{(v)}$, and the hat marks a candidate not yet checked. We set $k = 4$, which bounds the step the way a learning rate bounds a parameter update and keeps each candidate a small inspectable change. The strategy is a bounded greedy ascent, proposing one candidate per round and retaining the deployed protocol whenever the candidate fails the check below, so the deployed sequence never moves downhill in collaboration quality.

Admissibility and Quality Check. A step is checked twice before it is deployed. The first check is a structural validity test $\Pi_{\mathcal{V}}$ on the candidate topology, which returns 1 when the candidate is a well-formed protocol and 0 when it carries malformed handoff edges, unresolved role references,

duplicate node identifiers, or completion states no path can reach. The second check executes a candidate that passes the first one and hands the resulting traces to the evolving agent. The agent returns a collaboration quality score q_{col} from the realized coordination, reading whether each role delivered what its successors required and whether the completion conditions closed on a usable output. The deployment rule is

$$\mathcal{P}^{(v+1)} = \begin{cases} \hat{\mathcal{P}}^{(v+1)}, & \Pi_{\mathcal{V}}\big(\hat{\mathcal{P}}^{(v+1)}\big) = 1 \text{ and } q_{\text{col}}\big(\hat{\mathcal{P}}^{(v+1)}\big) \geq q_{\text{col}}\big(\mathcal{P}^{(v)}\big), \\ \mathcal{P}^{(v)}, & \text{otherwise}, \end{cases} \tag{3}$$

the candidate is deployed only when it is well formed and does not degrade the observed coordination.

4 Experiments

4.1 Benchmarks

We evaluate OpenAgency on fourteen benchmarks organized into three categories. *General question answering* contains HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), and HotpotQA (Yang et al., 2018), and tests protocol adaptation when each instance admits a short answer. *Long-horizon tasks* contains SWE-bench Verified (Jimenez et al., 2024), DeepPlanning (Zhang et al., 2026), and the text-only subset of Humanity's Last Exam (HLE), and tests governed handoffs on dependent decisions. *Production-oriented tasks* contains DevEval (Li et al., 2024), DDR-Bench (Liu et al., 2026), TheAgentCompany (Xu et al., 2024), MARBLE (Zhu et al., 2025), AutomationBench (Shepard & Salimans, 2026), and OfficeQA Pro (Opsahl-Ong et al., 2026), and tests one protocol across heterogeneous tools and task formats. Appendix A describes each benchmark.

4.2 Experimental Setup

For fairness, we use GPT-5.5 as backbone across all benchmarks. For each benchmark, we compare three conditions, a single-agent ablation, the initialized multi-agent system v_0, and the evolved protocol after four rounds (v_4). All three are reported against a prior system (Prior Sys.), the best of the officially reported results and our GPT-5.5 implementation for each benchmark. Each benchmark evolves its own protocol, and the evolving agent reads only execution traces, so no task label or reference answer reaches it, following the label-free self-refinement setting of Shinn et al. (2023). Model, temperature, and tool budget stay fixed across rounds, and only the protocol changes.

4.3 Main Results

Capability on General Question Answering. The five single-turn benchmarks probe *short-chain reasoning capabilities*, since each instance has a localized, automatically verifiable output. As shown in Table 1, OpenAgency obtains the best primary metric on all five, reaching 100.0 pass@1 on HumanEval and lifting MBPP from 73.5 to 80.9 over the initialization. Agent count alone does not produce this gain, since the initialization trails the single-agent ablation on HotpotQA at 64.0 against 75.9 F1 while the governed protocol reaches 82.3. Evolution prunes handoffs whose upstream output the trace flags as ineffective, and each benchmark settles on a solve-critique-revise loop matched to its dominant failure mode. Overall, OpenAgency's strong results here highlight its capability to reshape a protocol even where a short answer suffices.

Capability on Long-horizon Tasks. SWE-bench Verified, DeepPlanning, and HLE stress *long-horizon reasoning capabilities*, since a bad handoff at any stage can invalidate the entire run. Table 1 reports a lead on all three, reaching 86.2 file match on SWE-bench Verified against a prior system of 42.8 and 37.7 accuracy on HLE. More importantly, the protocol each benchmark converges to shows why. On SWE-bench Verified the evolving agent installs a retry edge when a patch fails and splits an overloaded Verifier into structural and semantic components, DeepPlanning hands its Verifier the constraints one at a time, and HLE makes verification conditional on answer confidence. All three shapes come from the same initialization, so the handoff and completion rules adapt when local errors are unrecoverable. Overall, OpenAgency's strong results here highlight its capability to keep dependent decisions recoverable.

Table 1: Results of OpenAgency on each benchmark. Prior Sys. is the best of the officially reported results and our GPT-5.5 implementation for each benchmark. Multi-agent (init) is the unstructured initialization and OpenAgency is the evolved protocol, both sharing the GPT-5.5 backbone. Each row uses the benchmark's native metric, where EM is exact match, FM is file match, and AT is the acceptance-test pass rate.

