COLLABORCHESTRA: LEARNING TO ORCHESTRATE HUMAN-AGENT TEAMS

Anonymous authors
Paper under double-blind review

ABSTRACT

AI agents are increasingly deployed in occupational contexts, where many real-world scenarios involve humans and agents working together. Effective human-agent collaboration can outperform either party alone while preserving human agency, but requires understanding each other's capabilities and deciding when humans should delegate, verify, or intervene. We present **CollabOrchestra**, a practical method for learning an orchestrator policy that delegates work across a human-agent team. CollabOrchestra casts orchestration as *adaptive workflow plan generation*, where the orchestrator constructs and revises a sequence of steps specifying goals, assignees, and handoff deliverables. Our training proceeds in two stages: we first profile agent workers through sandboxed execution and learn a feasibility predictor from their execution traces, then train the orchestrator through multi-turn reinforcement learning to jointly optimize plan quality and execution feasibility. Across GDPval, JobBench and their underspecified variants, CollabOrchestra outperforms prompting-based orchestrators in both plan quality and execution outcome. The resulting team also outperforms the best solo agent, improving Task Performance from 0.474 to 0.744 on GDPval and from 0.260 to 0.286 on JobBench. Our analysis shows that CollabOrchestra learns to assign humans diverse, task-dependent roles, including directly performing work when appropriate. In a user study on participants' own tasks, users prefer CollabOrchestra-trained plans 16% more and report high collaboration legibility.

1 INTRODUCTION

The rapid advancement of AI agents has enabled their deployment across a broad range of human work (Patwardhan et al., 2025; Li et al., 2026; Wang et al., 2025b). Beyond fully autonomous deployment, many real-world settings involve humans and agents working together (Shao et al., 2026; Feng et al., 2026; Mozannar et al., 2025), with effective collaboration promising to outperform either party alone and unlock more insights and opportunities (Acemoglu et al., 2026). Yet as AI agents take on increasingly long-horizon tasks (Kwa et al., 2025), they sharpen a classic irony of automation: the more capable an automated system becomes, the more crucial the remaining human contribution may be, and the greater the need to support humans in deciding what to delegate, when to verify agent outputs, and when to intervene (Bainbridge, 1983).

One way to provide such support is to scaffold humans to move up the stack through intelligent delegation, where an AI orchestrator coordinates the division of work between humans and agents (Tomašev et al., 2026). However, prior ML approaches are designed for specific tasks like classification and navigation (Hemmer et al., 2023; Lai et al., 2022; Fuchs et al., 2024; 2023); directly prompting an LLM as the orchestrator can be applied to more tasks but may collapse delegation onto a single worker or produce seemingly balanced plans that are infeasible to execute (Masters et al., 2025; Su et al., 2026). A more principled approach is to post-train an LLM to act as an orchestrator that accounts for human and agent capabilities while learning effective teaming strategies, enabling feasible agent delegation and meaningful human involvement.

We present **CollabOrchestra**, a practical training method for an orchestrator that teams up human and agents at the workflow level, a higher level of abstraction compared to individual actions. CollabOrchestra builds on three technical ideas. First, we make workflow-level orchestration tractable by casting it as adaptive workflow plan generation (§2). The Orchestrator constructs and revises

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.

a structured division of work through interaction with the user. Second, we ensure this workflow plan can translate to feasible execution through agent worker profiling (§3.1). Because executing every candidate workflow during training is prohibitively expensive, we collect agent execution traces offline and learn a feasibility predictor that estimates whether proposed agent-assigned steps can be reliably executed. Third, we train the Orchestrator through multi-turn reinforcement learning with a simulated user, optimizing a two-term reward which combines (i) plan quality that captures coherent decomposition, effective handoffs, and appropriate human involvement with (ii) execution-grounded plan feasibility (§3.2). Together, these components enable the Orchestrator to learn feasible, user-adaptive divisions of work without requiring end-to-end execution during training.

We evaluate CollabOrchestra on a human-agent team equipped with a coding agent and a browser-use agent, two complementary agent types covering a broad range of real-world work. Across GDPval (Patwardhan et al., 2025), JobBench (Li et al., 2026), and their underspecified variants, CollabOrchestra consistently improves both plan quality and execution feasibility across three base models, outperforming prompting baselines even when the evaluation criteria are explicitly provided in the prompt. More importantly, these gains translate to better execution with simulated users, measured by Delivery Rate (whether the team produces a final deliverable within a predefined step limit) and Task Performance (the quality of the delivered outcome). CollabOrchestra substantially improves Delivery Rate over a rubric-enhanced prompting baseline, while the resulting human-agent team also outperforms the best solo agent on Task Performance, improving from 0.474 to 0.744 on GDPval and from 0.260 to 0.286 on JobBench. Our analysis further shows that CollabOrchestra training shifts the policy from primarily asking users to supply input and review outputs toward assigning diverse, task-dependent roles, including having users directly perform work when appropriate. In a user study where participants bring their own tasks, CollabOrchestra similarly adapts human involvement to each task, with its plans preferred 16% more often than those from the prompting baseline and report high collaboration legibility and perceived human agency. Together, these results demonstrate the promise of learned orchestration for helping humans effectively team up with AI agents on increasingly long-horizon tasks[1].

2 Preliminaries: Human-Agent Collaboration Workflows

We study human-agent collaboration at the workflow level. A workflow specifies how a task is decomposed, which parts are delegated to which workers, and what each worker should produce. Constructing such workflows requires global knowledge of each party's capabilities and careful judgment about when the user should get involved. Here, we introduce an Orchestrator policy that adaptively constructs and revises workflow plans for a team of human and agent workers.

2.1 Workflow Plan

Following Wang et al. (2025b), we define a *workflow* as a sequence of steps toward a goal, where each step accomplishes a distinguishable sub-goal and specifies what concrete deliverable should be produced to support handoffs between workers. Formally, given a task T and worker set $\mathcal{W} = \{U, A_1, \ldots, A_N\}$, a workflow plan is an ordered sequence of m steps: $\mathcal{P} = \{(g_i, w_i, d_i)\}_{i=1}^{m}$, where g_i denotes the step's goal, $w_i \in \mathcal{W}$ its assignee, and d_i the concrete deliverable it produces.

Human-agent collaboration workflow differs from standard agentic workflows in two important ways. First, each step specifies a goal g_i rather than concrete actions, leaving the execution strategy to the assigned worker. Second, the user is a potential assignee rather than a passive recipient, enabling human involvement where their input, judgment, or oversight is valuable. Such involvement matters not only when humans are better positioned to execute or supervise particular steps, but also for maintaining an understanding of how the task is being carried out and the ability to exercise control.

2.2 Orchestration as Adaptive Workflow Plan Generation

As illustrated in Figure 1, the Orchestrator is parameterized by a policy π_θ that maintains the global workflow plan $\mathcal{P}$ and decides how work should be distributed across the team. In contrast, the

[1]Our code and training data are available at `https://anonymous.4open.science/r/collabochestra-iclr-annonymous-repo-77E6/`

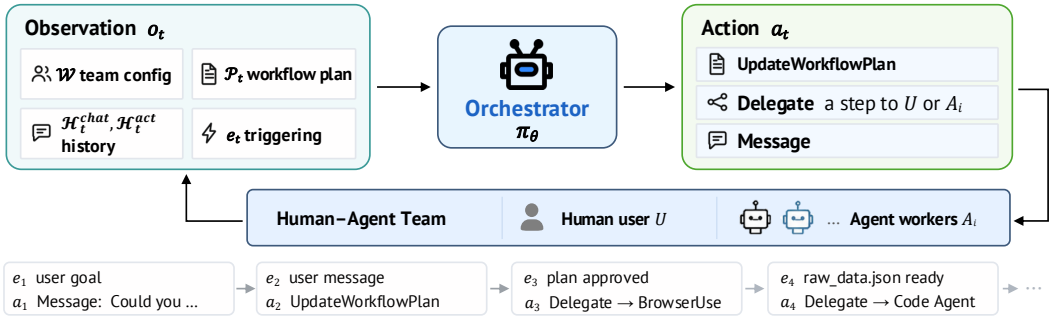

Figure 1: **Orchestrating the human-agent team at the workflow level.** The Orchestrator maintains the global workflow plan and decides how work should be distributed across the team.

human worker U (the user) and the agent workers A_i provide feedback or execution outcomes, and their behavior is not controlled by π_θ. Thus, the Orchestrator operates at the workflow level rather than directly controlling the low-level actions of individual workers. At each decision step t, the Orchestrator receives an observation $o_t = (\mathcal{W}, \mathcal{P}_t, \mathcal{H}_t^{\text{chat}}, \mathcal{H}_t^{\text{act}}, e_t)$ comprising: (1) the *team configuration* $\mathcal{W}$, where each agent worker is described by a natural-language specification and the human worker U is described by the user profile; (2) the *current workflow plan* $\mathcal{P}_t$ described in §2.1; (3) the *chat history* $\mathcal{H}_t^{\text{chat}}$, which contains the full transcript of messages exchanged between the Orchestrator and the user or agent workers; (4) the Orchestrator's own *action history* $\mathcal{H}_t^{\text{act}}$; and (5) the *triggering event* e_t, which is either a new user message or an agent worker status-and-log update.

Given o_t, the Orchestrator samples an action $a_t \sim \pi_\theta(\cdot \mid o_t)$ from three operations: (1) `UpdateWorkflowPlan`, which constructs or revises $\mathcal{P}$ as new information becomes available; (2) `Delegate`, which delegates a workflow step to a human or agent worker; and (3) `Message`, which communicates with the user to provide guidance or request clarification. We thus cast orchestration as **adaptive workflow plan generation**. Rather than producing a fixed plan in one shot, the Orchestrator can continually update the division of work in response to user input and intermediate outcomes.

3 CollabOrchestra

We parameterize the Orchestrator policy π_θ as a language model (see Appendix B.1) for the full prompt). An instruction-tuned language model can readily follow the action format and perform basic reactive coordination, such as revising a workflow after receiving a worker failure. The harder problem is learning *how the work should be divided*: which steps each agent can reliably complete, where human involvement is valuable, and how these choices should adapt to the user's needs.

To learn these decisions, we optimize the Orchestrator at the *planning level*. Given an unlabeled dataset of task descriptions $\mathcal{D}$, CollabOrchestra proceeds in two stages. First, we profile agent workers through sandboxed execution and train a *feasibility predictor* that estimates whether an agent can successfully complete a proposed workflow step (§3.1). Second, we post-train the Orchestrator through multi-turn reinforcement learning with a simulated user, optimizing the final workflow for both predicted feasibility and plan quality (§3.2). Figure 2 illustrates how the training approach updates the Orchestrator policy.

3.1 Agent Worker Profiling

Natural-language specifications in team configuration $\mathcal{W}$ provide coarse information like which tools an agent can access, but do not capture the nuances of its capabilities across different goals and deliverables. We model the observed capabilities of agent workers by collecting their execution outcomes and training an offline feasibility predictor $f_\varphi(g, w, d) \in [0, 1]$, which estimates the probability that worker w successfully completes goal g and produces the expected deliverable d.

