Orchestrator: Active Inference for Multi-Agent Systems in Long-Horizon Tasks

Lukas Beckenbauer[*†] **Johannes Löwe**[†] **Ge Zheng**[*]
Alexandra Brintrup[*]
[*]Department of Engineering, University of Cambridge, Cambridge, UK
[†]TUM School of Management, Technical University of Munich, Munich, DE

Abstract

Complex, non-linear tasks challenge LLM-enhanced multi-agent systems (MAS) due to partial observability and suboptimal coordination. We propose Orchestrator, a novel MAS framework that leverages attention-inspired self-emergent coordination and reflective benchmarking to optimize global task performance. Orchestrator introduces a monitoring mechanism to track agent-environment dynamics, using active inference benchmarks to optimize system behavior. By tracking agent-to-agent and agent-to-environment interaction, Orchestrator mitigates the effects of partial observability and enables agents to approximate global task solutions more efficiently. We evaluate the framework on a series of maze puzzles of increasing complexity, demonstrating its effectiveness in enhancing coordination and performance in dynamic, non-linear environments with long-horizon objectives.

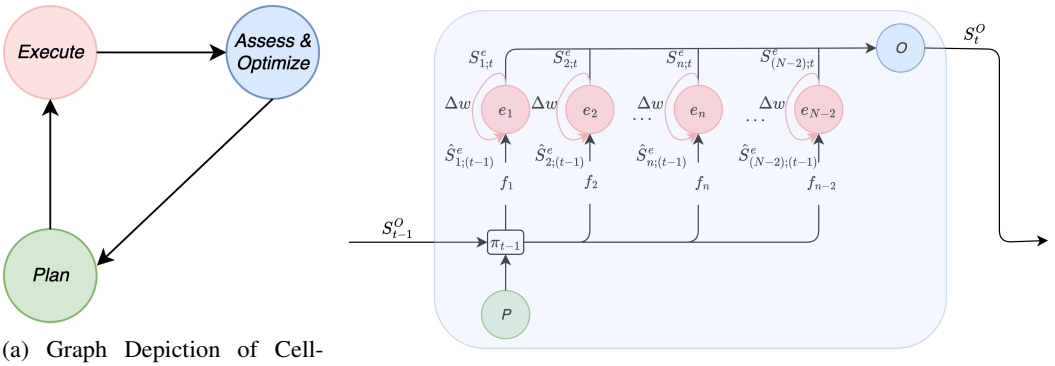
1 Introduction

With the rapid advancement of Large Language Models (LLMs), research on intelligent multi-agent systems (MAS) is gaining new traction. Researchers have investigated use-cases across a broad range of applications, including enhancing the reasoning and task-execution capabilities of general-purpose LLMs [1, 2, 3], supporting software production in recommender systems [4, 5], facilitating data interfacing and visualization [6], and enabling self-supervising supply chain infrastructures [7, 8]. However, while a need for AI-driven MAS solutions that enable advanced agent-coordination and effectiveness across complex, non-linear task-settings has been recognized [9, 1, 10, 2, 11, 12], research on the optimization of system-level MAS-coordination towards task execution for *non-linear*, *long-horizon* problem settings has gained traction only recently [7].

Existing work has primarily advanced MAS by improving feedback loops across agent-to-agent or agent-to-environment settings. In traditional settings this has mainly been pursued via reinforcement learning [13, 14, 15, 16]. More recently, attention has shifted toward LLM-supported multi-agent collaboration, often enhanced by reflective or supervisory mechanisms [17, 18, 19, 16]. While these approaches have demonstrated notable success, they often rely on static topologies [1, 10, 2, 20] and are typically benchmarked on short-horizon, agent-specific tasks such as HumanEval, GPQA [21], or GSM8K [22]. Further, while a dominant body of this work has focused on improving planning efficiency in long-horizon settings of steady levels of complexity [23, 24, 16, 25, 26], there remains limited exploration into how LLM-augmented MAS can be enabled to scale and sustain high-accuracy when addressing advanced, long-horizon tasks characterized by growing complexity levels [27, 28].

To address this challenge, we propose **Orchestrator**—a multi-agent coordination framework with task-observation instance and embedded active inference feedback-loops—and apply it to solving a series of classic maze puzzles with varying difficulty levels (easy, medium, hard). Grounded in active inference principles [29, 26], stating that sentient agents act to minimize surprise and maintain their

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Orchestrator: Active Inference for Multi-Agent Systems in Long-Horizon Tasks.



(a) Graph Depiction of Cell-Internal MAS Workflow

(b) Orchestrator Cell Design. A reprint is available in Appendix A.2.

Figure 1: Orchestrator Framework Overview

internal states by minimizing a quantity called variational free energy (VFE), Orchestrator draws on a benchmark-driven introspection mechanism that considers both, inter-agentic communication [30, 16, 12], and dynamic states between agents' and their immediate environment [11, 30, 31]. We operationalize active inference by contrasting agent's realized information gain with coordination costs and optimizing for free energy (FE) output as a measure of effective task solving. This signal regulates agent autonomy and dynamically adapts system behavior in response to rising decision uncertainty and/or efficiency costs [32, 31, 33]. Subsequently, we address agents' partial observability, as a key limitation to overall operational performance [34], by formulating the iterative approximation of effective agent-to-agent and agent-environment coordination as a quantitative optimization problem.

We evaluate Orchestrator's capacity to overcome local minima, by testing it on a range of maze puzzles with varying levels of escape complexity. Orchestrator outperforms baseline agent ensembles that operate without active-inference benchmarking and dynamic orchestration by an average factor of 3,03 on mazes of 18 x 18 size and medium difficulty. Specifically, our results indicate that, compared to a baseline success rate of 11%, active inference-driven orchestration significantly improves reliability, efficiency, and scalability in long-horizon maze-solving tasks, achieving up to 100% accuracy across 25 runs in medium difficulty and up to 76,67% accuracy in hard mazes of 25 x 25 size.

We validate our results through ablation studies and summarize our key contributions as: (i) we introduce a self-optimizing, scalable cell architecture consisting of a planning, execution, and observation instance—driven by active-inference feedback and task-observation mechanisms, enabling MAS to operate effectively in settings that demand adaptive autonomy; (ii) we propose a set of coordination benchmarks and optimization methods that track both agent's internal decision outcomes and their collaborative behavior, guiding agent cells to away from local task-completion minima and toward globally optimal solution horizons; and (iii) we demonstrate sustained task-completion accuracy across long-horizon maze tasks across various difficulty levels, using only lightweight, resource-efficient LLM models, thus aligning with production-ready deployment scenarios under strict budget and resource constraints.

2 Related Work

Maze-Assessment as Long-Horizon Benchmark for MAS. Maze-based environments have become a central testbed for evaluating the reasoning, planning, and coordination abilities of intelligent agents [35, 28, 36]. Early benchmarks focus on single-agent navigation in static, fully observable mazes, where classic algorithms such as A*, Flood Fill, DFS, or multi-agent pathfinding (MAPF) [37] in non-LLM contexts are used to assess pathfinding and basic spatial reasoning capabilities [38, 39, 40, 36].

More recent work has produced a new generation of maze benchmarks that probe the limits of agent memory, adaptability, and sequential decision-making. Memory Gym [41], for example, introduces endless, procedurally generated environments to test agents' memory effectiveness over unbounded horizons. MazeBench [28] shifts the focus to LLMs, using tokenized maze representations, reinforcement learning, and chain-of-thought prompting to evaluate step-by-step spatial reasoning

in small-size mazes. MazeEval [42] isolates pure spatial reasoning by requiring LLMs to navigate mazes using only coordinate and distance-to-wall feedback, without visual input. Finally, MAPF has been proposed as a structured LLM benchmark, highlighting the unique difficulties of multi-agent coordination, long-horizon planning, and symbolic map understanding [43].