Benchmark	Metric	Prior Sys.	Single-agent	Multi-agent (init)	OpenAgency
		General Question Answering Tasks			
HumanEval	pass@1	92.2	87.5	96.9	**100.0**
MBPP	pass@1	69.3	73.5	73.5	**80.9**
GSM8K	EM	97.6	97.2	97.4	**98.0**
MATH	EM	83.3	82.3	86.0	**87.0**
HotpotQA	F1	76.4	75.9	64.0	**82.3**
		Long-horizon Tasks			
SWE-bench Verified	FM	42.8	75.6	67.0	**86.2**
DeepPlanning	Case Acc	46.9	45.0	33.8	**54.2**
HLE (text-only)	Accuracy	18.0	18.4	5.6	**37.7**
		Production-oriented Tasks			
DevEval	AT	62.5	62.5	82.1	**96.9**
DDR-Bench	Overall	69.1	42.8	44.0	**76.0**
TheAgentCompany	Score	41.5	46.3	41.5	**53.7**
MARBLE	Overall	79.3	78.4	50.0	**98.9**
AutomationBench	Acc	42.5	23.7	23.6	**54.8**
OfficeQA Pro	Acc@1%	57.1	35.3	21.1	**69.9**

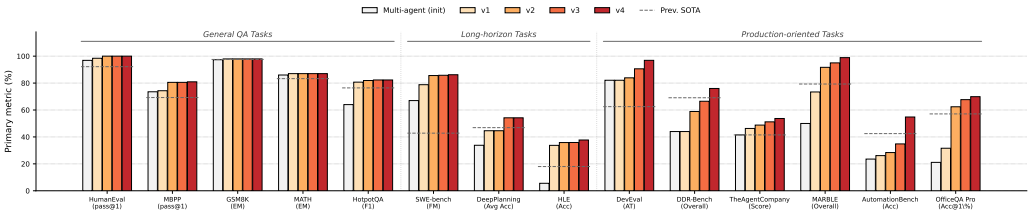


Figure 2: Round-wise trajectory of the primary metric from the initialization v_0 through the evolved rounds v_1–v_4. Grey dashed lines mark each benchmark's prior system.

Capability on Production-oriented Tasks. The six production-oriented benchmarks test *cross-domain deployment capabilities*, since each exposes a different set of tools. As shown in Table 1, OpenAgency leads on all six, improving over the prior system by 34.4 points on DevEval, 19.6 points on MARBLE, and 12.8 points on OfficeQA Pro. The template differs per benchmark, and in particular two cases expose what governance contributes. On DDR-Bench, an Auditor role introduced during evolution forces every quantitative finding in the trace into the final report, fixing a silent omission free-form baselines leave undetected. The admissible-set constraint on MARBLE keeps each handoff terminating where any-to-any messaging would not, so governance supplies a constraint the task format alone does not. Overall, OpenAgency's strong results here highlight its capability to carry one governed protocol across heterogeneous tools.

5 Analysis

5.1 Effectiveness of Protocol Evolution

To verify that bounded edits keep improving collaboration after the first revision, we track the primary metric from the initialization v_0 to the evolved rounds v_1 through v_4 on all fourteen benchmarks. Figure 2 and Table 5 record an improvement over the initialization on every benchmark, reaching 48.9 points on MARBLE and 48.8 points on OfficeQA Pro. The deployed sequence is non-decreasing in collaboration quality, since the check keeps the deployed protocol whenever a candidate scores lower, and on these fourteen benchmarks the ordering also holds for the primary task metric.

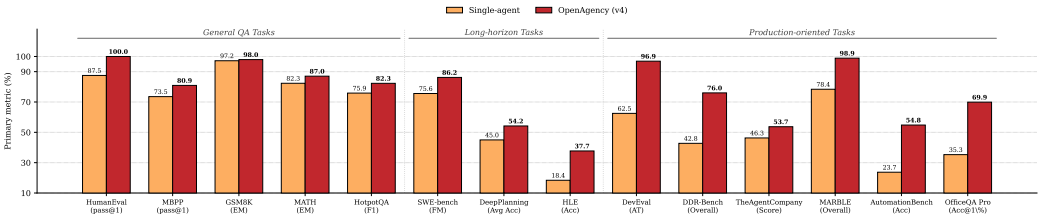


Figure 3: OpenAgency versus its single-agent ablation under one model and tool budget.

5.2 Detailed Improvement across Evolution

To locate the gain inside a benchmark, we decompose the SWE-bench Verified and DevEval results of every round into their component measurements. As shown in Table 2, the two improve at different points. On SWE-bench Verified the file-match rate given a patch moves only from 84.6 to 86.2, while the fraction of instances producing a patch at all rises from 79.2 to 100.0, so the gain is consistent with closing the no-output failure mode that the retry edges of Section 3.3 target. On DevEval the unit-test rate stays near saturation while acceptance rises from 82.1 to 96.9, so the admitted edits track the repository-level correctness that unit tests leave unchecked.

Table 2: Improvements across evolution.