Profiling Data Acquisition For each agent $A_i \in \{A_1, \ldots, A_N\}$, we construct (g, A_i, d) profiling instances from tasks in $\mathcal{D}$ in three steps: (1) expand a task prompt into a fully specified request with concrete deliverables; (2) materialize any required reference files in an isolated sandbox; and (3) decompose the request into workflow steps, each defined by a goal g and expected deliverable d. We then ask A_i to execute each step independently in the sandbox. For non-initial steps, we prepend a

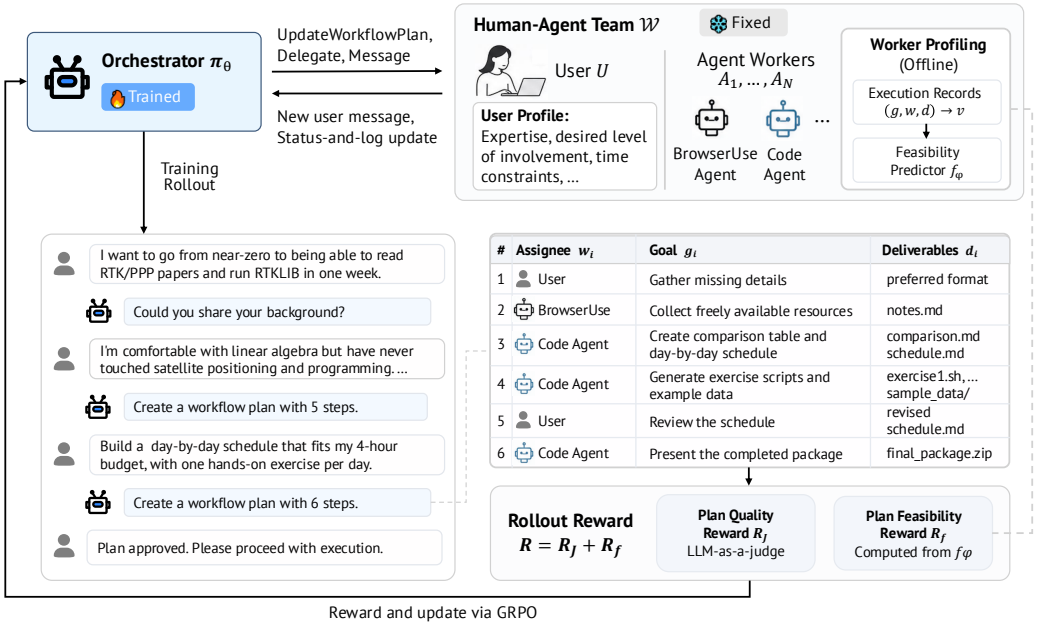

Figure 2: **CollabOrchestra overview.** We train the orchestrator with GRPO using rewards that capture both plan quality and execution feasibility. Appendix C provides the pseudo code.

summary of what successful upstream steps would have produced, approximating the handoff context available during actual workflow execution. An LLM judge evaluates the resulting output against the step goal and expected deliverable to obtain a binary success label $v \in \{0, 1\}$. Appendix B.2 provides the complete synthesis and evaluation prompts.

Feasibility Predictor Training We train $f_\varphi(g, w, d)$ using the acquired profiling data to predict the step feasibility without requiring live execution. (g, w, d) is serialized into a text record and encoded using a pretrained sentence encoder. The resulting representation is passed through a two-layer MLP that predicts the probability of success. We optimize f_φ using binary cross-entropy against v, weighting the positive class by $n_{\text{neg}}/n_{\text{pos}}$ to account for class imbalance. Appendix E validates f_φ quality and compares this lightweight approach with prompted and fine-tuned LLM alternatives: the sentence-encoder model achieves the best performance at substantially lower inference cost.

Estimate Workflow Plan Feasibility Given the learned f_φ, we estimate whether the agent-assigned portion of a workflow can be reliably executed by characterizing plan feasibility as

$$R_f(\mathcal{P}) = \frac{1}{2}\Big(\min_{1 \le i \le n} p_i + \frac{1}{n}\sum_{i=1}^{n} p_i\Big), \tag{1}$$

where $\min_{1 \le i \le n} p_i$ measures *Bottleneck Feasibility* capturing the weakest step and $\frac{1}{n}\sum_{i=1}^{n} p_i$ measures *Average Feasibility* capturing overall plan reliability. We track the latter separately at evaluation time because a workflow typically fails as soon as one delegated step fails, making it a more faithful predictor of end-to-end failure than the average.

3.2 Orchestrator Policy Training

We next train the Orchestrator to construct human-agent collaboration workflows that balance feasible delegation with meaningful human involvement.

User Simulation We associate the user U in the human-agent team with a textual user profile specifying factors relevant to collaboration, such as expertise, desired level of involvement, learning preference, *etc.*, and train the Orchestrator through multi-turn interactions with a simulated user conditioned on the user profile. The user simulator is instantiated with an LLM instructed to behave according to the given user profile and respond naturally to the evolving workflow. Since our training focuses on the planning stage, we restrict the simulator's action space to two actions during CollabOrchestra training: `SendMessage`, which allows the user to provide information, express preferences, ask questions, or give feedback on the proposed workflow; and `Finish`, which indicates

that the user is satisfied with the current workflow and ready to proceed with execution. The episode terminates when the simulated user issues `Finish` or a maximum number of turns is reached. Appendix B.3 provides the full simulator prompt and qualitative checks of the simulated behavior.

Composite Reward Our learning objective is to train the Orchestrator to construct workflows that achieve two complementary goals: (i) forming a high-quality division of work between the human and agent workers, operationalized with a judge-based *Plan Quality* reward R_J; and (ii) delegating work to agents that they can reliably execute, as captured by the execution-grounded *Plan Feasibility* reward R_f defined in §3.1. The final optimization target is

$$R(\mathcal{P}) \;=\; R_J(\mathcal{P}) \;+\; \alpha \cdot R_f(\mathcal{P}), \tag{2}$$

where α is a hyperparameter. To compute R_J, we condition an LLM judge on the task T and worker set $\mathcal{W}$ which includes the user profile, and ask it to judge along four dimensions: (1) *handoff quality*: whether each step specifies a concrete deliverable that provides the information or artifacts needed for downstream workers to proceed without unnecessary clarification; (2) *workflow coherence*, whether the steps form a complete, non-redundant sequence that achieves the original goal; (3) granularity appropriateness, whether steps represent meaningful units of work rather than micro-actions or overly broad bundles; and (4) *user involvement appropriateness*, whether the workflow involves the user where their contribution, understanding, or oversight is valuable while avoiding unnecessary mechanical work. Appendix B.4 provides the *Plan Quality* judge prompt and validation details.

Orchestrator Policy Update with Multi-Turn RL For each training instance $x = (T, \mathcal{W})$, the policy π_θ induces a multi-turn trajectory $\tau = (o_1, a_1, \ldots, o_{|\tau|}, a_{|\tau|})$, where each Orchestrator action $a_t \sim \pi_\theta(\cdot \mid o_t)$ is a token sequence and subsequent observations are produced through interaction with the simulated user. Let $\mathcal{P}^\star(\tau)$ denote the workflow plan committed at termination. We assign each trajectory a terminal reward $R(\mathcal{P}^\star(\tau))$ with no intermediate rewards, such that earlier decisions are credited according to their downstream effect on the final workflow rather than through manually specified per-turn rewards. We optimize this terminal reward using GRPO (Shao et al., 2024). For each instance x, we sample a group of G trajectories $\{\tau_k\}_{k=1}^{G}$ and compute their terminal rewards $R_k = R(\mathcal{P}^\star(\tau_k))$. We then compute a group-normalized advantage for each trajectory and assign it to every policy-generated token in τ_k, thereby optimizing the entire multi-turn planning process toward higher-quality and more feasible final workflows.

4 Experiments

We evaluate CollabOrchestra to orchestrate a human-agent team $\mathcal{W} = \{U, A_1, A_2\}$ comprising a human user U and two complementary agent workers: A_1, a BrowserUseAgent[2] for web-based tasks, and A_2, GeminiCLIAgent[3] for coding and terminal-based tasks, both powered by `gemini-3-flash`. We choose this configuration because these two agents are open-source and together cover a broad range of economically valuable tasks (Patwardhan et al., 2025; Boisvert et al., 2024). We instantiate the orchestrator on three base policies of increasing scale, Qwen3-4B-Instruct, Qwen3-30B-A3B-Instruct, and gpt-oss-120b.

We construct the training set from a seven-day snapshot of publicly accessible job postings on a remote-work platform. We use an LLM to rewrite each job description into two self-contained task prompts: a *specified* version that preserves the details provided in the original posting, and an *underspecified* version that omits selected information to elicit clarification and adaptive planning from the orchestrator. This yields 16,534 training instances. During training, we use a frozen `Qwen3-235B-A22B-Instruct` for both the Plan Quality reward R_J and user simulation, and set $\alpha = 0.5$. Appendix D provides further data, training, and hyperparameter details.

4.1 Evaluation Setup With Simulated User

We primarily evaluate with simulated users which provide a controlled testbed for comparing human-agent teaming strategies.

Planning Stage Evaluation We configure the user profile U to represent a balanced user with moderate desired level of involvement who is willing to spend time for iteration and refinement.

[2] `https://github.com/browser-use/browser-use`

[3] `https://github.com/google-gemini/gemini-cli`

Following the simulation protocol in §3.2, we use the same system prompt and action space (`SendMessage` and `Finish`) but instantiate the simulator with `gemini-3-flash` to avoid the policy learns to hack the simulator. We evaluate the final workflow plan using two metrics:

- PLAN QUALITY: We apply the same rubric used to define Plan Quality reward but use `gemini-3-flash` for grading, which is distinct from the judge model used during training. The original score ranges from 1 to 10 and is normalized to $[0, 1]$.
- BOTTLENECK FEASIBILITY: The lowest feasibility f_φ among agent-assigned steps, capturing the workflow's weakest delegation.

Workflow Execution Evaluation To assess whether the resulting workflow plans can be successfully carried out in practice, we freeze the final workflow plan and execute its steps sequentially without further orchestration in a sandbox containing the task's reference files, propagating completed outputs as handoff context. We instantiate U with a GeminiCLIAgent that simulates the user's execution behavior, allowing it both to respond to messages and to complete steps assigned to the user. Appendix F provides implementation details and characterizes the behavior of the simulated user. We set a 15-minute timeout for each agent step and a 3-minute timeout for each simulated user step to model moderate user involvement. Following Shao et al. (2026), we evaluate:

- DELIVERY RATE: A binary metric whether the human-agent team successfully deliver a task outcome after the execution.
- TASK PERFORMANCE: A task-specific scoring function evaluates the quality of the outcome for delivered class with scores normalized to $[0, 1]$.

4.2 BENCHMARK DATASETS AND BASELINES

We evaluate on four datasets spanning real-world economically valuable work: (1) **GDPval** (Patwardhan et al., 2025) contains 220 tasks across 44 O*NET occupations (National Center for O*NET Development, 2026) and nine major U.S. economic sectors; 125 tasks include reference files, and each has a human-authored rubric with 47.5 criteria on average. (2) **JobBench** (Li et al., 2026) contains 128 agentic tasks across 35 occupations, each with a workspace of heterogeneous reference files and a fact-anchored rubric averaging 35.6 binary criteria.