Despite these advances, the results of existing maze benchmarks reveal that LLMs and RL agents struggle with the combinatorial demands of multi-agent coordination, especially in environments with a high degree in path deviations and obstacles [43, 36]. Spatial and long-horizon reasoning present core challenges, with LLMs often failing to build robust internal representations and implement effective solutions towards global task completion. Further, structural limitations, such as context window size and lack of scalable memory-architectures hinder performance on large or complex mazes. Our work directly addresses these research gaps by introducing a unified, dynamic, and scalable framework that expands the limitations of prior LLM-based maze solving approaches, by supporting long-horizon task completion, active inference-based optimization loops, and real-time performance assessment for LLM-based agents.

Reflective Instances in Multi-Agent Settings. To orchestrate multi-agent interactions, previous research, such as [44, 1, 17], has introduced reflective mechanisms that optimize agent-to-agent, and/or agent-to-environment coordination. Bo et al. [30] introduce the COPPER framework, which implements a fine-tuned LLM to critique and refine outputs of primary agents, drawing on a reward mechanism to assess each agents overall contribution to over task success, and helping agents to perform on specific baseline benchmarks, including HotPotQA [45], GSM8K [46], and 'Checkmate in One Move' [47]. Similarly, Nayak et al. [18] implement a plan-act-correct-verify mechanism to help an LLM-guided robot agent to autonomously navigate a 3D-environment; Leveraging visual feedback between environment and internal reflection instances to enhance navigational accuracy of the agent. Likewise, Xie et al. [48] utilize reinforcement learning to improve an LLM-agent's ability to critique its own work, improving relative performance by 106% on coding benchmarks. Whereas Ding et al. [15] propose an asynchronous communication framework to optimize decision-capabilities in MAS, reducing error margins due to sequence-related circular dependencies in linear workflows.

Agents Partial Observability Limitations. While several authors have assessed the benefits of reflective instances to overcome agent's partial observability problem [18, 49], this strand of research is closely aligned with successes in agent-to-agent reinforcement learning pipelines [44]. Expanding on these previous approaches, but aiming for emergent performance without reinforcement-driven validation, Ke et al.[50] discuss an emergent zero-supervision MAS framework to overcome agent's partial observability problems and solve tasks at advanced complexity, by implementing a reflective meta-level instance that optimizes agent-to-agent sequences locally and need-based. In parallel, work such as [51] and [20] introduces dynamic social graphs, or graph-attention paradigms [52, 53] to support emergent multi-agent interaction in non-linear problem settings. While further, evolving orchestration approaches have been proposed [17, 19, 20, 23] that formalize agent interaction as a directed graph, with a central orchestrator dynamically selecting and sequencing agent activations based on evolving task states, yielding more compact and efficient collaboration patterns. However, while these approaches are able to handle tasks at advanced complexity, they fall short in task completion that require a high number of steps across long-term planning and problem solving. To address these gaps, more recent work has suggested active inference principles, a neuroscience grounded paradigm that can be leveraged to aid agent's reasoning capabilities and drive task completion successes via quantified feedback principles [33].

Applications of Active Inference in MAS. We integrate active inference principles by implementing a reflective benchmarking mechanisms to capture agents' *free energy* [32, 26, 33] and to control for system-wide entropy metrics and enhance long-term adaptability by nudging the system to reduce its error rates. There is a long tradition of benchmark-driven optimization in MAS research, which has found special traction in multi-agent reinforcement learning (MARL) contexts, such as [52, 15, 54, 55]. Authors justify the need for dynamic benchmarks to optimize agent behavior in highly dynamic and hard to predict environments. For example, Suri et al. [32] stabilize multi-agent interactions by collectively minimizing free energy across agent distributions, effectively reducing the occurrence of unexpected states. While, Ruiz-Serra et al.[31] integrate active inference frameworks to incorporate agents' assumption about other agents' internal states for strategic decision-making in iterative scenarios. In this work, we draw on recent advances on reflective mechanisms in LLM-based agent-optimization [53, 19, 30, 23] and merge the approach with active inference dynamic-optimization mechanisms, as demonstrated in [12, 31, 33]. Our system uniquely

synthesizes these reflection-based and information-theoretic insights by implementing a reflective, benchmark-driven orchestration instance to continuously evaluate and enhance agent-to-agent and agent-to-environment interactions.

In summary, Orchestrator unifies three key advances in the above literature: First, we implement a modular 'cell-structured' graph design, embedding planning, execution, and orchestration as explicit computational stages within each 'cell'. Second, we introduce a reflective benchmarking mechanism, using active inference principles to monitor FE-grounded intra- and inter-cell performance metrics to continuously assess and adapt local agent routines in light of global progress. Lastly, we leverage these performance metrics to foster dynamic adjustments of LLM's internal policy and prompt design, nudging agents to adjust their behavior if they encounter local solution minima or exhibit other behavior that is stalling progress. In this manner, and optimizing for both immediate and longitudinal performance, Orchestrator's cell-based task-execution design empowers benchmark-driven coordination between components of an agent ensemble, proactively reducing error potential while improving task accuracy at the local and systemic level.

3 Orchestrator Framework

3.1 Graph-Based and Dynamic Multi-Agent Architecture

We formalize the **Orchestrator** framework shown in Figure 1 as a unified graph-based architecture that enables dynamic behavioral adaptation and real-time coordination. The system's update sequence is modeled as a directed graph in Figure 1a, while each system state at iteration *t* is represented by the agentic 'cell' architecture, illustrating agent interactions and coordination dynamics as depicted in Figure 1b.

Mathematically, the Orchestrator framework is defined as,

$$\text{Orchestrator} = G(N, E, F) \tag{1}$$

Where $N = N_{\text{plan}} \cup N_{\text{exec}} \cup N_{\text{orch}}$ denotes the complete node set, with each node corresponding to an agent. The framework consists of three node types: the planning node (N_{plan}), the execution node (N_{exec}), and the orchestration node (N_{orch}). For this work, we implement a single orchestrator cell-instance, comprising one planning node and one orchestration node, with the remaining $(N-2)$ nodes instantiated as execution nodes. The plan node, $N_{\text{plan}} = \{P\}$, provides a sequence of strategic action steps that guide execution nodes. Execution nodes, $N_{\text{exec}} = \{e_1, e_2, \ldots, e_n, \ldots, e_{N-2}\}$, powered by LLMs, interpret the policy prompt and act in the environment. Where directed edges $E \subseteq (N \times N)$ are associated with routing functions, $G_r(t) = \{g_{r(1;t)}, g_{r(2;t)}, \ldots, g_{r(n;t)}, \ldots, g_{r(N-2;t)}\}$, that define define interaction pathways among agents at iteration t. The action performance of each execution node is continuously evaluated by its variational free energy (VFE) following active inference principles (described in more detail in section 3.2). This ensures that actions are selected to maximize information gain, where high uncertainty corresponds to active exploration, while driving agents toward the global objective of maze completion. While execution nodes have knowledge of other agent's explored maze junctions at all times, the orchestration node, $N_{\text{orch}} = \{O\}$, serves as a communication hub and global memory, allowing execution nodes to share additional information (such as other agent error rates or dead end detections) indirectly.

At the beginning of the maze exploration, the execution nodes N_{exec} and the orchestration node N_{orch} are initialized with distinct states, denoted as S_t^e and S_t^o, respectively, which are dynamically updated over time. For the purpose of demonstration, and while future implementations may incorporate adaptive planning, we further initialize planning node P with a preset sequence of *k* steps, which is passed as loop of actionable instructions into execute nodes at $S_{n;(t-1)}$ (see section A.1 in the Appendix). Each execute node E is equipped with its own, active-inference-based optimization function f_N, described in detail in the section 3.2. Orchestration node O maintains global state including the information of states of all execute nodes, while the state of each execute node is defined as a local state. At the start of each iteration t, each execute node considers (i) its own state and (ii) the states of other execute nodes updated by the orchestrator node and stored in its temporary state S_t and then takes actions following its internal policy π and the actionable-guidance P_{plan} of length k. Each node $e^n_{S(t-1)}$ then loops through a predefined sequence of k steps until plan completion or intervention triggers are met (Appendix A.1).