Component	**Init**	v_1	v_2	v_3	v_4
SWE-bench Verified					
Patch applied	79.2	93.4	100.0	100.0	**100.0**
File match	67.0	78.8	85.6	85.8	**86.2**
FM given a patch	84.6	84.4	85.6	85.8	**86.2**
DevEval					
Unit test	98.1	98.1	97.3	99.1	**99.1**
Acceptance test	82.1	82.1	83.9	90.6	**96.9**

5.3 Ablation of Evolving Space

To identify which levels of the evolving space carry the gain, we restrict that space in two nested steps and rerun the same evolution loop on three benchmarks. The full setting edits all three levels of Section 3.3, the fixed-count setting holds the agent count $|\mathcal{N}|$, and the division-only setting freezes the whole graph. Figure 4 reports the best protocol each setting reaches, and the three benchmarks order the three settings identically. Freezing the agent count alone costs 5.8 points on SWE-bench Verified, 12.6 on DevEval, and 15.3 on DDR-Bench, so the freedom to change the agent count is the level that contributes most, and freezing the remaining handoff structure costs a further 2.3 to 5.2 points. In essence, a hand-designed system commits both structural levels before reading any trace.

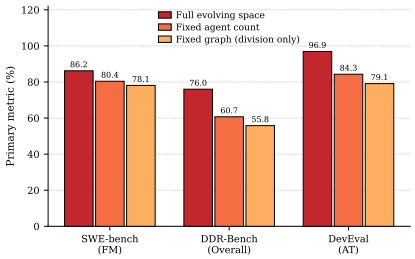


Figure 4: Effect of three evolving spaces. The fixed-count setting holds the agent number, and the division-only setting freezes the whole graph.

5.4 Comparison against Free-form and SOP Baselines

To compare the three ways of expressing collaboration, we evaluate OpenAgency against the free-form baseline AutoGen (Wu et al., 2023) and the SOP baselines ChatDev (Qian et al., 2024) and MetaGPT (Hong et al., 2024). As shown in Table 3, OpenAgency obtains the highest score, exceeding the strongest SOP baseline by 26.4 points on SWE-bench Verified. Notably, the free-form baseline is the weakest everywhere, and the SOP baselines recover part of the gap because the collaboration order is enforced, yet their fixed procedure absorbs no trace-level failure signal.

Table 3: Results of OpenAgency against the free-form and SOP baselines.

System	**SWE-bench**	**Dev Eval**	**DDR-Bench**	**TheAgent Company**
AutoGen	51.4	68.9	39.2	38.7
ChatDev	57.6	78.3	44.1	41.9
MetaGPT	59.8	79.5	46.8	42.3
OpenAgency	**86.2**	**96.9**	**76.0**	**53.7**

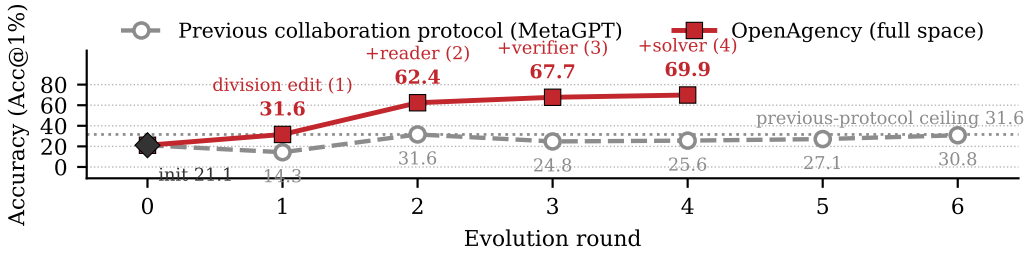


Figure 5: Evolution of OpenAgency and MetaGPT on OfficeQA Pro.

5.5 Advantage of Multi-agent Protocols

To attribute the gain to governance and not to agent count, we compare OpenAgency with its single-agent ablation under one GPT-5.5 executor and identical tool access. As shown in Figure 3, OpenAgency exceeds that ablation on every benchmark, with the widest gaps on OfficeQA Pro, DevEval, and DDR-Bench. A single node interleaves solving and checking inside one context, while the governed protocol routes a partial output to a downstream role that validates it independently. The initialization column of Table 1 agrees, since a multi-agent initialization without a governed protocol falls short.

5.6 Flexibility of Governed Protocols across Tasks

To show what one governed protocol deploys once the task changes, we read out the deployed v_4 protocol on three production-oriented benchmarks. As shown in Table 4, the three protocols differ in agent count, role vocabulary, and handoff order. DevEval converges on a coder checked by a reviewer, DDR-Bench on a researcher feeding a reporter, and TheAgentCompany on a planner, an executor, and a reviewer, so one loop settles on the division each task rewards. The admissible handoff set stays active on all three chains, ruling out 3.8% to 8.2% of proposed handoffs and changing the outcome of 2.9% to 5.3% of runs. A fixed procedure commits to one of these patterns before any trace is seen, and the refused events are the evidence the evolving agent consumes in Section 3.3.

Table 4: Governance activity on three deployed v_4 protocols with different handoff chains, as the share of proposed handoffs $\Gamma(n, s)$ rules out and the share of runs whose outcome that refusal changes.