Because these benchmarks provide detailed task specifications upfront, we additionally construct underspecified variants to stress-test whether the Orchestrator can identify missing information and adapt the workflow through interaction with the user. This also reflects realistic settings where users understand the full task but may not communicate all relevant requirements in their initial request. We use `claude-sonnet-4` to rewrite each task in GDPval and JobBench into a "vague and brief" request while retaining the original description as hidden information available only to the user. This yields (3) **GDPval-underspecified** consisting of 220 tasks with 43.1 words on average, compared with 337.2 in the original GDPval, and (4) **JobBench-underspecified** consisting of 128 tasks with 48.7 words on average, compared with 308.1 in the original. During evaluation, the user simulator retains the full original task description as hidden context. Following Li et al. (2026), we use `grok-4-1-fast-reasoning` to grade Task Performance against each task's rubric.

Baselines To our knowledge, there is no existing training method for learning workflow-level delegation between human and agent workers. We therefore compare against two prompting-based Orchestrators using the same base policies: (1) **Direct Prompting**, which receives the task, worker set, and Orchestrator system prompt; and (2) **Rubric Prompting**, which additionally receives the Plan Quality criteria, testing whether explicitly specifying desirable orchestration behavior can substitute for training. For workflow execution, we additionally report **Best Solo Agent**, the best outcome from any $A_i \in \mathcal{W}$ completing the task autonomously.

5 RESULTS

5.1 COLLABORCHESTRA IMPROVES PLAN QUALITY AND EXECUTION OUTCOME

As shown in Table 1, CollabOrchestra consistently improves both Plan Quality and Bottleneck Feasibility across three base models and four benchmark variants. These improvements cannot be

Orchestrator Base Model	Method	GDPval		GDPval-underspecified		JobBench		JobBench-underspecified	
		Plan Quality	Bottleneck Feasibility	Plan Quality	Bottleneck Feasibility	Plan Quality	Bottleneck Feasibility	Plan Quality	Bottleneck Feasibility
Qwen3-4B	Direct Prompting	0.413	0.432	0.364	0.506	0.512	0.485	0.495	0.652
	Rubric Prompting	0.537	0.367	0.532	0.450	0.553	0.335	0.623	0.552
	CollabOrchestra	**0.722**	**0.700**	**0.768**	**0.811**	**0.646**	**0.720**	**0.692**	**0.848**
Qwen3-30B-A3B	Direct Prompting	0.510	0.407	0.518	0.435	0.655	0.287	0.610	0.508
	Rubric Prompting	0.631	0.352	0.666	0.336	0.742	0.187	0.780	0.349
	CollabOrchestra	**0.775**	**0.854**	**0.779**	**0.894**	**0.852**	**0.878**	**0.825**	**0.899**
gpt-oss-120B	Direct Prompting	0.409	0.595	0.395	0.511	0.420	0.390	0.483	0.615
	Rubric Prompting	0.691	0.560	**0.745**	0.664	0.765	0.539	**0.891**	0.713
	CollabOrchestra	**0.728**	**0.815**	0.707	**0.839**	**0.778**	**0.767**	0.858	**0.836**

Table 1: **Planning stage evaluation results.** Plan Quality is evaluated by `gemini-3-flash`, which is distinct from the judge model used during training. Bottleneck Feasibility is computed using the trained feasibility predictor f_φ.

Orchestrator Base Model	Method	GDPval		GDPval-underspecified		JobBench		JobBench-underspecified	
		Delivery Rate	Task Perf.	Delivery Rate	Task Perf.	Delivery Rate	Task Perf.	Delivery Rate	Task Perf.
/	Best Solo Agent	0.940	0.474	–	–	0.867	0.260	–	–
Qwen3-4B	Rubric Prompting	0.908	0.708	0.650	0.454	0.938	0.278	0.766	0.313
	CollabOrchestra	**0.970**	**0.741**	**0.952**	**0.645**	**0.977**	**0.282**	**0.805**	**0.364**
Qwen3-30B-A3B	Rubric Prompting	0.829	**0.659**	0.558	0.402	0.930	0.265	0.672	0.245
	CollabOrchestra	**0.957**	0.597	**0.952**	**0.474**	**0.984**	**0.274**	**0.812**	**0.301**
gpt-oss-120b	Rubric Prompting	0.637	0.521	0.512	0.371	0.646	0.204	0.459	0.162
	CollabOrchestra	**0.926**	**0.744**	**0.931**	**0.703**	**0.998**	**0.286**	**0.689**	**0.214**

Table 2: **Execution results with simulated user.** Delivery Rate and Task Performance (on delivered cases) for orchestrated human-agent teams using Gemini CLI Agent (Gemini-3-Flash) as the simulated user. CollabOrchestra raises delivery rate over the prompting baseline across all settings.

attributed simply to exposing the orchestrator to the grading criteria: although Rubric Prompting generally improves Plan Quality over direct prompting, explicitly training the policy through CollabOrchestra yields further gains in terms of both Plan Quality and Bottleneck Feasibility.

More importantly, these improvements translate into better execution outcomes. Table 2 shows that CollabOrchestra improves delivery rate over Rubric Prompting across all four benchmarks. The gains are especially significant when the initial task description is underspecified: On GDPval-underspecified, delivery rises from 0.650 to 0.952 for Qwen3-4B, from 0.558 to 0.952 for Qwen3-30B-A3B, and from 0.512 to 0.931 for gpt-oss-120b. The trained orchestrator also leads to more effective human-agent teaming in terms of task performance compared to the human-agent team orchestrated with Rubric Prompting method and the best solo agent. GDPval performance reaches 0.741, 0.597, and 0.744 for Qwen3-4B, Qwen3-30B-A3B, and gpt-oss-120b as orchestrator base model, respectively, compared with 0.474 for the best solo agent; on JobBench, the corresponding scores are 0.282, 0.274, and 0.286, compared with 0.260. These results show that rather than merely producing plans that look better to a judge, CollabOrchestra yields human-agent workflows that are more reliably completed and, in most settings, produce higher-quality outcomes.

5.2 CollabOrchestra Assigns Human To Different Roles

We analyze what steps are assigned to the human in the final workflow plan surfaced by CollabOrchestra. On GDPval with gpt-oss-120b as the backbone model, we deterministically classify each human-assigned step based on the verb lexicon of its goal g_i (*e.g.*, "review*", "confirm*", "design*", "provide*", *etc.*) into three roles: "*review/approve*", "*supply input*", "*do the work*", which correspond to the collaboration patterns previously identified in Shao et al. (2026) including enhancing human control, communication, and distribution of work based on expertise. Appendix G gives the deterministic coding rules. Most human-assigned steps fall in the *Review/approve* (82.8%), but the learned policy also assigns 12.5% of human steps for the human to do the work which generally encompasses two categories: closing out the last hop of the task ("deliver*", "close*"), especially

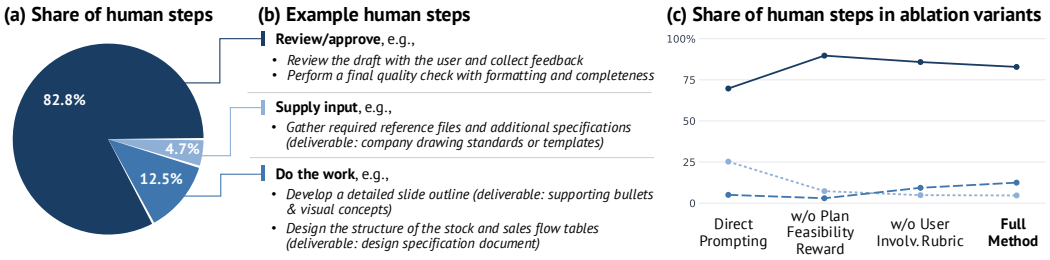

Figure 3: **Analysis of human steps in orchestrated human-agent collaboration.** We categorize human-assigned steps in GDPVal workflows produced by different policies and report their shares.

Measure	Description	Response distribution (Scale 1-5)	Mean
PLAN QUALITY			
Handoff quality	Each step produces a clear output that the next step can build on.		3.84
Workflow coherence	The steps form a complete, logical sequence that achieves the goal.		3.96
Granularity fit	Each step represents a meaningful, self-contained unit of work.		3.76
Involvement fit	The plan assigns me tasks that require human judgment.		3.96
LEGIBILITY			
Agent action clarity	I have a clear understanding of what the AI agents are doing.		3.75
User role clarity	I know what I am expected to do during the session.		3.69
AGENCY			
Action clarity	I know what actions I can take to get what I need from this system.		3.73
Autonomy	I feel like I have meaningful control over how the plan is carried out.		3.73
OUTCOME			
Goal achievement	The session effectively addresses the goal I started with.		3.75
Overall satisfaction	Overall, I am satisfied with this system.		3.62

Table 3: **User study post-session ratings.** Circle size indicates the percentage of responses at each level and an open black circle marks a level that received no responses.

when addressed to a downstream recipient such as a manager, client, or leadership team; and authoring at an early stage of the task ("draft", "write", "create", or "design"), especially on open-ended tasks.

To understand how this teaming strategy emerged from training, we compare CollabOrchestra against Direct Prompting and two ablations: "w/o Feasibility Reward", which removes the feasibility term from the training reward, and "w/o User Involv. Rubric" which removes the User Involvement Appropriateness component of of the Plan Quality reward during training. As shown in Figure 3(c), both ablations reduce *do the work* assignments to 9.3% and 3.0%, respectively; without the Feasibility Reward, 89.7% of human steps concentrate on *review/approve*. In contrast, Direct Prompting assigns 25.3% of human steps to *supply input*, despite GDPval prompts being largely specified upfront. These patterns that intelligent delegation is not reducible to a fixed rule-based handoff policy: different reward terms induce qualitatively different human roles, separating review, feasibility checking, and meaningful execution as distinct forms of collaboration.

6 User Study

While simulated users enable controlled evaluation at scale, they cannot fully capture how people perceive and participate in AI-orchestrated collaboration. We therefore conduct a user study with 24 participants recruited through Upwork and social media, using tasks drawn from participants' own work. For each task, participants first interact side-by-side with two Orchestrators: gpt-oss-120b[4] trained with CollabOrchestra and the same base model with Rubric Prompting. Policy identities are hidden, left-right order is randomized, and participants may interact with either Orchestrator in any order before selecting their preferred workflow plan. To avoid requiring participants to execute the task twice, they then proceed only with the plan produced by CollabOrchestra, with CollabOrchestra continuing to orchestrate its execution. After each session, participants rate ten statements on a five-point Likert scale: four corresponding to our Plan Quality rubric and six adapted from the Explainable AI User Experiences framework (Liao et al., 2020), covering collaboration legibility,

[4]We use gpt-oss-120b because it was available for serving through Tinker during the user study period.

human agency, and overall satisfaction. We collect 55 valid task sessions in total. Appendix H provides recruitment details, study interface and example human-agent teaming trajectories.

Users prefer plans produced by the learned orchestrator. Participants prefer the CollabOrchestra plan in 31 of 55 tasks (56.4%), compared with 22 for Rubric Prompting and 2 ties. Consistent with §5.2, CollabOrchestra involves humans in more diverse activities: 32.9% of its workflow steps are human-assigned, of which 17.0% ask the user to *do the work*, compared with 15.6% human-assigned steps and no *do the work* steps under Rubric Prompting. In post-session ratings, User Involvement Appropriateness and Workflow Coherence are the highest-rated items (3.96 out of 5).