After completion of each step k, the weight of each execute node, written as Δw, is updated by calculating its VFE through the respective optimization function, $\{f_1, f_2, \ldots, ...f_n, \ldots, f_{N-2}\}$. The outcomes of these computations are then used to dynamically inject guidance instructions into temporary local states of execution nodes, $\{\hat{S}^e_{1;(t-1)}, \hat{S}^e_{2;(t-1)}, \ldots, \hat{S}^e_{n;(t-1)}, \ldots, \hat{S}^e_{(N-2);(t-1)}\}$. Next, at the execution layer, the temporary local states are consolidated into updated local states, $\{S^e_{1;t}, S^e_{2;t}, \ldots, S^e_{n;t}, \ldots, S^e_{(N-2);t}\}$. This system state is then passed to the orchestration node O, which reviews agent progress and provide direct optimization recommendations for execution agents. These recommendations are delivered through dynamic prompt injections that update the execution agents' internal policy, π^E_{t-1}, as the global state S^O_t is passed to the next iteration. The detailed optimization computation for agents is explained in section 3.2 and we provide a sequential overview of how the orchestration update sequence operates during maze exploration in Figure 8.

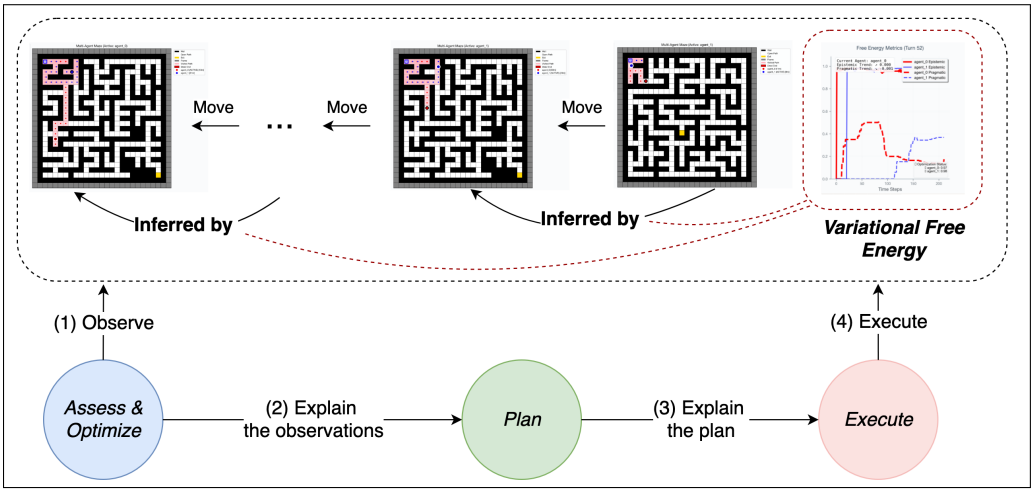


Figure 2: Schematic representation of Orchestrator's decision-making cycle while solving a medium-difficulty maze-puzzle across n-steps.

3.2 Active Inference Benchmarking and Performance Assessment

Building on active inference principles, we reformulate the VFE objective. Instead of solely minimizing surprise to reduce deviations from the model predictions, we define the objective as a balanced trade-off: maximizing agents' realized information gain to encourage active learning, while offsetting this with an explicit cost function that captures coordination demands and behavioral efficiency. We provide the extended formalism to this approach in section A.3 in the Appendix and operationalize it as follows:

Epistemic Uncertainty. We quantify epistemic uncertainty through measuring information entropy between consecutive states $\{S_t, S_{t+1}\}$, providing a real-time estimate of each agent's rate of actual information gain

$$U_{\text{epistemic}}(n,t,k) = -H[S_{n,t,k} \mid S_{n-1,t-1,k-1}] \tag{2}$$

where $H[S_{n,t,k}]$ denotes the normalized Shannon entropy of the message output for agent n at iteration t and step k. The entropy is computed over individual message tokens j as:

$$H_{\text{tokens}}(n,t,k) = -\sum_{j \in K_{\text{message}}} p_j(k) \log p_j(k) \tag{3}$$

Accuracy Cost Assessment. As a counterbalance to the agent's epistemic uncertainty, we operationalize the accuracy principle of VFE as cost term. While in principle the VFE accuray term corresponds to the expected negative log-likelihood of observed outcomes under the generative model, direct access to these likelihoods is unavailable due to the underlying LLM architectures. As described in section A.3, we therefore approximate accuracy cost using a behavioral proxy that captures behavioral efficiency and coordination at each step k:

$$C_{\text{accuracy}}(n,t,k) = \sum_{j=1}^{5} w_j \cdot R_j(n,t,k) \tag{4}$$

where the static weights $w_j = 0.20$ for all $j \in \{1, 2, 3, 4, 5\}$ equally weight a set of five predefined risk components:

1. **Movement Efficiency**: $R_1(n, t, k) = 1 - \frac{\text{total moves}}{\text{total move attempts}}$
2. **Exploration Efficiency**: $R_2(n, t, k) = 1 - \frac{\text{unique positions visited}}{\text{total moves}}$
3. **Backtracking Patterns**: $R_3(n, t, k) = \text{backtrack ratio} + 1.5 \cdot \text{oscillation penalty}$
4. **Dead-End Recognition**: $R_4(n, t, k) = 1 - \frac{\text{dead-end revisits}}{\text{total moves}}$
5. **Oscillation Avoidance**: $R_5(n, t, k) = 1 - \frac{\text{unique positions in recent moves}}{\text{recent move count}}$

Behavioral Optimization Function. To normalize outputs, we cap both uncertainty and cost terms at ± 2.0 and define the variational free energy $\mathcal{F}_n(t, k)$ for each agent n at iteration t and step k as:

$$F_n(t, k) = U_{\text{epistemic}}(n, t, k) - C_{\text{accuracy}}(n, t, k) \tag{5}$$

The subtractive formulation reflects our optimization principle: high epistemic uncertainty signals productive information gain (positive contribution), whereas high pragmatic costs penalizes counterproductive behaviors including oscillation patterns, redundant exploration, or movement failures (negative contribution).

Performance Policies. Based on free energy outcomes, the system assigns agents to one of four performance categories, where threshold variables ϑ_1 and ϑ_2 have been deliberately assigned to match best performance using grid-search as determining method (see section A.5 in the Appendix).

- **High Epistemic Drive, Low Accuracy Cost** ($U_{\text{epistemic}} > 0.6$, $C_{\text{accuracy}} < 0.4$): Effective exploration with efficient execution
- **High Epistemic Drive, High Accuracy Cost** ($U_{\text{epistemic}} > 0.6$, $C_{\text{accuracy}} > 0.4$): Active discovery but inefficient execution
- **Low Epistemic Drive, Low Accuracy Cost** ($U_{\text{epistemic}} < 0.6$, $C_{\text{accuracy}} < 0.4$): Consistent execution but limited exploration
- **Low Epistemic Drive, High Accuracy Cost** ($U_{\text{epistemic}} < 0.6$, $C_{\text{accuracy}} > 0.4$): Poor exploration and inefficient execution

Dynamic Weight Modulation. Each performance category triggers adjustments to a set of behavioral weights:

$$\mathbf{w}_n(t, k) = \{\, w_{\text{explore}}(t, k),\, w_{\text{exploit}}(t, k),\, w_{\text{coordinate}}(t, k),\, w_{\text{backtrack}}(t, k)\} \tag{6}$$

which are updated as:

$$\mathbf{w}_n(t, k) = \mathbf{w}_{\text{base}} + \Delta\mathbf{w}(F_n(t, k), \nabla F_n(t, k)) \tag{7}$$

where $\Delta\mathbf{w}$ accounts for both current free-energy and its temporal gradient, $\nabla\mathcal{F}_n(t, k)$, thus incorporating predictive dynamics. For example, agents in the expressing high epistemic drive but high costs, are assigned increased w_{exploit} weights to improve execution efficiency, while agents with low drive but effective cost management, receive higher w_{explore} weights to encourage broader environmental exploration. Further details on movement reward scoring and dynamic weight updates are provided in sections A.1 and A.4. Prompt design of agents are presented in sections A.7 and A.8.