	Dev Eval	**DDR-Bench**	**TheAgent Company**
Agents	2	2	3
Refused (%)	4.9	3.8	8.2
Recovered (%)	3.7	2.9	5.3

5.7 Superiority of Governed Protocols for Evolution

To measure what a fixed collaboration graph costs, we plot the OfficeQA Pro evolution record under two settings from one initialization. The previous-protocol setting applies MetaGPT (Hong et al., 2024) and rewrites only the functional division over a single-node graph. As shown in Figure 5, six such rewrites peak at 31.6 in round 2, and the four later ones fall back below it. The full setting, which may also change the agent count and the handoff structure, reaches 62.4 once a reader node is added and 69.9 after four rounds, matching Table 5. Both settings share the executor, the tool budget, and the 133 scored tasks, so the gap comes from the level an edit may reach. A fixed graph keeps the search inside a level the trace already reports as saturated, and one structural edit clears that ceiling by more than thirty points.

6 Conclusion

We introduced OpenAgency, a self-evolving multi-agent system whose governed collaboration protocol makes the collaboration pattern an evolvable object, updated from ordered traces under a validity check. Experiments on fourteen benchmarks show that OpenAgency leads the compared systems on every primary metric, and that the gain keeps accumulating across the evolution rounds.

AI USE STATEMENT

In this work, we used generative AI tools only for final-stage language polishing and grammar refinement of the manuscript. We have not used generative AI tools for generating research ideas, designing methods, conducting experiments, implementing code, analyzing data, interpreting results, or producing scientific claims, and other generative AI-assisted tasks requiring disclosure are not applicable to this work. We have reviewed all AI-assisted revisions to ensure their correctness and consistency with the authors' intended content. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have made efforts to ensure the reproducibility of this work by providing detailed descriptions of the proposed method, experimental settings, implementation details, and evaluation protocols in the main paper and supplementary materials. The supplementary materials include the source code required to reproduce our experiments, and we will publicly release the code after the review process to further facilitate verification and future research. The content of this paper can be completely reproduced.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. Chateval: Towards better LLM-based evaluators through multi-agent debate, 2023. URL https://arxiv.org/abs/2308.07201.

Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. Reconcile: Round-table conference improves reasoning via consensus among diverse llms, 2024a. URL https://arxiv.org/abs/2309.13007.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Optima: Optimizing effectiveness and efficiency for LLM-based multi-agent system, 2024b. URL https://arxiv.org/abs/2410.08115.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, Xuantang Xiong, Lei Han, Zhiyuan Liu, and Maosong Sun. Multi-agent collaboration via evolving orchestration, 2025. URL https://arxiv.org/abs/2505.19591.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate, 2023. URL https://arxiv.org/abs/2305.14325.

Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Erkang Zhu, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, Peter Chang, Ricky Loynd, Robert West, Victor Dibia, Ahmed Awadallah, Ece Kamar, Rafah Hosn, and Saleema Amershi. Magentic-one: A generalist multi-agent system for solving complex tasks, 2024. URL https://arxiv.org/abs/2411.04468.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS Track on Datasets and Benchmarks*, 2021. URL https://arxiv.org/abs/2103.03874.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, 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, 2024. URL https://arxiv.org/abs/2308.00352.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems, 2025. URL https://arxiv.org/abs/2408.08435.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.06770.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan A, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=sY5N0zY5Od.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society, 2023. URL https://arxiv.org/abs/2303.17760.

Jia Li, Ge Li, Yunfei Zhao, Yongmin Li, Huanyu Liu, Hao Zhu, Lecheng Wang, Kaibo Liu, Zheng Fang, Lanshen Wang, Jiazheng Ding, Xuanming Zhang, Yuqi Zhu, Yihong Dong, Zhi Jin, Binhua Li, Fei Huang, and Yongbin Li. DevEval: A manually-annotated code generation benchmark aligned with real-world code repositories. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024. URL https://arxiv.org/abs/2405.19856.

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, 2023. URL https://arxiv.org/abs/2305.19118.

Wei Liu, Peijie Yu, Michele Orini, Yali Du, and Yulan He. Hunt instead of wait: Evaluating deep data research on large language models, 2026. URL https://arxiv.org/abs/2602.02039.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic llm-powered agent network for task-oriented agent collaboration, 2024. URL https://arxiv.org/abs/2310.02170.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. 2023.

OpenAI. Introducing ChatGPT. https://openai.com/blog/chatgpt, 2022.

Krista Opsahl-Ong, Arnav Singhvi, Jasmine Collins, Ivan Zhou, Cindy Wang, Ashutosh Baheti, Owen Oertell, Jacob Portes, Sam Havens, Erich Elsen, Michael Bendersky, Matei Zaharia, and Xing Chen. OfficeQA Pro: An enterprise benchmark for end-to-end grounded reasoning, 2026. URL https://arxiv.org/abs/2603.08655.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior, 2023. URL https://arxiv.org/abs/2304.03442.