Learned orchestration remains legible while preserving user agency. All ten questionnaire items are rated significantly above the neutral midpoint of 3 (all $p < .001$, Holm-corrected). Participants report understanding what the agents are doing (3.75) and what is expected of them (3.69), while maintaining a sense of autonomy (3.73). In practice, participants complete 86.2% of their assigned steps; unfinished steps are predominantly final review or confirmation steps where participants choose to leave with the existing output. However, free-text feedback reveals heterogeneous preferences for participation. Some participants value visibility and opportunities to contribute (*e.g.*, "Compared to the black box feel of traditional AI, this AI provides clearer steps, which makes it more trustworthy." from P23), while others prefer faster execution and find CollabOrchestra slower than a fully autonomous agent. Although CollabOrchestra models these preferences through user profiles, reliably eliciting them from real users remains an important direction for future work.

7 Related Work

Intelligent Delegation in Human-Agent Collaboration Effective human-agent collaboration requires deciding what to delegate, where human verification is needed, and when a request requires further clarification. These decisions are non-trivial even for discerning users, especially when they are collaborating with AI agents on a task that is new to them (Laban et al., 2026; Padmakumar et al., 2026; Guo et al., 2024). Intelligent delegation instead uses an AI model to orchestrate the human-agent team (Tomašev et al., 2026). Existing approaches largely follow two paradigms. *Heuristic approaches* allocate work using fixed confidence thresholds (Fügener et al., 2022), task types (He et al., 2025), or hand-crafted human roles in multi-agent systems (Wu et al., 2024; Hong et al., 2024). *Learning-based approaches* learn delegation from data but typically operate at the action level, deciding whether the human or AI performs the next action in domains such as classification, content moderation, driving, and navigation (Hemmer et al., 2023; Lai et al., 2022; Fuchs et al., 2024; 2023). In contrast, real occupational tasks often require coordinated sequences of interdependent steps, where each delegation decision depends on the surrounding workflow.

Multi-turn Training In multi-turn training, a policy takes multiple rounds of actions in an interactive environment before receiving a reward. The underlying optimization methods for multi-turn training can extend from single-turn reinforcement learning, including PPO with a learned reward model (Schulman et al., 2017) and group-relative methods such as GRPO (Shao et al., 2024). Prior work has applied multi-turn training to dialogue, text games, and agent-environment interaction (Zhou et al., 2024; Abdulhai et al., 2025; Wang et al., 2025a), while another line extends RLHF and preference optimization to multi-turn settings (Shani et al., 2024; Shi et al., 2024). More closely related to our setting, recent work trains language models through multi-turn user interaction. CollabLLM trains a policy through multi-turn interaction with an LM-based user simulator, enabling it to reason about how an action affects subsequent interaction rather than optimizing each turn in isolation. We adopt this paradigm for intelligent delegation among human-agent teams.

8 Conclusion

We present CollabOrchestra, a practical method for learning an orchestrator policy that delegates work across human-agent teams at the workflow level. CollabOrchestra models agents' observed capabilities through offline profiling and learns to jointly optimize plan quality and execution feasibility while adapting to user input. Across four occupational benchmarks, learned orchestration improves both workflow plans and execution outcomes over prompting-based orchestrators. Our user study further suggests that explicit, adaptive workflow planning can make human-agent collaboration more legible while preserving perceived human agency.

AI USE STATEMENT

In this work, LLMs were used to assist with code development and as LLM-based judges for grading final deliverables in GDPval and JobBench, following the exact evaluation procedures associated with these two benchmarks. We manually checked the generated code, grading outputs, and reported results. LLMs were not used to autonomously determine the paper's claims, analyses, or conclusions. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This paper presents CollabOrchestra, the first practical training method for learning an orchstrator policy that delegates work across a human-agent team. While the learned policy discovers team-up plans that assign meaningful work to the user and enables the human-agent team to achieve better outcomes, several ethical considerations should be addressed when applying CollabOrchestra in practice to team up humans and agents.

First, although the orchestrator proposes the workflow plan, the human should retain absolute agency to revise the plan directly and discuss alternatives. The workflow plan proposed by the orchestrator should not be treated as authoritative commands for either human workers or AI agents. Specifically, the operational, ethical, and legal boundaries within which the task must be executed should be defined by the human principal, not by the agent. In our setting, the learned delegation policy is therefore best understood as a decision-support mechanism for coordinating work, rather than as an autonomous authority over either the human user or downstream agents.

Second, learned delegation policies raise concerns about accountability in long delegation chains. When an orchestrator assigns subtasks to multiple downstream agents and incorporates their outputs into subsequent decisions, errors or harms may be difficult to attribute to any single component. We therefore view transparent records of delegation decisions, intermediate outputs, and human interventions as important for auditability and post-hoc review. We design the user interface used in our own user study to ensure such transparency.

Finally, we consider the risk of de-skilling. Although our experiments show that learned policies can assign meaningful steps to humans, including asking users to complete parts of the work and inserting additional breakpoints for human review or input, the use of AI agents may still reduce opportunities for users to exercise domain expertise, planning, or judgment over time. Future deployments should therefore consider user-configurable delegation preferences, for example by incorporating user preferences or user-specific profiles into the R_J rubric. Users should also receive appropriate training so that they understand the orchestrator's recommendations as fallible, objective-dependent, and subject to distribution shift, rather than as definitive instructions.

The user study in this paper was conducted with approval from our Institutional Review Board and with full informed consent from all participants.

REFERENCES

Marwa Abdulhai, Isadora White, Charlie Snell, Charles Sun, Joey Hong, Yuexiang Zhai, Kelvin Xu, and Sergey Levine. LMRL gym: Benchmarks for multi-turn reinforcement learning with language models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

Daron Acemoglu, David Autor, and Simon Johnson. Building pro-worker artificial intelligence. Technical report, National Bureau of Economic Research, 2026.

Lisanne Bainbridge. Ironies of automation. In *Analysis, design and evaluation of man–machine systems*, pages 129–135. Elsevier, 1983.

Léo Boisvert, Megh Thakkar, Maxime Gasse, Massimo Caccia, Thibault L De Chezelles, Quentin Cappart, Nicolas Chapados, Alexandre Lacoste, and Alexandre Drouin. Workarena++: Towards compositional planning and reasoning-based common knowledge work tasks. *Advances in Neural Information Processing Systems*, 37:5996–6051, 2024.

K. J. Kevin Feng, Kevin Pu, Matt Latzke, Tal August, Pao Siangliulue, Jonathan Bragg, Daniel S. Weld, Amy X. Zhang, and Joseph Chee Chang. Cocoa: Co-planning and co-execution with AI agents. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*, 2026.

Andrew Fuchs, Andrea Passarella, and Marco Conti. Optimizing delegation between human and AI collaborative agents. In *Towards Hybrid Human-Machine Learning and Decision Making (HLDM) workshop at ECML PKDD*, 2023.

Andrew Fuchs, Andrea Passarella, and Marco Conti. Optimizing delegation in collaborative human-AI hybrid teams. *ACM Transactions on Autonomous and Adaptive Systems*, 19(4):1–33, 2024. doi: 10.1145/3687130.

Andreas Fügener, Jörn Grahl, Alok Gupta, and Wolfgang Ketter. Cognitive challenges in human–artificial intelligence collaboration: Investigating the path toward productive delegation. *Information systems research*, 33(2):678–696, 2022.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. In *International Conference on Learning Representations*, volume 2026, pages 127924–127945, 2026.

Ziyang Guo, Yifan Wu, Jason D Hartline, and Jessica Hullman. A decision theoretic framework for measuring ai reliance. In *Proceedings of the 2024 ACM conference on fairness, accountability, and transparency*, pages 221–236, 2024.

Gaole He, Gianluca Demartini, and Ujwal Gadiraju. Plan-then-execute: An empirical study of user trust and team performance when using LLM agents as a daily assistant. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*, 2025. doi: 10.1145/3706598.3713218.

Patrick Hemmer, Monika Westphal, Max Schemmer, Sebastian Vetter, Michael Vössing, and Gerhard Satzger. Human-AI collaboration: The effect of AI delegation on human task performance and task satisfaction. In *Proceedings of the 28th International Conference on Intelligent User Interfaces (IUI)*, 2023.

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. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Thomas Kwa, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx, et al. Measuring ai ability to complete long software tasks. *Advances in Neural Information Processing Systems*, 38:92213–92266, 2025.

Philippe Laban, Tobias Schnabel, and Jennifer Neville. Llms corrupt your documents when you delegate. *arXiv preprint arXiv:2604.15597*, 2026.

Vivian Lai, Samuel Carton, Rajat Bhatnagar, Q. Vera Liao, Yunfeng Zhang, and Chenhao Tan. Human-AI collaboration via conditional delegation: A case study of content moderation. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, 2022. doi: 10.1145/3491102.3501999.

Yuetai Li, Yichen Feng, Zhangchen Xu, Zixian Ma, Kaiyuan Zheng, Fengqing Jiang, Xinghua Sun, Rulin Shao, Zichen Chen, Yue Huang, et al. Jobbench: Aligning agent work with human will. *arXiv preprint arXiv:2605.26329*, 2026.

Q Vera Liao, Daniel Gruen, and Sarah Miller. Questioning the ai: informing design practices for explainable ai user experiences. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, pages 1–15, 2020.

Charlie Masters, Advaith Vellanki, Jiangbo Shangguan, Bart Kultys, Jonathan Gilmore, Alastair Moore, and Stefano Albrecht. Orchestrating human-ai teams: The manager agent as aunifying research challenge. In *Proceedings of the 2025 7th International Conference on Distributed Artificial Intelligence*, pages 91–107, 2025.

Hussein Mozannar, Gagan Bansal, Cheng Tan, Adam Fourney, Victor Dibia, Jingya Chen, Jack Gerrits, Tyler Payne, Matheus Kunzler Maldaner, Madeleine Grunde-McLaughlin, et al. Magentic-UI: Towards human-in-the-loop agentic systems. *arXiv preprint arXiv:2507.22358*, 2025.

National Center for O*NET Development. O*NET OnLine help: The database. O*NET OnLine, 2026. URL https://www.onetonline.org/help/onet/database. Accessed: March 22, 2026.

Vishakh Padmakumar, Lujain Ibrahim, Zora Zhiruo Wang, Jennifer Wang, Q Vera Liao, and Diyi Yang. Offloading score: Measuring ai reliance through counterfactual workflows. *arXiv preprint arXiv:2605.29392*, 2026.

Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simón Posada Fishman, Marwan Aljubeh, Phoebe Thacker, Laurance Fauconnet, et al. GDPVal: Evaluating AI model performance on real-world economically valuable tasks. *arXiv preprint arXiv:2510.04374*, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Lior Shani, Aviv Rosenberg, Asaf Cassel, Oran Lang, Daniele Calandriello, Avital Zipori, Hila Noga, Orgad Keller, Bilal Piot, Idan Szpektor, Avinatan Hassidim, Yossi Matias, and Rémi Munos. Multi-turn reinforcement learning with preference human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Yijia Shao, Vinay Samuel, Yucheng Jiang, John Yang, and Diyi Yang. Collaborative gym: A framework for enabling and evaluating human-agent collaboration. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=GDYueXtKXT.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. Introduces Group Relative Policy Optimization (GRPO).