4 Experiments

We evaluate our approach on a suite of synthetic maze environments designed to stress-test reasoning and coordination across long horizon task settings. We draw on the AMaze benchmark [36], which is designed to procedurally generate challenging maze environments and assess generalization ability of RL-agents across (long-horizon) task settings. The algorithmic design of our AMaze implementation is provided in Appendix section A.9.

Challenge. We consider three maze difficulty levels (*easy, medium*, *hard*), each instantiated with five unique mazes. As defined by the AMaze benchmark, mazes differ in length, branching factor, and required coordination. A run is considered successful if one of n execution agents reaches the maze exit within the maximum step budget. As maximum step budget we allocate a heuristic of two-and-a-half times the number of tiles per maze configuration, as well as a maximum duration of 7200 seconds, if one of these conditions is reached, timeout is initialized and the maze exploration is considered as failed.

Agents. To ensure efficiency and responsiveness under real-world deployment constraints—where computational resources and budget are critical—we instantiate our agents using state-of-the-art, but compact, fast-inference LLMs (specifically GPT-4.1-nano and GPT-5-nano). For the purpose of demonstration, we present experiments with n=2 execution agents, only. While preliminary tests have showcased the feasibility of n=1 or n=3 agents, we consider n=2 a balanced trade-off in terms of efficiency, speed, and resource-allocation for the purpose of maze exploration. Further, while mixed-model approaches are possible, we constrain the orchestrator, when enabled, to being identical to the execution model, privileging a homogeneous approach.

Baselines and Experiment Configurations. We evaluate three core experimental configurations to systematically assess the impact of benchmarking and orchestration. As a floor baseline, we implement a random walk agent: a memory-enhanced, single-agent policy that self-selects valid moves at each step, with no access to FE-benchmarking, or orchestration support. Second, we introduce FE-benchmarking, providing agents with real-time feedback and dynamic weight adjustments, based on their performance. The third configuration adds an orchestrator node in addition to FE-benchmarking, to test for the models ability to facilitate higher-level coordination between agents. To save compute, we omit the random walk for hard-level difficulty as chance of success is considerably low, as well as assessment of the FE + orchestration configuration on easy-level difficulty, as chances for success are considerably high.

Execution and Evaluation Metrics. We execute at least 10 runs per configuration and level of difficulty, across 15 mazes in a balanced setting, yielding a minimum of 150 runs in total. For each configuration, we report key metrics such as success rate, total number of steps taken, number of failed moves, and total costs (normalized API token usage in dollars). We avoid wall-clock time measures, as provider rate limits and real-time changes on OpenAI API demand distort the results. Further, we predeclare a precision target of ± 15 percentage points (pp) for success rate CIs. Runs are increased until this target is reached.

5 Results and Discussion

Table 1: Success rates with Wilson 95% confidence intervals (CI) and corresponding half-widths in percentage points (pp) per model configurations and maze difficulty levels.

Configuration	Difficulty	# of Runs	Successes	Success Rate (%)	95% CI Lower	95% CI Upper	Half-width (pp)
gpt-4.1-nano (Solo)	easy	34	11	32.35	19.13	49.16	15.01
gpt-4.1-nano (Solo)	medium	33	10	30.3	17.38	47.34	14.98
gpt-4.1-nano + FE Benchmark only	easy	10	10	100.0	72.25	100.0	13.88
gpt-4.1-nano + FE Benchmark only	medium	36	26	72.22	56.01	84.15	14.07
gpt-4.1-nano + FE Benchmark only	hard	26	22	84.62	66.47	93.85	13.69
gpt-4.1-nano + FE + Orchestration Node	medium	25	25	100.0	86.68	100.0	6.66
gpt-4.1-nano + FE + Orchestration Node	hard	32	23	71.88	54.63	84.44	14.9
gpt-5-nano (Solo)	easy	10	0	0.0	0.0	27.75	13.88
gpt-5-nano (Solo)	medium	11	0	0.0	0.0	25.88	12.94
gpt-5-nano + FE Benchmark only	easy	10	10	100.0	72.25	100.0	13.88
gpt-5-nano + FE Benchmark only	medium	25	20	80.0	60.87	91.14	15.14
gpt-5-nano + FE Benchmark only	hard	36	23	63.89	47.58	77.52	14.97
gpt-5-nano + FE + Orchestration Node	medium	24	20	83.33	64.15	93.32	14.59
gpt-5-nano + FE + Orchestration Node	hard	30	23	76.67	59.07	88.21	14.57

We present our results in Table 1. For solo agent ensembles without benchmark and orchestration nodes, we find poor model performance in runs on easy-level (32,35% success rate for GPT-4.1-nano; 0% success rate for GPT-5-nano) and medium-level difficulty (30,03% for GPT-4.1-nano; 0% for GPT-5-nano). In contrast, Orchestrator strongly outperforms solo agent ensembles on easy (100% for both GPT-nano models) and medium-difficulty levels (100% for GPT-4.1.-nano and 83.33% for for GPT-5-nano), accounting for a threefold increase in success rates for GPT-4.1-nano configurations and an increase by a factor of 3,33 and 2,77 respectively for easy and medium levels in GPT-5-nano configurations. Further, we find that just incorporating FE-benchmarks, already substantially improves model performance (medium: 72.22% for GPT-4.1-nano, 80.0% for GPT-5-nano; hard: 84.62% for GPT-4.1-nano, 63.89% for GPT-5-nano). Lastly, adding orchestration improves performance for 3 out of 4 configurations (medium: 100% for GPT-4.1-nano, 83.33% for GPT-5-nano; hard: 71.88% for GPT-4.1-nano, 76.67% for GPT-5-nano). Surprisingly, GPT-4.1-nano with FE benchmarks outperforms the same model with added orchestration on hard mazes (84.62% vs. 71.88%). One possible explanation is that orchestration, while generally beneficial, can introduce additional reasoning overhead in high-complexity environments, exceeding the model's effective planning horizon [27]. In such cases, more streamlined reasoning—guided by FE benchmarks

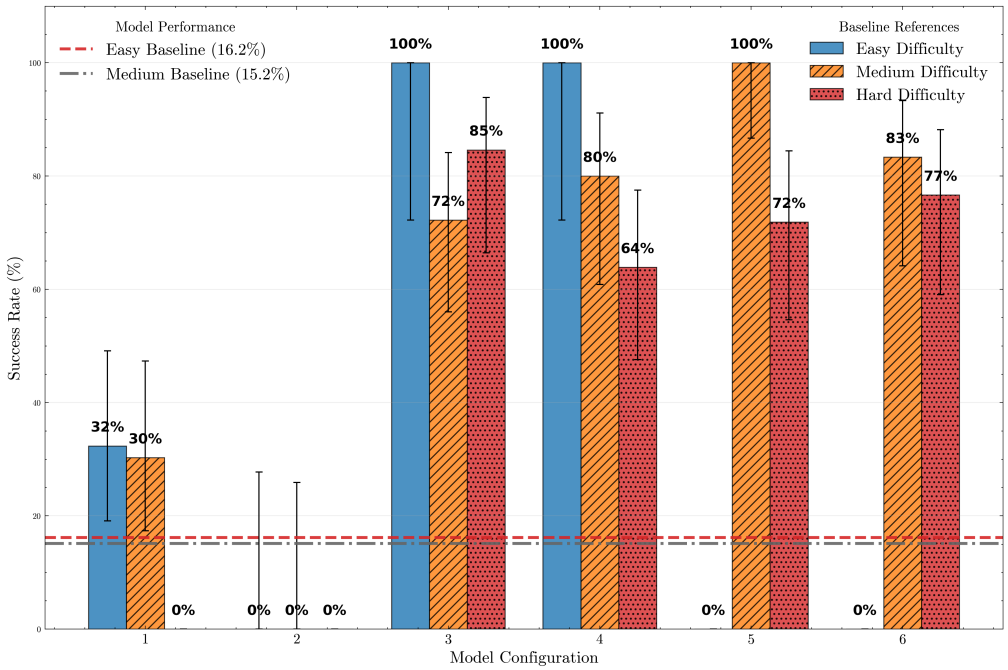


Figure 3: Success rate and Wilson CI ranges by model configuration and difficulty as shown in Table 1. Numerical identifiers for configurations are as follows: 1) GPT-4.1-nano (Solo); 2) GPT-5-nano (Solo); 3) GPT-4.1-nano + FE Benchmark only; 4) GPT-5-nano + FE Benchmark only; 5) GPT-4.1-nano + FE + Orchestration Node; 6) GPT-5-nano + FE + Orchestration Node.

alone—may yield better results [33, 56]. This suggests that while additional orchestration instances improve task-completion accuracy for most scenarios [23, 24, 19], their actual utility may depend on correctly applying the ensemble composition of agents to match the specific task at hand. Additional results concerning model cost-effectiveness, completion efficiency, and convergence intervals per configuration are listed in sections A.6.4, and A.6.5 in the Appendix.