Chen Qian, Yufan Dang, Jiahao Li, Wei Liu, Zihao Xie, Yifei Wang, Weize Chen, Cheng Yang, Xin Cong, Xiaoyin Che, Zhiyuan Liu, and Maosong Sun. Experiential co-learning of software-developing agents, 2023. URL https://arxiv.org/abs/2312.17025.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. Chatdev: Communicative agents for software development, 2024. URL https://arxiv.org/abs/2307.07924.

Daniel Shepard and Robin Salimans. AutomationBench: Evaluating agents on cross-application business workflows, 2026. URL https://arxiv.org/abs/2604.18934.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Qiushi Sun, Zhangyue Yin, Xiang Li, Zhiyong Wu, Xipeng Qiu, and Lingpeng Kong. Corex: Pushing the boundaries of complex reasoning through multi-model collaboration, 2023. URL https://arxiv.org/abs/2310.00280.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6), March 2024. ISSN 2095-2236. doi: 10.1007/s11704-024-40231-1. URL http://dx.doi.org/10.1007/s11704-024-40231-1.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for ai software developers as generalist agents, 2025. URL https://arxiv.org/abs/2407.16741.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation, 2023. URL https://arxiv.org/abs/2308.08155.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. The rise and potential of large language model based agents: A survey, 2023. URL https://arxiv.org/abs/2309.07864.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. Agentgym: Evolving large language model-based agents across diverse environments, 2024. URL https://arxiv.org/abs/2406.04151.

Frank F. Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z. Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. TheAgentCompany: Benchmarking LLM agents on consequential real world tasks, 2024. URL https://arxiv.org/abs/2412.14161.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations (ICLR)*, 2024a.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018. URL https://arxiv.org/abs/1809.09600.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. TextGrad: Automatic "differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet, 2025a. URL https://arxiv.org/abs/2502.04180.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations (ICLR)*, 2025b. URL https://arxiv.org/abs/2410.10762.

Yinger Zhang, Shutong Jiang, Renhao Li, Jianhong Tu, Yang Su, Lianghao Deng, Xudong Guo, Chenxu Lv, and Junyang Lin. DeepPlanning: Benchmarking long-horizon agentic planning with verifiable constraints, 2026. URL https://arxiv.org/abs/2601.18137.

Yuntong Zhang, Haifeng Ruan, Zhiyu Fan, and Abhik Roychoudhury. AutoCodeRover: Autonomous program improvement. In *ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*, 2024a.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö. Arik. Chain of agents: Large language models collaborating on long-context tasks, 2024b. URL https://arxiv.org/abs/2406.02818.

Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, and Jiaxuan You. MultiAgentBench: Evaluating the collaboration and competition of LLM agents, 2025. URL https://arxiv.org/abs/2503.01935. MARBLE framework and benchmark; code at https://github.com/MultiagentBench/MARBLE.

A Benchmark Details

HumanEval (Chen et al., 2021) evaluates function-level Python code synthesis from natural-language docstrings.

MBPP (Austin et al., 2021) evaluates short-form program synthesis from natural-language descriptions.

GSM8K (Cobbe et al., 2021) evaluates multi-step reasoning over grade-school math word problems.

MATH (Hendrycks et al., 2021) evaluates competition-level mathematical reasoning across five subject areas.

HotpotQA (Yang et al., 2018) evaluates multi-hop question answering that links evidence from multiple documents.

SWE-bench Verified (Jimenez et al., 2024) evaluates repository-level software repair on human-validated GitHub issues, where the agent must produce a patch that resolves the reported bug against a hidden test suite.

DeepPlanning (Zhang et al., 2026) evaluates travel and shopping planning under user-specified constraints, and the primary metric is mean case-level accuracy across the two task families.

Humanity's Last Exam (HLE) evaluates expert-level question answering across mathematics, physics, biology, humanities, computer science, and engineering, and we use its 2,158-question text-only subset.

DevEval (Li et al., 2024) evaluates repository-level code generation under acceptance-test oracles that check whether the generated code works end-to-end with existing modules.

DDR-Bench (Liu et al., 2026) evaluates deep data research over 10-K SEC filings, and the overall metric averages message-wise correctness (per-question factuality) and trajectory-wise correctness (multi-turn consistency).

TheAgentCompany (Xu et al., 2024) evaluates end-to-end task completion in a simulated enterprise environment, where each task requires a full containerized stack.

MARBLE (Zhu et al., 2025) evaluates open-ended research and buyer-seller negotiation under rubric-based scores within the MultiAgentBench framework.

AutomationBench (Shepard & Salimans, 2026) evaluates cross-application business-workflow orchestration through REST APIs, and the primary metric is accuracy across its 600 tasks.

OfficeQA Pro (Opsahl-Ong et al., 2026) evaluates enterprise-grade grounded numerical reasoning over the U.S. Treasury Bulletin corpus, with correctness scored under a 1% numerical tolerance.

B Case Study of Protocol Evolution

Figure 6 shows how the protocol evolves across rounds on the nine long-horizon and production-oriented benchmarks, which jointly cover repository-level coding, planning, retrieval, workflow orchestration, and open-ended research. On every benchmark the number of nodes remains small (≤ 4 across rounds), so evolution acts as a rewiring process over a protocol that stays small across very different tasks.