Wentao Shi, Mengqi Yuan, Junkang Wu, Qifan Wang, and Fuli Feng. Direct multi-turn preference optimization for language agents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

Hongjin Su, Shizhe Diao, Ximing Lu, Mingjie Liu, Jiacheng Xu, Xin Dong, Yonggan Fu, Peter Belcak, Hanrong Ye, Hongxu Yin, et al. ToolOrchestra: Elevating intelligence via efficient model and tool orchestration. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026.

Nenad Tomašev, Matija Franklin, and Simon Osindero. Intelligent AI delegation. *arXiv preprint arXiv:2602.11865*, 2026.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Monica Lam, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025a.

Zora Zhiruo Wang, Yijia Shao, Omar Shaikh, Daniel Fried, Graham Neubig, and Diyi Yang. How do AI agents do human work? comparing AI and human workflows across diverse occupations. *arXiv preprint arXiv:2510.22780*, 2025b.

Zora Zhiruo Wang, Sanidhya Vijayvargiya, Aspen Chen, Hanmo Zhang, Venu Arvind Arangarajan, Jett Chen, Valerie Chen, Diyi Yang, Daniel Fried, and Graham Neubig. How well does agent development reflect real-world work? *arXiv preprint arXiv:2603.01203*, 2026.

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. In *First Conference on Language Modeling (COLM)*, 2024.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

A Limitations and Future Work

Evaluation design. In this work, our main experiments use simulated users to enable controlled, scalable comparisons across benchmark tasks. While we introduce several design choices to better approximate constraints on human involvement such as (i) including underspecified tasks where only the simulated user has access to the missing information and (ii) imposing shorter execution timeouts on human-assigned steps, simulation cannot fully capture how people perceive and participate in AI-orchestrated collaboration. Our user study provides initial evidence with real users, but is limited in scale and interaction duration. We hope to conduct future work to evaluate learned orchestration through longer-term field deployments with diverse users and tasks, particularly to understand effects on collaboration legibility and human agency, which are critical for AI systems but are inherently difficult to assess through simulation alone.

Capturing the nuances of human-agent teaming. CollabOrchestra optimizes workflow plans along a specific set of objectives: handoff quality, workflow coherence, granularity appropriateness, user involvement appropriateness, and execution feasibility grounded in agents' observed capabilities. We evaluate the resulting collaboration through delivery rate, task performance, user preference, legibility, and perceived agency. These objectives, however, do not capture all considerations that should shape human-agent collaboration. In safety-critical or high-stakes settings, for example, a step may be technically feasible for an agent yet still warrant human review because of its potential consequences. Future work could incorporate additional considerations such as risk, uncertainty, accountability, and task criticality into both orchestration objectives and worker profiles.

Better modeling the human worker. Unlike agent workers, whose capabilities can be grounded through execution-based profiling, profiling human capabilities through deployment is substantially more difficult. We therefore represent human capabilities and preferences through a textual user profile rather than modeling them directly from observed behavior. While practical, this abstraction may not fully capture what users can reliably contribute or how their capabilities and preferences vary across tasks and evolve during collaboration. Moreover, eliciting such profiles from real users is challenging. In our user study, we fix the profile to a user with moderate desired involvement as our pilot studies suggested that direct profile elicitation was difficult and that users tended to select the balanced option when presented with predefined profiles. Future work should explore inferring and continually updating user profiles from interaction rather than relying on upfront specification. A deeper challenge is balancing user preferences with the involvement needed to maintain legibility and control. A user may prefer faster completion while meaningful checkpoints are still valuable for understanding the workflow and intervening when necessary. Learning how to negotiate this trade-off rather than optimizing purely for short-term user satisfaction is an important direction for human-agent teaming.

B Prompts Used in CollabOrchestra

B.1 Orchestrator Prompt

As formulated in §2.2, at each step, the Orchestrator observes o_t which includes the team configuration $\mathcal{W}$, the current workflow plan $\mathcal{P}_t$, the chat history $\mathcal{H}_t^{\text{chat}}$, the Orchestrator's own action history $\mathcal{H}^{\text{act}}$, and the triggering even e_t, and will be prompted to choose an action in its action space.

Orchestrator Prompt

A conversation between a User and an Orchestrator. The User starts the conversation with a task, and the Orchestrator coordinates a team of workers — specialized agents and the User themself — to complete it. You are the Orchestrator. You do not execute the task yourself: you maintain the workflow plan, delegate each step to the worker best suited to it, and keep the User informed and involved. The User and the Orchestrator can continue the conversation for multiple turns to better solve the problem. When you choose your next action, not only consider the current turn but also potential future turns.

#Actions
##Guidelines
1. You can call one action or multiple actions in a single response.
2. For each action, call it in the following JSON format:

```
{
    "action": "<action_name>",
    "params": "<action_parameters_as_json_object>"
}
```

##Available Actions

```
{
    "name": "delegate",
    "description": "Delegate a previously planned workflow step to
    a worker. \"worker_name\" is the name of the agent (or the human
    user) to delegate to among the available workers. \"uuid\" is the
    step ID from the current workflow pool.",
    "params": {
      "worker_name": "string",
      "uuid": "string"
    }
}
{
    "name": "message",
    "description": "Message the user to elicit additional information
    or confirmation, reply to the user message, or to give suggestion.
    \"tid\" is the thread ID to which the message belongs; do not
    provide it if the message will be sent in the main conversation
    thread. \"message\" is the content to be sent.",
    "params": {
      "tid?": "string",
      "message": "string"
    }
}
{
    "name": "update_workflow_plan",
    "description": "Update the workflow plan with new steps. The whole
    plan will be replaced. \"steps\" is the list of new workflow steps
    to replace the existing plan. Each step includes \"goal\" which is
    the objective of the step, \"deliverables\" which can can either be
    specific files or a text output that satisfies certain criteria,
    and \"assignee\" which should be the name of the agent (or the
    human user) to delegate to among the available workers.",
    "params": {
      "steps": [
        {
          "goal": "string",
          "deliverables": [
            "string"
          ],
          "assignee": "string"
        }
      ]
    }
  }
```

Workflow Management

Guidelines

1. You are collaborating with the user at the workflow level. This means you should delegate individual steps to specialized agents or the user as needed. For workflow step that is about discussion, do it yourself instead of delegating.
2. When updating the workflow plan, provide a list of steps with clear content and expected deliverables.
3. Always keep the user informed about the workflow plan and any changes made to it. Create the workflow plan using the "update_workflow_plan" action first before delegating any steps.
4. You can delegate workflow steps to available workers including the human user based on their capabilities and descriptions. If the user wants to do something by themselves, delegate suitable steps to the user.

Available Workers

{team configuration}

Current Workflow Pool

{current workflow plan}

Chat
Guidelines
1. There is a main thread of conversation and threads for each workflow step if needed.
2. Your responses should primarily focus on the main thread unless specifically addressing a workflow step.
Current Chat History
{chat history with the form of "[timestamp, id]: message"}

Action History
{the Ochestrator's action history with the form of "[timestamp]: message"}

Latest Event
{latest event description}

Output Format
Now give your output in the following JSON format:

```
{
    "reasoning": "brief_reasoning_about_your_next_steps",
    "actions": [
        {
            "action": "<action_name>",
            "params": {<action_parameters_as_json_object>}
        },
    ]
}
```

B.2 Prompts Used in Agent Worker Profiling

To obtain agent worker profiling results, we use the following prompts to synthesize (g, A_i, d) tuples from training task descriptions $\mathcal{D}$ and to evaluate whether the deliverable d is produced after agent worker A_i executes goal g in the sandbox. Specifically, the "Task Rewriting Prompt" and "Spec Splitting Prompt" standardize tasks from $\mathcal{D}$ into a unified format containing the detailed task description, user-specific information, and expected deliverables. The "Reference File Generation Prompt" prepares the sandbox by creating concrete reference files required for execution. The "Workflow Decomposition Prompt" and "Upstream Context Prompt" decompose each task into workflow steps and provide upstream execution summaries, enabling non-initial steps to be executed independently. Finally, the "LLM Judge Prompt" assesses whether the expected deliverable for each workflow step is produced after the agent worker executes the step in the sandbox.

Task Rewriting Prompt

Rewrite the following task description into the same style as the example.
Style rules.

- First person, as the user speaking to an assistant.
- Match the length and depth of the example: produce at least 1500 characters. Expand with business context, motivation, constraints, and expected deliverables wherever they are implied; do not invent numbers or names that contradict the original.
- One long paragraph, or a short preamble followed by a numbered action list when the task has explicit steps.
- Include concrete business context (who the user is, what organization they belong to, why the task matters) when natural.
- Natural and conversational; the reader should believe a real person sent this message to an assistant.
- Preserve every concrete detail (numbers, filenames, dates, locations) from the original.

Example. `{style_example}` **Task description to rewrite.** `{raw_task}`
Output only the rewritten description, with no preamble, quotes, or markdown.

Spec Splitting Prompt

Given a self-contained task description, split it into three fields.

1. `prompt`: what the user would actually say to the assistant first, natural and conversational, omitting details a realistic user would forget or assume the assistant can infer.
2. `hidden_information`: the specific details the user left out but that are required to complete the task, such as numbers, file paths, thresholds, and exact field names. The assistant should ideally ask a clarifying question to recover these.
3. `expected_deliverables`: two to five concrete outputs the completed task should produce, such as files, reports, or decisions.

Full task description. `{full_task}`
Respond with a JSON object of exactly this shape:

```
{
  "prompt": "...",
  "hidden_information": "...",
  "expected_deliverables": ["...", "..."]
}
```

Reference File Generation Prompt

You are generating a reference file for a task profiling dataset, given the task context (`{task_context}`) and the desired file spec (`{file_spec}`).
Requirements.

- Work in the current directory (`{workdir}`); do not touch files outside it.
- Produce the final artifact directly, named exactly `{artifact_name}`.
- A small helper script may be used as an intermediate step, but it must be run to produce the final artifact, the artifact must exist on disk before finishing, and the helper script must not be the only output.
- Verify the final artifact before finishing, for example by opening a CSV and counting columns and rows, or opening a YAML and confirming the schema.
- Do not embed mock user credentials or personally identifiable information.
- When done, print a one-line summary beginning with `ARTIFACT_READY:` followed by the final filename.

Workflow Decomposition Prompt

You are decomposing a user task into a workflow plan. The inputs are the available workers (`{workers_info}`), the fully specified task description (`{task_prompt}`), the available reference files (`{reference_files}`) and websites (`{reference_websites}`), and the enumerated task-level deliverables the user ultimately wants (`{expected_deliverables_enumerated}`).
Task. Produce a workflow plan with three to six steps. Each step must be concrete enough that the assigned worker can execute it without guessing, and small enough that its success or failure is checkable.
Deliverable allocation rule.

- Every task-level deliverable must be covered by exactly one step, through `covers_task_deliverables`, a list of integer indices into the enumerated list.
- Steps may additionally declare internal intermediate outputs in `step_deliverables`; task-level deliverables must not be duplicated there.