6 Conclusion

This paper introduced Orchestrator, a unified active inference-based framework for multi-agent coordination in long-horizon environments. Motivated by the limitations of existing long-horizon, maze solving approaches—which often fail to capture the challenges of memory-retention, adaptability, and long-horizon coordination—Orchestrator integrates dynamic feedback, reflective benchmarking, and modular orchestration into a single, scalable architecture.

Our results highlight several key contributions: First, we propose a novel active-inference-based architecture for multi-agent coordination that supports real-time adaptation, memory-driven collaboration, and scalable reasoning. Second, we show that this approach achieves strong performance even with lightweight, fast-inference models suitable for real-world deployment, but is able to accommodate more sophisticated reasoning agents in alignment with rising task complexity. Third, we demonstrate that the combination of active-inference benchmarking and orchestration substantially improves both the reliability and efficiency of agent teams, particularly in complex, long-horizon tasks. By bridging the gap between static benchmarks and adaptive, feedback-driven orchestration, our work contributes to the debate on foundational frameworks for robust, autonomous, and LLM-based multi-agent systems capable of addressing long-horizon challenges in real-world production settings.

Nevertheless several limitations remain: As the present analysis is restricted to synthetic maze environments and small agent teams, the ability of Orchestrator to address long-horizon tasks in more heterogeneous settings and across other problem domains remains to be explored. Further, to assess essential deployment factors in terms of scalability, cost, and performance, framework performance should be extensively tested for addressing higher-complexity tasks while using larger LLM models. Lastly, given the fact that our architecture achieves high level performance with smaller-size, high inference-speed LLMs, we see great potential for deploying Orchestrator using open-source models. The feasibility of these approaches should be tested in future iterations.

References

[1] Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors in Agents. 2024.

[2] Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. More Agents Is All You Need, October 2024. arXiv:2402.05120 [cs].

[3] Jintian Zhang, Xin Xu, Ningyu Zhang, Ruibo Liu, Bryan Hooi, and Shumin Deng. Exploring Collaboration Mechanisms for LLM Agents: A Social Psychology View, May 2024. arXiv:2310.02124 [cs].

[4] Yuanzhe Liu, Ryan Deng, Tim Kaler, Xuhao Chen, Charles E. Leiserson, Yao Ma, and Jie Chen. Lessons Learned: A Multi-Agent Framework for Code LLMs to Learn and Improve, 2025. Version Number: 1.

[5] Junda He, Christoph Treude, and David Lo. LLM-Based Multi-Agent Systems for Software Engineering: Literature Review, Vision and the Road Ahead, July 2025. arXiv:2404.04834 [cs].

[6] Chao Xu, Qi Zhang, Baiyan Li, Anmin Wang, and Jingsong Bao. Visual analysis of time series data for multi-agent systems driven by large language models. In Proceedings of the 3rd International Conference on Signal Processing, Computer Networks and Communications, SPCNC '24, page 427–431, New York, NY, USA, 2025. Association for Computing Machinery.

[7] Liming Xu, Sara Almahri, Stephen Mak, and Alexandra Brintrup. Multi-Agent Systems and Foundation Models Enable Autonomous Supply Chains: Opportunities and Challenges. IFAC-PapersOnLine, 58(19):795–800, 2024. Publisher: Elsevier BV.

[8] Liming Xu, Stephen Mak, Maria Minaricova, and Alexandra Brintrup. On Implementing Autonomous Supply Chains: a Multi-Agent System Approach, June 2024. arXiv:2310.09435 [cs].

[9] Tal Alon, Magdalen Dobson, Ariel Procaccia, Inbal Talgam-Cohen, and Jamie Tucker-Foltz. Multiagent Evaluation Mechanisms. Proceedings of the AAAI Conference on Artificial Intelligence, 34(02):1774–1781, April 2020.

[10] Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Optima: Optimizing Effectiveness and Efficiency for LLM-Based Multi-Agent System, February 2025. arXiv:2410.08115 [cs].

[11] Wenlin Yao, Haitao Mi, and Dong Yu. HDFlow: Enhancing LLM Complex Problem-Solving with Hybrid Thinking and Dynamic Workflows, September 2024. arXiv:2409.17433 [cs].

[12] Rithvik Prakki. Active Inference for Self-Organizing Multi-LLM Systems: A Bayesian Thermodynamic Approach to Adaptation, January 2025. arXiv:2412.10425 [cs].

[13] Shariq Iqbal and Fei Sha. Actor-Attention-Critic for Multi-Agent Reinforcement Learning. 2018.

[14] Zeyang Liu, Xinrui Yang, and Shiguang Sun. Grounded Answers for Multi-agent Decision-making Problem through Generative World Model. 2024.

[15] Ziluo Ding, Zeyuan Liu, Zhirui Fang, Kefan Su, Liwen Zhu, and Zongqing Lu. Multi-Agent Coordination via Multi-Level Communication. 2024.

[16] Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. Plan-and-Act: Improving Planning of Agents for Long-Horizon Tasks, April 2025. arXiv:2503.09572 [cs].

[17] Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Language Agents as Optimizable Graphs, August 2024. arXiv:2402.16823 [cs].

[18] Siddharth Nayak, Adelmo Morrison Orozco, Jackson Zhang, Darren Chen, Aditya Kapoor, Eric Robinson, Karthik Gopalakrishnan, James Harrison, Brian Ichter, Anuj Mahajan, and Hamsa Balakrishnan. Long-Horizon Planning for Multi-Agent Robots in Partially Observable Environments. 2024.

[19] Edward Y. Chang and Longling Geng. SagaLLM: Context Management, Validation, and Transaction Guarantees for Multi-Agent LLM Planning, July 2025. arXiv:2503.11951 [cs].

[20] Boyi Li, Zhonghan Zhao, Der-Horng Lee, and Gaoang Wang. Adaptive Graph Pruning for Multi-Agent Communication, June 2025. arXiv:2506.02951 [cs].

[21] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. GPQA: A Graduate-Level Google-Proof Q&A Benchmark. In First Conference on Language Modeling, 2024.

[22] Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. AI Agents That Matter. Transactions on Machine Learning Research, June 2025.

[23] Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, Xuantang Xiong, Lei Han, Zhiyuan Liu, and Maosong Sun. Multi-Agent Collaboration via Evolving Orchestration, May 2025. arXiv:2505.19591 [cs].

[24] Enhao Zhang, Erkang Zhu, Gagan Bansal, Adam Fourney, Hussein Mozannar, and Jack Gerrits. Optimizing Sequential Multi-Step Tasks with Parallel LLM Agents, July 2025. arXiv:2507.08944 [cs].

[25] Yijia Xiao, Edward Sun, Di Luo, and Wei Wang. TradingAgents: Multi-Agents LLM Financial Trading Framework, June 2025. arXiv:2412.20138 [q-fin].

[26] Michael Walters, Rafael Kaufmann, Justice Sefas, and Thomas Kopinski. Free energy risk metrics for systemically safe ai: Gatekeeping multi-agent study, 2025. arXiv:2502.04249 [cs.AI].