The three edit levels are unevenly used. Structural edits dominate, appearing in 22 of the 27 revisions in Figure 6, and the remaining five revise a predicate or the shared context without touching the graph. DevEval is the clearest case, where the second round moves unit-test injection into the shared rule and the third round bounds the review loop and records known error patterns, both without adding a role. Both rounds change how the protocol behaves while leaving the graph fixed, which a static topology has no place to record.

On the coding and planning benchmarks, evolution mainly redistributes verification responsibility. The SWE-bench Verified protocol first attaches a retry edge from the Verifier back to the Editor, then re-routes the Verifier back to the Locator, and finally splits the Verifier into structural and semantic components. DevEval gains a self-test node between the Coder and the Reviewer. DeepPlanning

grows from a monolithic solver into a Researcher-Solver-Verifier chain with an explicit constraint checklist.

On the retrieval and research benchmarks, evolution adds evidence-preserving roles. The DDR-Bench protocol fixes the Researcher-to-Reporter handoff to forward the raw evidence set, splits a single reporting phase into two, and installs a quantitative Auditor. OfficeQA Pro converts a monolithic solver into a Reader-Solver-Verifier chain over an external document store. MARBLE replaces the initial Reviewer-Brainstormer-Writer handoff with a compact Generator-Critic-Reviser pipeline and adds a feasibility-driven retry edge from the Critic back to the Generator.

Evolution also shrinks the graph when coordination is unnecessary. AutomationBench collapses a two-node Executor-Verifier pipeline into a single-node solver with progressively tightened internal phases, while HLE grows a confidence-routed subgraph that skips verification on high-confidence answers and adds a retry edge on refused ones. Section 5.2 locates the component measurement on which the accepted edits pay off.