Respond with a JSON array whose elements take the form:

```
{
  "goal": "<imperative sentence>",
  "assignee": "<exact worker name from the list above>",
  "covers_task_deliverables": [<indices into the task-level list>],
  "step_deliverables": ["<internal intermediate output>", ...],
  "requires_outputs_from": [<zero-indexed indices of previous steps
this one depends on>]
}
```

Upstream Context Prompt

Describe what a downstream worker should assume about a completed upstream workflow step, so the next step has the context it needs. The inputs are the task context (`{task_prompt}`), the reference files already on disk (`{reference_files}`), and the step just completed, given by its goal (`{step_goal}`), assignee (`{assignee}`), and expected deliverables (`{deliverables}`).
What to write.

- Describe the conclusions reached: key findings, specific values, schemas, and decisions made.
- If the deliverable is intangible information such as a count, a comparison, or a schema, state it concretely.
- If the deliverable is a new file that is not among the reference files above, do not claim it exists on disk; the downstream worker will need to recompute it. Phrase it as "The upstream analysis found . . . " rather than "The file now exists in the sandbox."

Length. Four to ten lines, concrete enough (column names, example values, dates) for a downstream agent to reason about, and not a mere restatement of the goal. Output the description directly, with no JSON or preamble.

LLM Judge Prompt

You are an independent judge evaluating whether an AI agent successfully completed a delegated workflow step. You are given the step goal (`{goal}`), the expected deliverables (`{deliverables}`), the agent name (`{agent_name}`), the agent's final output (`{final_output}`), the files it produced (`{files_listing}`), and previews of those files (`{file_previews}`).
Decision rules.

- Mark PASS only if the agent produced the expected deliverables and the final output is consistent with them.
- A claim of success without the corresponding files is a FAIL.
- Wrong, partial, or missing required content is a FAIL.
- Minor formatting differences are acceptable as long as the substantive deliverable is correct.

Respond with a single-line JSON object and no surrounding text:

```
{"label": 0 or 1, "judgment": "<one short sentence explaining the decision>"}
```

B.3 User Simulator Prompt in Multi-turn RL Training

In CollabOrchestra, we train the Orchestrator policy through multi-turn interactions with the simulated user. The simulated user is conditioned on the user profile that specifies factors like expertise, desired level of involvement, learning preference, *etc.*, as well as the additional information that is only available to the user and not revealed to the Orchestrator in the initial prompt. This information asymmetry encourages the Orchestrator to elicit missing information through interaction and adapt its workflow plan accordingly.

User Simulator Prompt

Task Description
I am a user, and you are a helpful assistant. In this task, I will show you a conversation between a human client with a bot, and you will need to predict what the human client would say next. In order to predict the client's response, you will also be given the client's profile and some prediction rules.

Note that this task is quite different from acting like a helpful assistant, or doing what a helpful assistant would do. For example:

- Most clients are quite terse and convey minimal information so that they can get things done quickly. There might be exceptions, but clients are different from assistants who tend to provide as much information as possible.
- When most clients ask about a question, they typically don't know the answer themselves. They may hold false beliefs, and you need to predict they might say based on their false beliefs.
- Most clients don't know how to ask the assistant to make changes. They seldom question the assistant's answer, like saying "I don't like that" or "That's not what I meant".

- The clients may choose to end the conversation. This is also quite different from assistants, who tend to keep the conversation going. If you predict that the client will end the conversation, output 'CLOSE', and say nothing else.

Depending on the **Client Persona** (which we will provide later), the client's behavior is sometimes different from a helpful assistant. Make sure to follow the Client Persona section later.

Conversation History
{conversation history}

Client Persona
{simulated user profile and additional information that is only available to the user}

Output Instructions

- Generate ONLY the client's next message OR the word "CLOSE"
- Output "CLOSE" if the client would end the conversation without responding
- Do not include both a message and "CLOSE"
- Do not add any explanations or additional text.

B.4 Workflow Plan Quality Grader Prompt

We evaluate the *Plan Quality* of a workflow plan using an LLM-as-a-judge. Following the implicit aggregation approach of Gunjal et al. (2026), we provide the judge with all rubric criteria and their categorical weights, and let the judge jointly aggregate them into a single scalar score. We develop and validate the rubric in three stages:

- **Defining rubric dimensions.** A high-quality workflow should decompose the task into meaningful units of work, connect steps through appropriate handoffs, and collectively accomplish the original task goal. For human-agent collaboration, it should additionally involve the user where human participation is valuable as well as for maintaining awareness and control over the workflow. We operationalize these desiderata with four dimensions: *handoff quality*, *workflow coherence*, *granularity appropriateness*, and *user involvement appropriateness*.
- **Calibrating the rubric against human annotations.** We first manually define grading criteria for each dimension. The authors then annotate 20 pairwise comparisons between workflow plans generated for the same task and worker set $(T, \mathcal{W})$. We use `claude-sonnet-4` to iteratively refine the rubric until its judgments agree with all 20 human-annotated comparisons.
- **Validating across judge models.** We sample 200 workflow plans from rollouts of different Orchestrator policies on GDPval and JobBench and independently grade each plan using `Qwen3-235B-A22B-Instruct`, `gemini-3-flash`, and `claude-sonnet-4.6`. The judges show consistent positive correlations:
  - Qwen–Gemini: Pearson $r = 0.640$ ($p < 1e-23$), Spearman $\rho = 0.646$ ($p < 1e-24$);
  - Qwen–Claude: Pearson $r = 0.632$ ($p =< 1e-22$), Spearman $\rho = 0.616$ ($p < 1e-21$);
  - Gemini–Claude: Pearson $r = 0.498$ ($p < 1e-13$), Spearman $\rho = 0.484$ ($p < 1e-12$).

The grader system prompt and full rubric are provided below.

Workflow Plan Quality Grader Prompt

You are an expert evaluator of workflow plans for human-agent collaborative systems. Given an original task request, a generated workflow plan, and a list of quality rubrics, please rate the overall quality of the workflow plan on a scale of 1 to 10 based on how well it satisfies the rubrics.

Consider all rubrics holistically when determining your score. A workflow that violates multiple rubrics should receive a lower score, while a workflow that excels across all rubrics should receive a higher score. Use the rubric weights to guide the relative importance of each dimension in your overall assessment.

Provide detailed reasoning that demonstrates your consideration of each rubric before assigning your final score. Start your response with a valid JSON object enclosed in “‘json and “‘. The JSON object should contain two keys:

- "rating": an integer between 1 and 10
- "reasoning": a string containing your detailed evaluation

Example response format:

```
{
  "rating": 7,
  "reasoning": "Your detailed analysis here..."
}
```

Given the following task request, workflow plan, and rubrics, please rate the overall quality of the workflow plan on a scale of 1 to 10 based on how well it satisfies the rubrics.

<task_request>
{original_task_description}
</task_request>

<workflow_plan>
{workflow_plan_json}
</workflow_plan>

<worker_profiles>
{worker_profiles_json}
</worker_profiles>

<rubrics>
{rubric_list_string}
</rubrics>

Your JSON Evaluation:

Workflow Plan Quality Rubric

1. **User Involvement Appropriateness** (Weight: HIGH)
   - Definition: How well the workflow assigns meaningful tasks to the human user that align with their skills, goals, preferences, and contexts requiring human oversight (auditing, ethical decisions, stakeholder communication), while delegating laborious, repetitive, or automatable work to agents.
   - Evaluation Criteria:
     - Are critical oversight points (auditing, verification, ethical decisions, stakeholder communication) assigned to humans?
     - Do human tasks align with their skills, goals, and preferences?
     - Are repetitive, automatable, or laborious tasks (execept for parts that the human wants to learn or be involved in) delegated to agents?
   - **CRITICAL RULE**: If any work assigned to the human violates the human preferences OR if ANY critical ethical/strategic decision lacks human oversight, the score CANNOT exceed 5.
2. **Handsoff Quality** (Weight: MEDIUM)
   - Definition: How clearly each step defines specific, measurable outputs with concrete formats and completion criteria that enable the next worker to proceed immediately without ambiguity or audit the previous steps.
   - Evaluation Criteria:
     - **For agent workers**: Does each deliverable specify a concrete file with an exact file name and format (e.g., "report.csv with columns X, Y, Z" not just "CSV file" or "analysis")?
     - **For human workers**: Deliverables can be either concrete files OR text outputs with specific criteria (e.g., "approval confirmation" or "feedback on design").
     - Are there clear completion criteria that allow verification?
     - Can the next step proceed immediately with the deliverable without asking for clarification?

- Are there vague deliverables without specific file names or concrete criteria?
- **CRITICAL RULE**: If any agent worker deliverable lacks a concrete file name OR if ANY deliverable would definitely require follow-up clarification to proceed, the score CANNOT exceed 5.

3. **Workflow Coherence** (Weight: HIGH)
   - Definition: How logically the workflow steps connect and build upon each other to form a complete, efficient path from the initial request to the final goal without gaps, redundancies, or unnecessary detours.
   - Evaluation Criteria:
     - Does each step logically follow from and build upon previous steps?
     - Are there missing steps needed to achieve the goal?
     - Are there duplicate efforts or unnecessary repeated work?
     - Are there steps that don't contribute to the final goal?
     - Could the goal be achieved more efficiently with fewer steps?
     - Does the final step actually achieve the stated goal?
   - **CRITICAL RULE**: If there is ANY critical missing step that prevents goal achievement OR if the final step does NOT produce the requested outcome OR if there are 2+ redundant/unnecessary steps, the score CANNOT exceed 5.
4. **Granularity Appropriateness** (Weight: MEDIUM)
   - Definition: How well the workflow is decomposed into steps that are neither too coarse (multiple unrelated sub-goals packed together) nor too fine (trivial micro-tasks), creating natural units of work that are easy to check.
   - Evaluation Criteria:
     - Are there steps that pack multiple unrelated sub-goals together?
     - Are there trivial micro-tasks that could be combined?
     - Does each step represent a coherent unit of work that can be completed, checked, and handed off cleanly?
     - Is each step substantial enough to be worth verifying but focused enough to verify efficiently?
     - Does the granularity minimize unnecessary context switching between workers?
   - **CRITICAL RULE**: If there are 2+ steps that pack unrelated sub-goals together OR if there are 2+ trivial micro-tasks creating unnecessary handoffs, the score CANNOT exceed 5.

C COLLABORCHESTRA TRAINING ALGORITHM

Algorithm 1 CollabOrchestra Training

Input: worker set $\mathcal{W} = \{U, A_1, ..., A_N\}$, training corpus $\mathcal{D}$, base policy π_θ.