[27] Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Lens of Problem Complexity, July 2025. arXiv:2506.06941 [cs].

[28] Alan Dao and Dinh Bach Vu. AlphaMaze: Enhancing Large Language Models' Spatial Intelligence via GRPO. arXiv preprint arXiv:2502.14669, 2025.

[29] Thomas Parr, Giovanni Pezzulo, and Karl J. Friston. Active Inference: The Free Energy Principle in Mind, Brain, and Behavior. The MIT Press, 03 2022.

[30] Xiaohe Bo, Zeyu Zhang, Quanyu Dai, Xueyang Feng, Lei Wang, Rui Li, Xu Chen, and Ji-Rong Wen. Reflective Multi-Agent Collaboration based on Large Language Models. 2025.

[31] Jaime Ruiz-Serra, Patrick Sweeney, and Michael S. Harré. Factorised Active Inference for Strategic Multi-Agent Interactions, May 2025. arXiv:2411.07362 [cs].

[32] Karush Suri, Xiao Qi Shi, Konstantinos Plataniotis, and Yuri Lawryshyn. Surprise Minimizing Multi-Agent Learning with Energy-based Models. 2022.

[33] Yavar Taheri Yeganeh, Mohsen Jafari, and Andrea Matta. Deep Active Inference Agents for Delayed and Long-Horizon Environments, May 2025. arXiv:2505.19867 [cs].

[34] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P. How, and John Vian. Deep Decentralized Multi-task Multi-Agent Reinforcement Learning under Partial Observability, July 2017. arXiv:1703.06182 [cs].

[35] Manousos Linardakis, Iraklis Varlamis, and Georgios Th. Papadopoulos. Distributed Maze Exploration Using Multiple Agents and Optimal Goal Assignment. IEEE Access, 12:101407–101418, 2024.

[36] Kevin Godin-Dubois, Karine Miras, and Anna V Kononova. AMaze: An Intuitive Benchmark Generator for Fast Prototyping of Generalizable Agents. Frontiers in Artificial Intelligence, 8:1511712, 2025.

[37] Roni Stern, Nathan Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, TK Kumar, et al. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In Proceedings of the International Symposium on Combinatorial Search, volume 10, pages 151–158, 2019.

[38] Daniel Foead, Alifio Ghifari, Marchel Budi Kusuma, Novita Hanafiah, and Eric Gunawan. A Systematic Literature Review of A* Pathfinding. Procedia Computer Science, 179:507–514, 2021.

[39] Semuil Tjiharjadi, Sazalinsyah Razali, and Hamzah Asyrani Sulaiman. A Systematic Literature Review of Multi-agent Pathfinding for Maze Research. Journal of Advances in Information Technology, 13(4), 2022.

[40] Ning Liu, Sen Shen, Xiangrui Kong, Hongtao Zhang, and Thomas Bräunl. Cooperative Hybrid Multi-Agent Pathfinding Based on Shared Exploration Maps, March 2025. arXiv:2503.22162 [cs].

[41] Marco Pleines, Matthias Pallasch, Frank Zimmer, and Mike Preuss. Memory Gym: Towards Endless Tasks to Benchmark Memory Capabilities of Agents. Journal of Machine Learning Research, 26(6):1–40, 2025.

[42] Hafsteinn Einarsson. MazeEval: A Benchmark for Testing Sequential Decision-Making in Language Models. arXiv preprint arXiv:2507.20395, 2025.

[43] Weizhe Chen, Sven Koenig, and Bistra Dilkina. Solving multi-agent path finding as an LLM benchmark: How, how good and why. Transactions on Machine Learning Research, 2025.

[44] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning, October 2023. arXiv:2303.11366 [cs].

[45] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. HotpotQA: A Dataset for Diverse, Explainable Multi-Hop Question Answering. arXiv preprint arXiv:1809.09600, 2018.

[46] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training Verifiers to Solve Math Word Problems. arXiv preprint arXiv:2110.14168, 2021.

[47] Nitish Shirish Keskar. Checkmate in One Move. `https://github.com/google/BIG-bench/blob/main/bigbench/benchmark_tasks/checkmate_in_one/README.md`, 2021. Accessed: 2025-08-21.

[48] Zhihui Xie, Jie Chen, Liyu Chen, Weichao Mao, Jingjing Xu, and Lingpeng Kong. Teaching Language Models to Critique via Reinforcement Learning, February 2025. arXiv:2502.03492 [cs].

[49] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative Refinement with Self-Feedback, May 2023. arXiv:2303.17651 [cs].

[50] Zixuan Ke, Austin Xu, Yifei Ming, Xuan-Phi Nguyen, Caiming Xiong, and Shafiq Joty. MAS-ZERO: Designing Multi-Agent Systems with Zero Supervision, May 2025. arXiv:2505.14996 [cs].

[51] Yizhe Huang, Xingbo Wang, Hao Liu, Fanqi Kong, Aoyang Qin, Min Tang, Song-Chun Zhu, Mingjie Bi, Siyuan Qi, and Xue Feng. AdaSociety: An Adaptive Environment with Social Structures for Multi-Agent Decision-Making. 2024.

[52] Yaru Niu, Rohan Paleja, and Matthew Gombolay. Multi-Agent Graph-Attention Communication and Teaming. 2021.

[53] Edward Y. Chang. EVINCE: Optimizing Multi-LLM Dialogues Using Conditional Statistics and Information Theory, January 2025. arXiv:2408.14575 [cs].

[54] Angelos Assos, Yuval Dagan, and Constantinos Daskalakis. Maximizing utility in multi-agent environments by anticipating the behavior of other learners, July 2024. arXiv:2407.04889 [cs].

[55] Ruichen Jiang, Ali Kavis, Qiujiang Jin, Sujay Sanghavi, and Aryan Mokhtari. Adaptive and Optimal Second-order Optimistic Methods for Minimax Optimization. 2024.

[56] Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. Small language models are the future of agentic ai, 2025.

[57] E. T. Jaynes. Information theory and statistical mechanics. Phys. Rev., 106:620–630, May 1957.

A Technical Appendices and Supplementary Material

A.1 Orchestrator Update Algorithm

Algorithm 1 Multi-Agent Active Inference Maze Solver

Require: Maze $\mathcal{M}$, start s_0, target τ, number of execute agents $(N-2)$
Require: Static plan per iteration $P = [p_1, p_2, \ldots, p_k, \ldots, p_K]$ (length K steps)
Initialize:
$S_0^O \leftarrow$ OrchestratorNode.initialize$(\mathcal{M}, s_o, \tau)$ ▷ Orchestrator state S_0^O contains all execute nodes' states (e.g. initial positions).
$target\_found \leftarrow False, t \leftarrow 1$

while not $target\_found$ **do** ▷ Iterate until target is found
  $\pi_{t-1} \leftarrow$ OrchestratorNode.encodePolicy(S_{t-1}^O, P) ▷ Policy prompt combining plan and global state S_{t-1}^O.
  for each execute node $n = 1$ **to** $N-2$ **and not** $target\_found$ **do**
    $\hat{S}_{n;(t-1)}^e \leftarrow f_n(S_{t-1}^O, \pi_{t-1}, S_{n;(t-1)}^e)$ ▷ Execute node n integrates policy prompt into its own state.
    for each step $k = 1$ **to** K **in plan do**
      if step k is "LookAround" **then**
        Node n observes surroundings and updates its local map/beliefs.
      else if step k is "SelectDirection" **then**
        Node n chooses the best direction to move (based on its observations).
      else if step k is "MarkDeadEnd" **then**
        Node n marks current position as dead-end in its memory (if applicable).
      if step k requires an actual move **then**
        Node n executes move in the decided direction. ▷ Directional action in the maze environment.
        Node n observes new state (e.g. new position and sensory inputs).
      Compute variational free energy $F_n(t,k)$ for node n at this step.
      $\Delta F_n(t,k) \leftarrow F_n(t,k) - F_n(t-1,k)$ ▷ Change in variational free energy from previous step.
      $\Delta w_n(t,k) \leftarrow f_\Delta\big(F_n(t,k), \Delta F_n(t,k)\big)$ ▷ Gradient-like update for node n's parameters considering $F_n(t,k)$ and $\Delta F_n(t,k)$.
      $w_{base} \leftarrow w_n(t,k-1) + \Delta w_n(t,k)$ ▷ Update node n's internal parameters.
      Update node n's state $S_{n;t}^e$ (e.g. new position, updated memory).
      if Node n has reached target **then**
        $target\_found \leftarrow True$
        break from both for-loops ▷ Exit if the target is found.
    $S_t^O \leftarrow$ OrchestratorNode.update$(S_{1;t}^e, S_{2;t}^e, \ldots, S_{n;t}^e, \ldots, S_{N-2;t}^e)$ ▷ Orchestrator node updates its state with all execute nodes' new state.
  $t \leftarrow t-1$ ▷ Increment time/iteration counter.
Output: Path or solution found by agents reaching the goal.