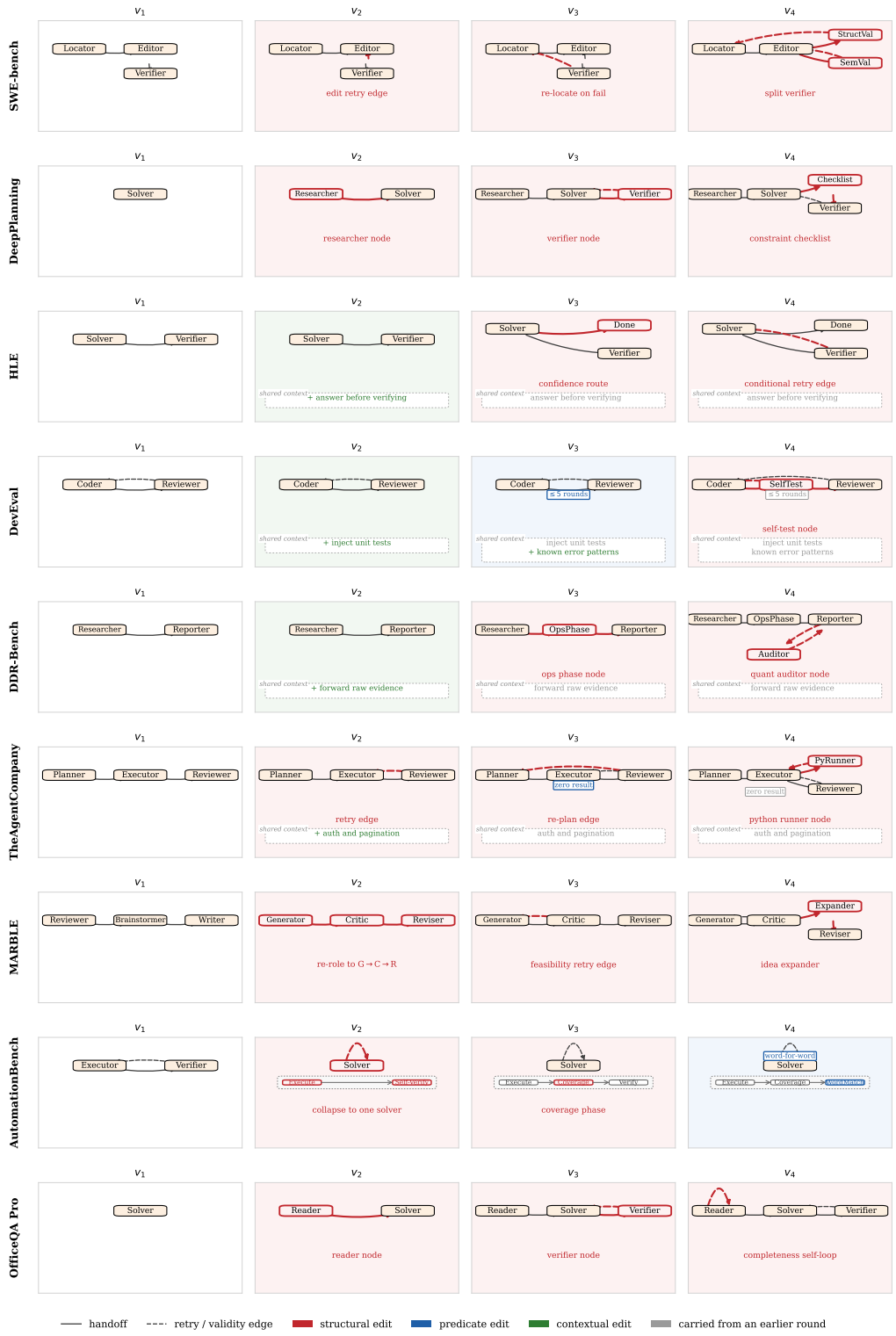


Figure 6: Protocol evolution across rounds on the nine long-horizon and production-oriented benchmarks, one per row. Solid ovals are agent roles, pills inside a dashed container are internal phases of a single-agent solver, solid edges are admissible handoffs, and dashed edges are validity or retry edges. Boxed text pinned to an element is a transition or completion predicate, and the lower container holds the shared rule and culture. Elements introduced by the current round carry the color of their edit level from Section 3.3 and earlier elements are gray, so a round that leaves the graph unchanged still shows what it revised.

Table 5: Primary metric across the four evolution rounds, each row in the benchmark's native metric.

Benchmark	**Single-agent**	**Multi-agent (init)**	v_1	v_2	v_3	v_4
HumanEval	87.5	96.9	98.4	100.0	100.0	**100.0**
MBPP	73.5	73.5	74.3	80.5	80.5	**80.9**
GSM8K	97.2	97.4	98.0	98.0	98.0	**98.0**
MATH	82.3	86.0	87.0	87.0	87.0	**87.0**
HotpotQA	75.9	64.0	80.7	81.9	82.3	**82.3**
SWE-bench Verified	75.6	67.0	78.8	85.6	85.8	**86.2**
DeepPlanning	45.0	33.8	44.6	44.6	54.2	**54.2**
HLE (text-only)	18.4	5.6	33.7	35.8	35.9	**37.7**
DevEval	62.5	82.1	82.1	83.9	90.6	**96.9**
DDR-Bench	42.8	44.0	44.0	58.9	66.5	**76.0**
TheAgentCompany	46.3	41.5	46.3	48.8	51.2	**53.7**
MARBLE	78.4	50.0	73.4	91.7	95.0	**98.9**
AutomationBench	23.7	23.6	26.1	28.4	34.8	**54.8**
OfficeQA Pro	35.3	21.1	31.6	62.4	67.7	**69.9**

Table 6: Performance on HumanEval (Chen et al., 2021). The metric is pass@1.

Model / System	**pass@1 (%)**
Direct LLM	92.2
OpenAgency	**100.0**

C Per-benchmark Detailed Results

This appendix reports the per-benchmark results summarized in Table 1, and we also include the results of several other systems (taken from the leaderboard) as performance references. Table 5 lists the primary metric of all fourteen benchmarks across the four evolution rounds. Tables 6, 7, 8, 9, and 10 detail the general question answering benchmarks, Tables 11, 12, and 13 the long-horizon benchmarks, and Tables 14, 15, 16, 17, 18, and 19 the production-oriented benchmarks.

Table 7: Performance on MBPP (Austin et al., 2021). The metric is pass@1.

Model / System	pass@1 (%)
Direct LLM	69.3
OpenAgency	**80.9**

Table 8: Performance on GSM8K (Cobbe et al., 2021). The metric is exact-match accuracy.

Model / System	Exact match (%)
Direct LLM	97.6
OpenAgency	**98.0**

Table 9: Performance on MATH (Hendrycks et al., 2021). The metric is exact-match accuracy.

Model / System	Exact match (%)
Direct LLM	83.3
OpenAgency	**87.0**

Table 10: Performance on HotpotQA (Yang et al., 2018). The metric is token-level F1.

Model / System	F1 (%)
Direct LLM	76.4
OpenAgency	**82.3**

Table 11: Performance on SWE-bench Verified (Jimenez et al., 2024). FM is file match, and the first three rows are reported by that work.

Model / System	%Apply	FM	%Resolved
Claude-2	–	–	4.8
GPT-4	–	–	1.7
SWE-Llama-13B	–	–	3.9
OpenHands	72.4	42.8	15.4
OpenAgency	**100.0**	**86.2**	**74.4**

Table 12: Performance on DeepPlanning (Zhang et al., 2026). Avg is the mean of travel and shopping case accuracy. External rows are reported by that work.