// Stage 1: Agent Worker Profiling
$\mathcal{L} = \{\}$
for each task $t \in \mathcal{D}$ **do**
  Synthesize fully-specified task t' and decompose into workflow steps $\{(g_i, w_i, d_i)\}$
  for each agent-assigned step (g_i, w_i, d_i) **do**
    $o_i = w_i.\texttt{execute}(g_i, d_i)$ ▷ Run agent with upstream context in sandbox
    $v_i = \texttt{judge}(g_i, d_i, o_i) \in \{0, 1\}$ ▷ Pass/fail from LLM judge
    $\mathcal{L} = \mathcal{L} \cup \{(g_i, w_i, d_i, v_i)\}$
  end for
end for
$f_\theta \leftarrow \texttt{train_predictor}(\mathcal{L})$ ▷ Fit feasibility predictor

// Stage 2: Orchestrator Policy Training
for each sample $(T, \mathcal{W}) \in \mathcal{D}$ **do**
  $\mathcal{H} = \{\}$, $P =$ null ▷ Initialize dialogue history and plan
  repeat ▷ Multi-turn rollout against simulated user
    $a \sim \pi_\theta(\cdot \mid T, \mathcal{W}, \mathcal{H})$ ▷ Sample orchestrator action
    if $a = \texttt{UpdateWorkflowPlan}(P')$ **then**
      $P = P'$
    else if $a = \texttt{SendMessage}(m)$ **then**
      $\mathcal{H} = \mathcal{H} \cup \{(m,\ \mathcal{U}(m, T))\}$ ▷ Simulated user responds to clarification
    end if
  until $\mathcal{U}$ sends `Finish` or step limit reached
  $R_J = \texttt{judge}(P)$ ▷ LLM judge of the plan quality
  if $|\{i : w_i \in \mathcal{A}\}| = 0$ **then**
    $R_f = 0$ ▷ Prevent reward gaming by full user offloading
  else
    $p_i = f_\theta(g_i, w_i, d_i)$ for each agent-assigned step
    $R_f = 0.5 \cdot \min_i p_i + 0.5 \cdot \frac{1}{n} \sum_i p_i$ ▷ Blend min and mean feasibility
  end if
  $\theta = \texttt{GRPO_update}(\theta,\ R_J + \alpha \cdot R_f)$ ▷ Update policy with composite reward
end for
return π_θ

D COLLABORCHESTRA TRAINING DETAILS

We construct the training set from publicly accessible job descriptions on a remote-work platform[5]. Using `claude-sonnet-4`, we rewrite each description into two self-contained task queries: one fully specified and one underspecified to reflect tasks where the Orchestrator must elicit missing information from the user. This yields 16,534 training instances in total, forming our training set $\mathcal{D}$.

For feasibility predictor training, we cap sandboxed agent profiling at 2,000 executed steps across GeminiCLIAgent and BrowserUseAgent due to execution cost. We encode each (g, w, d) tuple using gte-large-en-v1.5[6] and pass the resulting representation through a two-layer MLP with 128 hidden units to predict feasibility. We train f_φ for 20 epochs with a learning rate of 2×10^{-5}, weight decay of 0.01, batch size of 32, and a cosine learning-rate schedule with two warmup epochs.

[5]We manually collected 8,267 job descriptions from `https://www.upwork.com` in accordance with Upwork's Terms of Use.

[6]`https://huggingface.co/Alibaba-NLP/gte-large-en-v1.5`

For Orchestrator training, we use a frozen `Qwen3-235B-A22B-Instruct` both as the plan-quality judge for R_j and as the user simulator. We set $\alpha = 0.5$ in the composite reward $R(\mathcal{P})$. We train each of the three Orchestrator base models for one epoch with a batch size of 128, a GRPO group size of 8, and a learning rate of 3×10^{-5}. All Orchestrator training runs use Tinker,[7] an API provided by Thinking Machines Lab.

E Validating Feasibility Predictor Design

In CollabOrchestra, we parameterize the feasibility predictor as an encoder with an MLP head, trained with binary cross-entropy on execution-success labels. To validate this design, we conduct a controlled experiment comparing it against alternative LLM-based implementations. We use agent profiling records released by Wang et al. (2026)[8], which contain agent-executed workflow steps annotated with success or failure for six agent frameworks, namely SWE-agent, OpenHands, ReAct, MetaGen, WebVoyager, and IBM CUGA, across 12 benchmarks. After filtering invalid or unlabeled records and removing status markers from step descriptions to prevent label leakage, the resulting dataset contains 19,173 labeled steps (13,484 successes and 5,689 failures). We evaluate using 5-fold cross-validation with workflow-disjoint splits.

Model	Method	AUROC↑	Brier↓
LLM-based success predictor			
Qwen3-4B	Zero-shot	0.59	0.31
Qwen3-4B	LoRA GRPO	0.59	0.26
Encoder-based success predictor			
gte-large	MLP Fine-tune	**0.74**	**0.24**

Table 4: **Comparison of success-predictor implementations.** We report AUROC and Brier score under 5-fold cross validation on the profiling dataset.

We compare our encoder-based approach with two LLM alternatives: (1) **Qwen3-4B Zero-shot**, which directly predicts a success probability from the agent profile and step description; and (2) **Qwen3-4B LoRA GRPO**, which is post-trained with LoRA and GRPO on the same prediction task using execution labels as reward. As shown in Table 4, the encoder-based predictor provides substantially stronger discrimination and calibration: fine-tuned gte-large achieves an AUROC of 0.74 and a Brier score of 0.24, compared with 0.59/0.31 for zero-shot Qwen3-4B and 0.59/0.26 after GRPO. These results support our choice of a lightweight encoder-based model for estimating agent-step feasibility. Notably, a single predictor is trained across multiple agents, suggesting that it can generalize agent-step feasibility across heterogeneous workers rather than requiring an agent-specific model. Moreover, training the predictor offline allows feasibility to be estimated without executing every candidate workflow during RL rollouts, substantially reducing the cost of Orchestrator training.

F Simulating Human Execution

As described in §4.1, we complement planning-stage evaluation with end-to-end workflow execution to measure the collaboration outcomes induced by each plan. Agent-assigned steps are dispatched to the corresponding worker (GeminiCLIAgent or BrowserUseAgent), while human-assigned steps are carried out by a simulated user.

To model users who can both communicate with the Orchestrator and directly contribute to task execution, we implement the simulated user as an agent rather than a chat-only model. Specifically, we run GeminiCLIAgent powered by `gemini-3-flash` in autonomous mode within the rollout workspace, allowing it to inspect reference files and outputs from upstream steps and to create or modify artifacts. For each assigned step, the simulator is conditioned on (1) the user profile associated with the data point (*i.e.*, the balanced user profile in our evaluation); (2) user-only information for underspecified data points; (3) the original task prompt, (4) the full workflow plan, (5) the current step's goal and deliverables, and (6) outcomes from previous steps. We impose a three-minute timeout on each simulated user step to approximate bounded user effort.

[7] `https://thinkingmachines.ai/tinker`

[8] `https://github.com/zorazrw/ai4work-resources`

What this simulation does and does not measure. The simulation evaluates a cooperative setting in which the user attempts each step explicitly assigned to them, without guaranteeing success. Importantly, the simulated user contributes only when requested by the Orchestrator and does not proactively compensate for work assigned to an agent. Execution outcomes therefore reflect the Orchestrator's division of work rather than intervention from an unconstrained user simulator.

As shown in Table 2, Rubric Prompting yields lower Delivery Rate and Task Performance, particularly on underspecified tasks. Inspection of its workflows suggests that it less consistently involves the user in providing missing information, shaping intermediate outputs, or reviewing critical deliverables. We also observe smaller gains over Best Solo Agent on JobBench than on GDPval. Inspection of the execution trajectories shows that JobBench contains harder tasks with steps that are challenging for both agent workers and the simulated user. Overall, simulated execution provides a scalable way to study how orchestration decisions translate into end-to-end collaboration outcomes. However, it does not capture the full behavior or experience of real users. In particular, the simulator does not decline assigned work and cannot reliably measure collaboration legibility or human agency. We therefore complement the simulated evaluation with a real-user study that examines these aspects of orchestrated human–agent collaboration.

Prompt for Simulating Human Execution

You are simulating a real user collaborating with an AI assistant.

Your user profile
{user_profile}

Hidden information you (and only you) know
{hidden_information}

The original task you asked the assistant to solve
{task_prompt}

The full workflow plan the assistant proposed
{plan_json}

Your current step in the plan
Goal: {goal}
Deliverables: {deliverables}

Prior conversation / outputs
{upstream_context}

Shared workspace (your current working directory)
The current working directory is a shared workspace that persists across every step of this workflow. It has three top-level folders:

- reference_files/ — read-only inputs supplied with the task. Do not modify or delete anything here.
- handover_files/ — scratch space for work-in-progress that flows between steps. Read whatever previous steps left here, and write anything the next step will need to consume.
- deliverables/ — the FINAL artifacts the task expects

Your task
Complete this step as a user would do. If the assistant asks you a question, answer it based on your user profile and hidden information. If the assistant asks you to perform an action, complete it based on your user profile and hidden information — when that means producing or editing files, write them into the appropriate folder (handover_files/ for intermediate work, deliverables/ only on the final step). If the assistant asks you to review something, read the relevant files from reference_files/ or handover_files/ and be critical based on the hidden information.

G Coding Human Steps in Workflow Plans

As described in §5.2, we analyze orchestrated human-agent collaboration by examining the steps assigned to the human in the final workflow.We deterministically classify each human-assigned step

Role	Verb stems
review/approve	`review*, approv*, sign-off, accept*, verif*, valid*, audit*, check*, proofread*, inspect*, feedback, qa, confirm*, agree*, align*, acknowledg*, discuss*`
supply input	`clarif*, specif*, defin*, decid*, choos*, select*, requirement*, scope, preference*, upload*, obtain*, provid*, suppl*, shar*, send*, attach*, collect*, gather*, furnish*, receiv*, get`
do the work	`hand-off, hand-over, handoff, close-out, closeout, clos*, wrap-up, notif*, archiv*, distribut*, disseminat*, publish*, deliver*, present*, submit*, draft*, writ*, creat*, build*, design*, produc*, record*, shoot*, edit*, analyz*, analys*, comput*, calculat*, screen*, shortlist*, research*, prepar*, compil*, develop*, implement*, updat*, populat*, fill*, perform*, conduct*, execut*, assembl*, test*, train*, identif*, determin*, iterate`

Table 5: **Verb lexicon for the three human roles.** The leading clause of the goal g_i is matched against the lexicons.

based on the leading verb of its goal g_i into three roles: "*review/approve*", "*supply input*", "*do the work*". Table 5 provides the verb lexicon used for this classification.

Although *Review/approve*accounts for most human-assigned steps, CollabOrchestra assigns humans a broader range of roles than Direct Prompting and the ablated variants. In particular, CollabOrchestra sometimes delegates work to the human when their contribution can guide or complement subsequent agent execution or when the stake of the final deliverable is high. Representative examples include:

- **Human handles stakeholder-facing delivery.** *"Deliver the final proposal to the CPO and outline next-step recommendations"*, the last step of a partnership-proposal plan, with deliverable *"Suggested next-step checklist for the CPO"*.
- **Human presents the agent-produced deliverable.** The agent prepares *"a brief hand-over package and instructions for presenting to the client,"* including the final files and a presenter's guide, leaving the user responsible for the client-facing presentation.
- **Human provides creative direction, agent drafts.** The human writes *"a detailed scene-by-scene outline based on the story breakdown and character descriptions"* (`scene_outline.md`); the agent then *"writes the first draft of the screenplay following the outline"*).
- **Human specifies domain structure, agent implements** The human *"designs the structure of the stock and sales flow tables (layout, rows, columns, side-by-side format) and defines all necessary formulas"*; the agent then *"implements the Excel workbook: creates worksheets, populates tables with formulas"*.
- **Human authors the content, agent produces the artifact.** The human *"drafts the one-page Product Summary content"* (`draft_summary.md`); the agent then *"creates the PDF layout and generates the final PDF"*.