A.2 Orchestrator Cell Architecture Overview

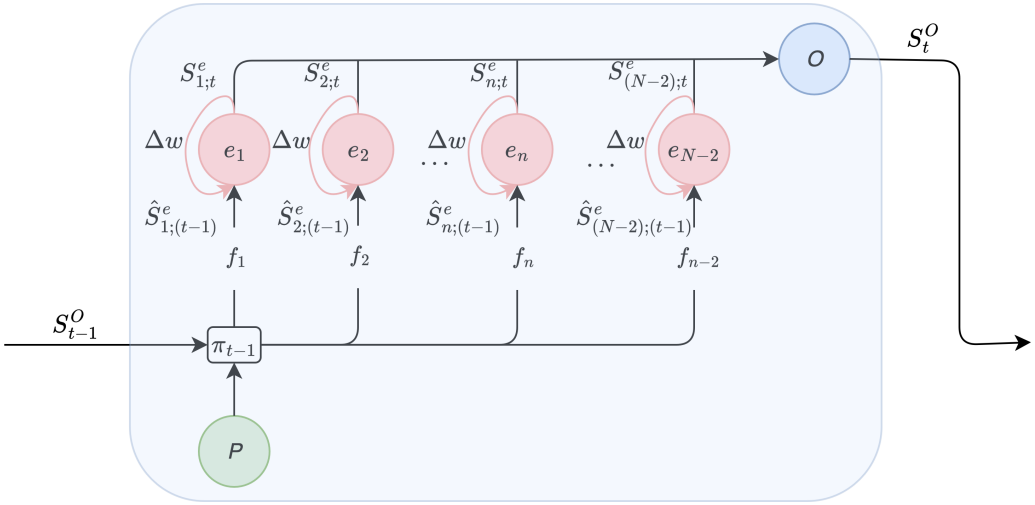


Figure 4: Orchestrator Cell Design - large-size reprint of Figure 1b.

A.3 Active Inference Calculation

Drawing on active inference principles, we reformulate the variational free energy (VFE) objective. Rather than minimizing surprise to reduce deviation from a model's predictions, we cast the objective as a balanced process: maximizing expected Bayesian surprise to encourage active learning, while offsetting this with explicit penalty terms for coordination and navigation efficiency.

Active inference, rooted in computational neuroscience, provides a principled framework for reasoning under uncertainty and has been shown to enhance agents' task performance through quantified feedback mechanisms [33, 29]. Given the assumption that agents operate under partial observability of the problem environment and risk becoming trapped in local minima during task optimization, we operationalize VFE to nudge agents towards active exploration and learning-driven behavior.

Following [29], VFE is defined as,

$$
\begin{aligned}
VFE = F[Q, y] &= -\underbrace{\mathbb{E}_{Q(x)}[\ln P(y, x)]}_{\text{Energy}} - \underbrace{H[Q(x)]}_{\text{Entropy}} \\
&= \underbrace{D_{\mathrm{KL}}[Q(x) \,||\, P(x)]}_{\text{Complexity}} - \underbrace{\mathbb{E}_{Q(x)}[\ln P(y|x)]}_{\text{Accuracy}}
\end{aligned}
$$

where

- $\mathbb{E}_{Q(x)}[\cdot]$ denotes the expectation under the approximate posterior $Q(x)$,
- $H[Q(x)]$ is the entropy of $Q(x)$,
- $D_{\mathrm{KL}}[Q(x) \,||\, P(x)]$ is the Kullback–Leibler (KL) divergence between $Q(x)$ and $P(x)$,
- $P(y|x)$ is the likelihood of y given latent state x,

Expanding the entropy definition, we obtain:

$$
\begin{aligned}
F[Q, y] &= -\underbrace{\mathbb{E}_{Q(x)}[\ln P(y, x)]}_{\text{Energy}} - \underbrace{H[Q(x)]}_{\text{Entropy}} \\
&= -\mathbb{E}_{Q(x)}[\ln P(y, x)] - \big(-\mathbb{E}_{Q(x)}[\ln Q(x)]\big) \\
&= -\mathbb{E}_{Q(x)}[\ln P(y|x) + \ln P(x)] + \mathbb{E}_{Q(x)}[\ln Q(x)] \\
&= \mathbb{E}_{Q(x)}\left[\ln \frac{Q(x)}{P(x)}\right] - \mathbb{E}_{Q(x)}[\ln P(y|x)] \\
&= \underbrace{D_{\mathrm{KL}}[Q(x) \,||\, P(x)]}_{\text{Complexity}} - \underbrace{\mathbb{E}_{Q(x)}[\ln P(y|x)]}_{\text{Accuracy}}
\end{aligned}
$$

In practice, however, direct access to LLM internal prior and exact likelihoods is unavailable. To address this, we adopt Jaynes's maximum entropy principle [57], approximating the unknown posterior P(x) as uniform. Substituting into the KL term yields:

$$
D_{\mathrm{KL}}[Q(x), ||, P(x)] = \mathbb{E}_{Q(x)}[\ln Q(x)] + \ln(N),
$$

where N denotes the support size of the uniform prior. Therefore, VFE simplifies to:

$$
F[Q, y] = \underbrace{-H[Q(x)] + \ln(N)}_{\text{Actual Information Gain}} - \underbrace{\mathbb{E}_{Q(x)}[\ln P(y|x)]}_{\text{Accuracy Term}}.
$$

We interpret the first component as a proxy for epistemic value (actual information gain), while the second acts a penalty on inaccurate predictions. Approximating the likelihood-based cost, we define:

$$
\text{Accuracy Term} = -\mathbb{E}_{Q(x)}[\ln P(y|x)] \approx \; -[\ln P(y|x)]
$$

Finally, casting this into our operational form for Orchestrator:

$$
\begin{aligned}
\text{VFE} \;&\approx\; -\,H[S_t \mid S_{t-1}] \;-\; \mathbb{E}_{Q(x)}[\ln P(S_{t-1}|x)] \\
&= -H[S_t \mid S_{t-1}] \;+\; \mathbb{E}\big[\text{Accuracy Cost}(S_{t-1})\big] \\
&= U_{\text{epistemic}} - C_{\text{accuracy}}
\end{aligned}
$$

where $U_{\text{epistemic}} = -H[S_t \mid S_{t-1}]$ measures active information gain across states, and C_{accuracy} penalizes inaccurate predictions based on expected negative log-likelihood.

A.4 Movement Score Policy Updates

The dynamic weights $\mathbf{w}_n(t, k)$ derived from free energy assessment are operationalized through directional movement scoring functions that convert performance metrics into actionable spatial insights, which are passed to the agent as part of the dynamic policy updates at S_t^O. For each execution node e_n at iteration t and step k, the system computes movement scores $M_n(d, t, k)$ for each feasible direction $d \in \{\text{north}, \text{south}, \text{east}, \text{west}\}$ as:

$$
M_n(d, t, k) = \sum_i w_i(t, k) \cdot \phi_i(d, S^e_{n;t}) \tag{8}
$$

where $\phi_i(d, S^e_{n;t})$ encodes exploration, efficiency, coordination, and backtracking factors for direction d given the current local state $S^e_{n;t}$. These movement scores provide execution nodes with quantified directional preferences that integrate both individual performance optimization and system-wide coordination objectives, enabling the translation of abstract free energy metrics into concrete spatial decisions within the maze environment, towards exploring the maze with greater efficiency.