Model / System	Avg	Tra-CS	Tra-PS	Tra-Comp	Tra-Acc	Shop-M	Shop-Acc
Grok-4.1-Fast	3.0	39.6	3.0	29.6	0.0	50.1	5.9
GLM-4.7	7.1	38.9	7.1	30.7	0.0	61.2	14.2
Qwen-Plus	7.5	37.3	7.5	25.1	0.0	63.9	15.0
Seed-1.8	11.3	43.0	11.3	45.3	0.0	68.1	22.5
Kimi-K2-Thinking	12.1	45.2	32.5	38.9	0.0	65.8	24.2
DeepSeek-V3.2	21.6	47.4	35.0	41.2	0.7	78.8	42.5
Gemini-3-Pro	23.2	58.4	25.1	41.8	0.7	78.0	45.8
o3	24.9	76.5	55.6	66.1	11.3	76.9	38.5
Claude-Sonnet-4.5	25.5	65.2	58.4	61.8	7.6	80.0	43.3
Qwen3-Max	28.7	64.0	61.7	62.8	13.8	82.6	43.5
Gemini-3-Flash	28.8	67.1	57.7	62.4	5.9	80.6	51.7
GPT-5	31.6	78.7	65.9	72.3	18.9	80.4	44.2
Claude-Opus-4.5	33.9	79.3	70.9	75.1	22.7	80.0	45.0
GPT-5.2	44.6	88.5	83.3	85.8	35.0	84.8	54.2
Qwen-Agent tool-calling	46.9	88.8	56.7	72.7	32.9	86.8	60.8
OpenAgency	**54.2**	**90.1**	**85.6**	**87.3**	**38.4**	**87.5**	**62.1**

Table 13: Performance on the Humanity's Last Exam text-only subset. External rows are reported by the benchmark authors.

Model / System	Accuracy (%)
GPT-4o	2.7
Grok-2	3.2
Claude-Sonnet-3.5	4.3
Gemini-1.5-Pro	5.3
Gemini-2.0-Flash-Thinking	8.1
o1	8.1
o3-mini	18.0
Direct LLM	13.4
OpenAgency	**37.7**

Table 14: Performance on DevEval (Li et al., 2024). UT is the unit-test pass rate and AT is the repository-level acceptance-test pass rate.

Method	UT	AT
Direct LLM	90.4	62.5
OpenAgency	**99.1**	**96.9**

Table 15: Performance on the DDR-Bench 10-K subset (Liu et al., 2026). Overall averages Message-Wise and Trajectory-Wise accuracy. External rows are reported by that work.

Model / System	Overall	Message-Wise	Trajectory-Wise
Proprietary models			
Gemini-2.5-Pro	20.1	24.5	15.6
Gemini-3-Flash	33.0	44.8	21.2
GPT-5.1	40.7	37.1	44.3
GPT-5-mini	42.0	46.8	37.1
GPT-5.2	43.0	44.9	41.1
Claude-Sonnet-4.5	69.1	77.6	60.6
Open-source models			
Llama-3.3-70B	6.5	9.9	3.0
Qwen2.5-72B	20.3	27.1	13.4
Qwen3-30B-A3B	28.4	42.4	14.4
MiniMax-M2	35.6	44.2	26.9
Qwen3-Next-80B-A3B	37.8	44.8	30.8
Kimi-K2	41.0	51.1	30.8
GLM-4.6	48.2	60.3	36.0
DeepSeek-V3.2	49.2	60.1	38.2
DDR reference agent	60.5	61.3	59.6
OpenAgency	**76.0**	76.1	**75.9**

Table 16: Performance on TheAgentCompany (Xu et al., 2024). Score is the aggregated task score, and Steps is the average number of steps. External rows are reported by that work.

Model / System	Backbone	Success (%)	Score (%)	Steps (↓)
OpenHands	Gemini-2.5-Pro	30.3	39.1	33.5
OpenHands	Claude-Sonnet-3.7	26.3	34.7	40.9
OpenHands	Claude-Sonnet-3.5	24.0	34.4	29.2
OpenHands	Gemini-2.0-Flash	11.4	19.0	39.9
OpenHands	GPT-4o	8.6	16.7	28.7
OpenHands	GPT-5.5	40.0	41.5	31.6
OpenAgency	GPT-5.5	**46.7**	**53.7**	**22.8**

Table 17: Performance on MARBLE (Zhu et al., 2025). Overall averages Research and World scores. External rows are reported by that work.

Model / System	Backbone	Research (%)	World (%)	Overall (%)
Meta-Llama-3.1-8B	Llama-3.1-8B	80.9	72.8	76.9
Meta-Llama-3.1-70B	Llama-3.1-70B	80.8	72.1	76.5
Meta-Llama-3.3-70B	Llama-3.3-70B	80.0	73.2	76.6
GPT-3.5-turbo	GPT-3.5-turbo	70.2	71.7	71.0
GPT-4o-mini	GPT-4o-mini	84.1	74.5	79.3
OpenAgency	GPT-5.5	**97.8**	**100.0**	**98.9**

Table 18: Performance on AutomationBench (Shepard & Salimans, 2026). External rows are reported by that work.

Model / System	Acc (%)
GPT-6-Sol	33.2
GPT-6-Astra	41.4
Claude-Opus-5.5	42.5
OpenAgency	**54.8**

Table 19: Performance on OfficeQA Pro (Opsahl-Ong et al., 2026). External rows are reported by that work.

Model / System	Corpus format	Acc@1% (%)
GPT-5-mini	Full PDF	30.8
GPT-5.4	Full PDF	36.1
Gemini-3.1-Pro	Full PDF	42.9
GPT-5.1	Full PDF	45.1
Claude-Opus-4.5	Full PDF	46.6
Claude-Opus-4.6	ai_parse_document	57.1
Direct LLM	Full PDF	26.3
OpenAgency	Full PDF	**69.9**