H HUMAN STUDY DETAILS

Participants were recruited through Upwork and social media, with a focus on non-technical users. In total, 24 participants completed the study, representing diverse occupations including lawyers, urban planners, and students outside computer science. Each participant brought one to three tasks of their own choosing and was compensated $20.00 per task. The study was approved by our Institutional Review Board, and all collected data are reported in anonymized or aggregate form.

Study Interface Because CollabOrchestra orchestrates collaboration at the workflow level, we design a desktop interface that makes the evolving division of work visible and easy to navigate. Unlike conventional agent interfaces that primarily organize interaction as a single conversation, our interface keeps the workflow plan in a persistent panel, allowing users to quickly understand what has been assigned, to whom, and what remains to be completed. During execution, the user-Orchestrator interaction serves as the main thread, while each delegated workflow step is organized as a subthread. This hierarchical design separates workflow-level coordination from step-level execution while preserving their relationship, helping users maintain awareness of both the overall plan and the progress of individual workers. Figure 4 shows the study interface. We further illustrates examples of human-agent collaboration trajectories collected in the user study in Figure 5 and Figure 6.

Statistical Details We test the 1-5 scale ratings against the neutral midpoint of 3. The ten-item composite has a mean of 3.78 ($SD = 0.67$, 95% CI $[3.60, 3.96]$), significantly above the midpoint according to one-sample t-test across the 55 evaluated tasks ($p < .001$). Each of the ten individual items also remains above the midpoint after Holm correction for multiple comparisons (all $p < .001$). Across the 550 item-level ratings, 71% are "agree" or "strongly agree," compared with 10% that are "disagree" or "strongly disagree." Among the 55 evaluated tasks, 45 have a ten-item mean above the neutral midpoint.

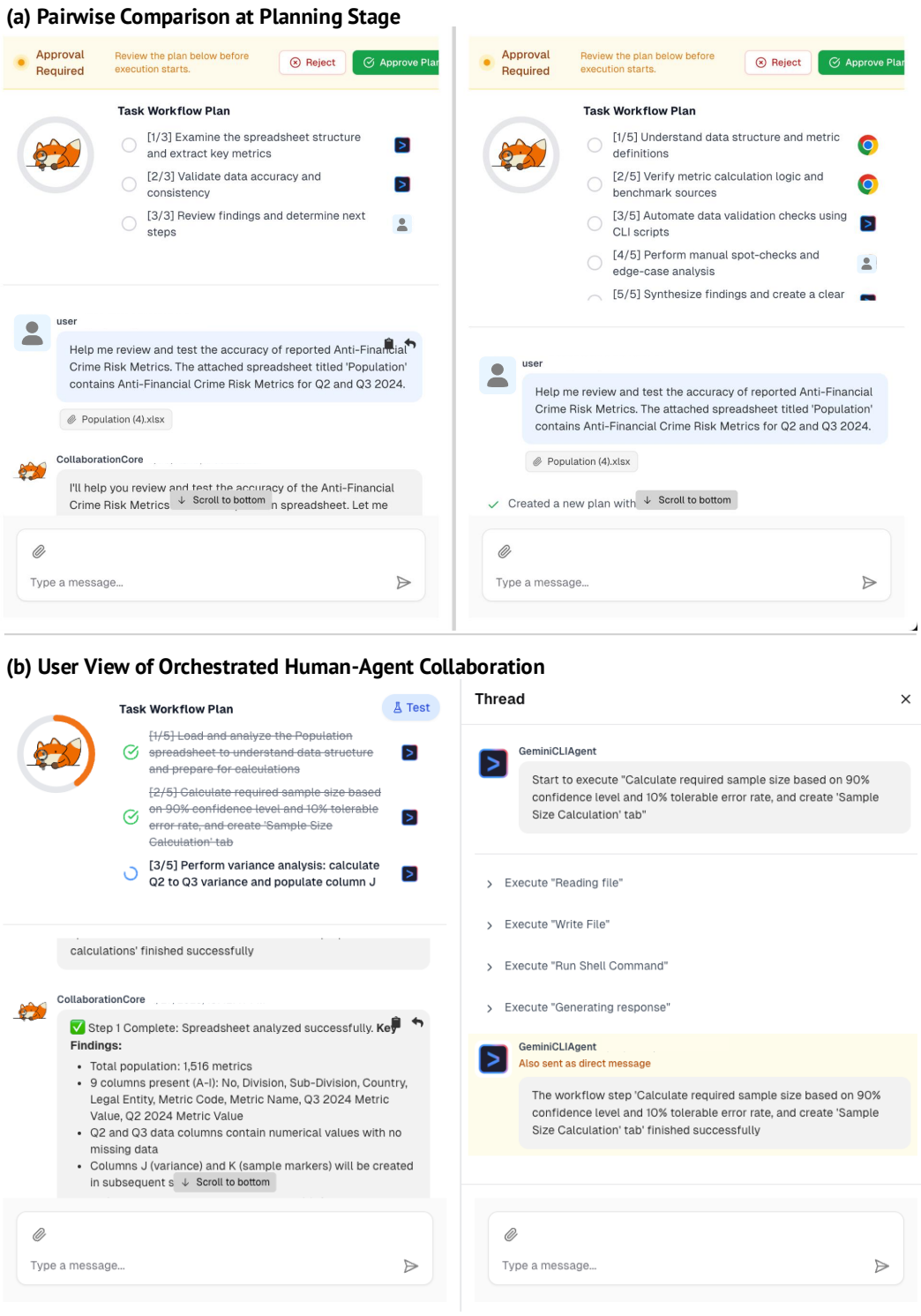

Figure 4: **User study interface for orchestrated human-agent collaboration.** Top: During planning, the participant interacts side-by-side with two Orchestrator policies and provides pairwise preference on them. Bottom: During execution, the interface displays the user-Orchestrator interaction as the main thread, with each delegated workflow step shown as a subthread. A separate panel tracks the status and progress of the workflow plan.

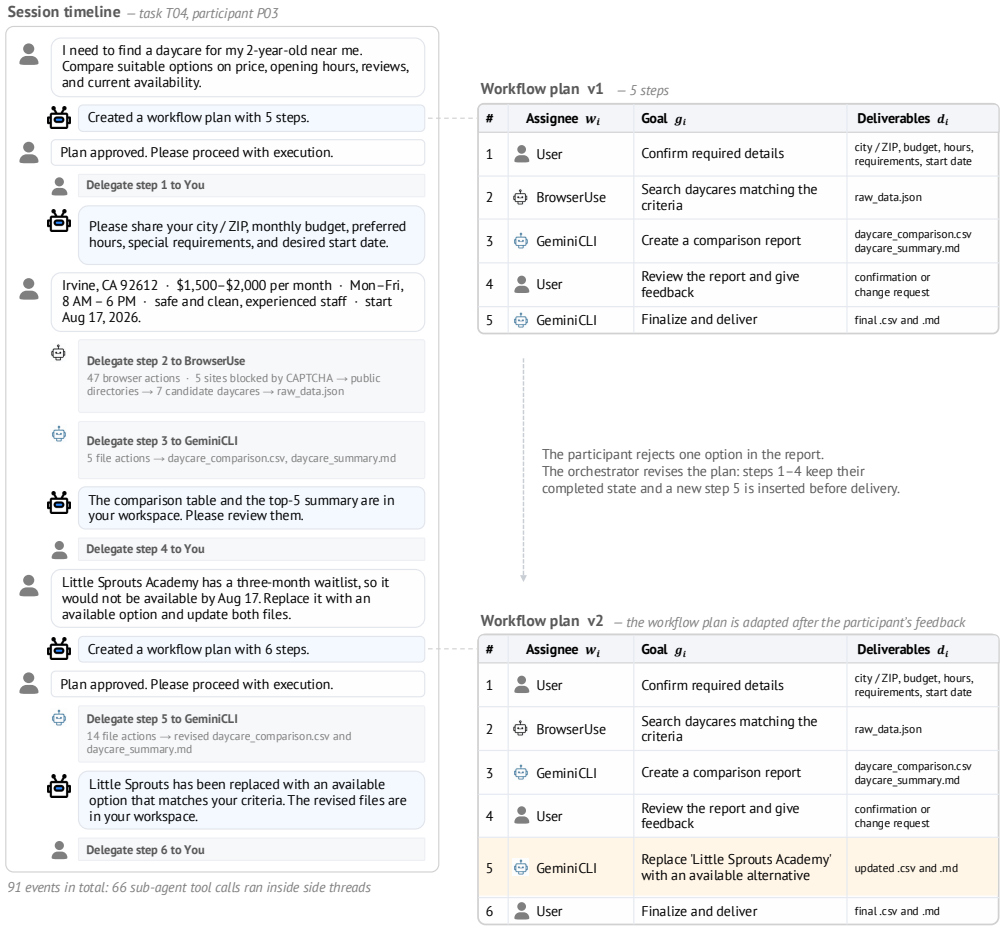

Figure 5: **Human-agent collaboration trajectory T04.** CollabOrchestra assigns the User to confirm key task details, including city/ZIP, budget, hours, requirements, and start dates, and to review the resulting report. In response to the User's feedback, CollabOrchestra adapts the workflow so that CodeAgent incorporates the requested revisions before the User finalizes the report.

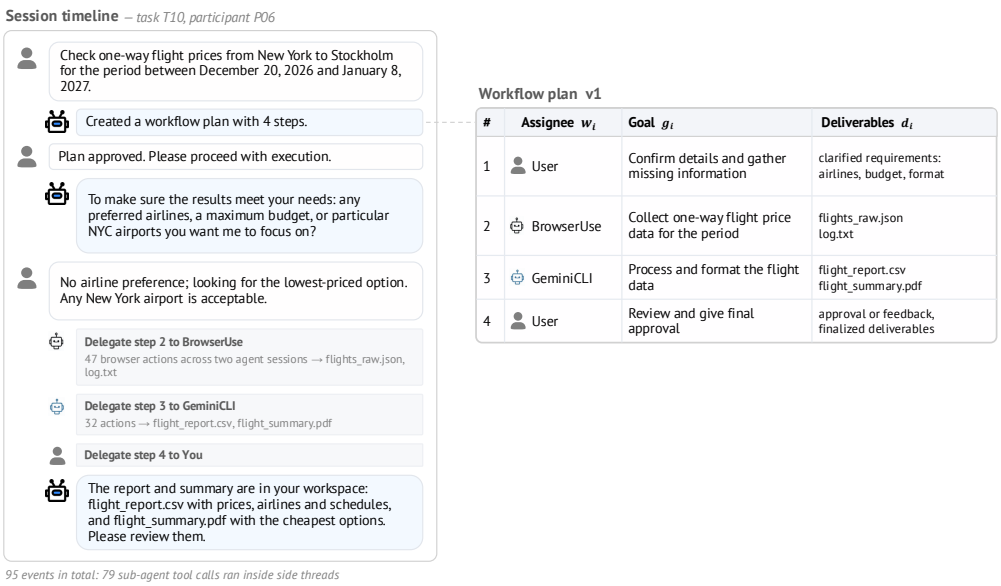

Figure 6: **Human-agent collaboration trajectory T10.** After CollabOrchestra proposes the workflow plan, all steps are successfully delegated and executed in sequence without replanning or further communication with the User.