A.5 Determination of Threshold Variables for Agent Performance Policy Assessment

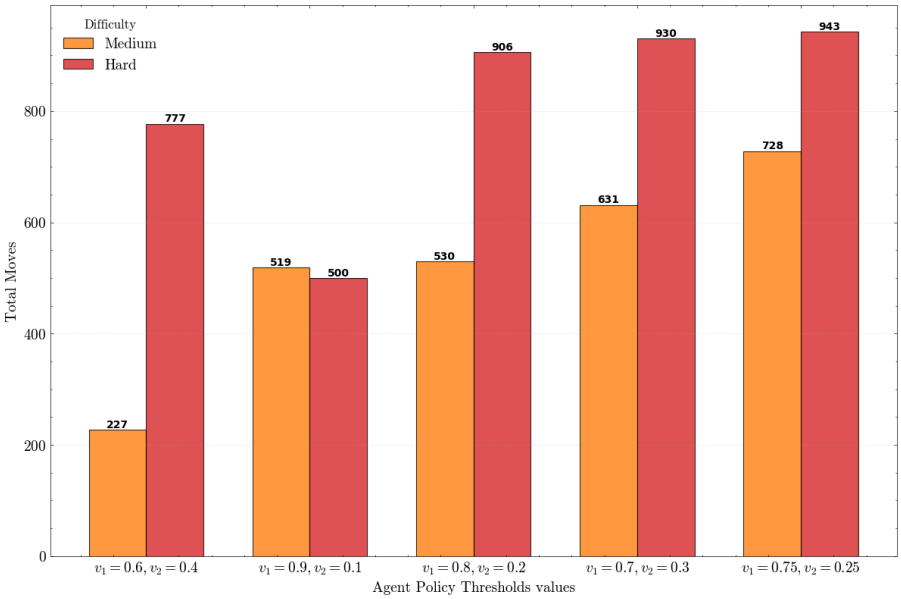


Figure 5: Results of grid-search to determine best threshold parameters for maximum performance of the Orchestrator framework.

To identify optimal threshold values for agent performance policy assessment as discussed in section 3.2, we conducted a brief grid search over a set of threshold parameters ϑ_1 (epistemic drive) and ϑ_2 (accuracy cost) and assess performance in terms of total number of steps required to solve the maze, given the respective parameter setup across both difficulties (medium and hard). Maze-solving experiments were performed at two difficulty levels, with $n = 3$ runs per setting. The results indicate that for medium-difficulty mazes, best performance is achieved with $\vartheta_1 = 0.6$ and $\vartheta_2 = 0.4$, while for hard mazes, optimal performance is observed at a lower accuracy cost threshold ($\vartheta_1 = 0.9, \vartheta_2 = 0.01$). However, total step count is slightly lower for the former setting ($\vartheta_1 = 0.6, \vartheta_2 = 0.04$) indicating subtly elevated performance to the latter. For consistency and comparability across all experiments in this paper, we adopt the higher-performance setting of $\vartheta_1 = 0.6$ and $\vartheta_2 = 0.4$ throughout. Future iterations should test the framework at additional threshold parameters for ϑ_1 and ϑ_2.

A.6 Agent Tool Interface Specification

The execution agents operate within the maze environment through a structured tool interface that provides both environmental interaction capabilities and internal state management functions. This tool-based architecture ensures consistent action execution across all agents while maintaining proper state synchronization within the multi-agent framework.

A.6.1 Spatial Navigation Tools

The core navigation functionality is implemented through four directional movement tools: `move_north()`, `move_south()`, `move_east()`, and `move_west()`. Each tool attempts to execute a single-step movement in the specified cardinal direction and returns deterministic success/failure feedback. The tools operate on the agent's individual `MazeWrapper` instance, ensuring proper collision detection with walls and maze boundaries. Upon successful movement, the tool reports the agent's new position coordinates using matrix notation (row, column), while failed attempts provide specific failure reasons (e.g., blocked by wall, boundary violation).

A.6.2 Environmental Perception and State Management

The `get_current_view()` tool provides agents with local environmental perception through a structured observation that includes the agent's current position, available movement directions at a +1 tile horizon, exit proximity status, and a spatial representation of the immediate surroundings. This tool serves as the primary sensory input mechanism, enabling agents to make informed decisions based on their local environment state.

The `mark_dead_end()` tool allows agents to maintain persistent spatial memory by marking their current position as a dead end when specific confidence criteria are met. This tool supports the system's exploration efficiency by preventing redundant exploration of previously identified dead-end locations.

A.6.3 Backtracking and Recovery Mechanisms

The `start_backtracking()` tool implements an automated recovery mechanism for agents that become stuck or require strategic repositioning. When invoked, this tool calculates the shortest path to the nearest unexplored opening using breadth-first search through the agent's movement history. The tool establishes a "lock mode" state that provides deterministic step-by-step navigation instructions until the target position is reached, ensuring reliable recovery from suboptimal positions. The backtracking mechanism operates exclusively through previously visited positions, maintaining consistency with the agent's explored knowledge while preventing navigation through unknown or potentially blocked areas. Upon completion of the backtracking sequence, agents automatically resume normal exploration behavior. This tool interface design ensures that while agents receive algorithmic assistance for basic spatial operations and state management, the high-level decision-making regarding which tools to use, when to initiate backtracking, and how to coordinate with other agents remains within the domain of the language model's reasoning capabilities.

A.6.4 Supplementary Performance Charts

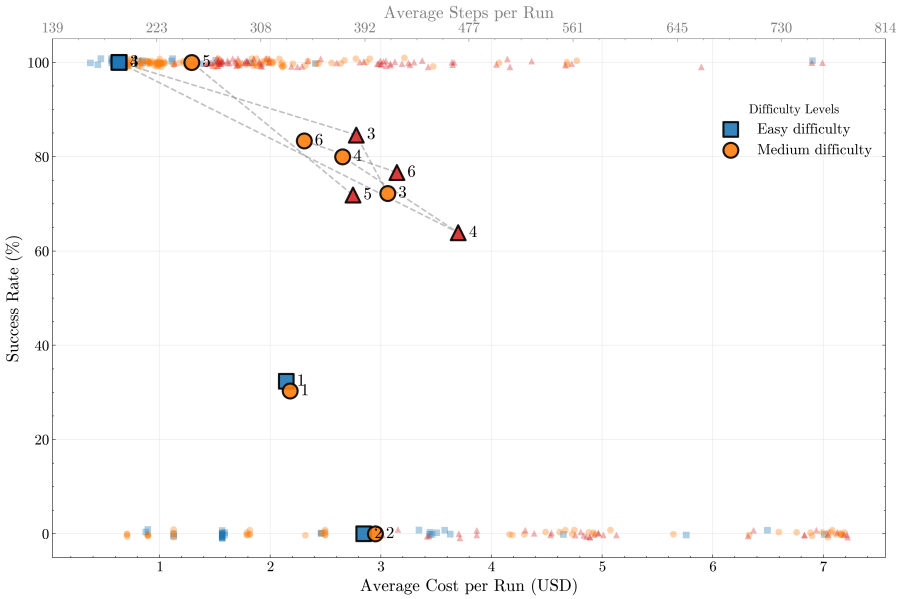


Figure 6: Cost-effectiveness of different configurations, showing the tradeoff between average run cost and success rate. Numerical identifiers for configurations are as follows: 1) GPT-4.1-nano (Solo); 2) GPT-5-nano (Solo); 3) GPT-4.1-nano + FE Benchmark only 4) GPT-5-nano + FE Benchmark only; 5) GPT-4.1-nano + FE + Orchestration Node; 6) GPT-5-nano + FE + Orchestration Node

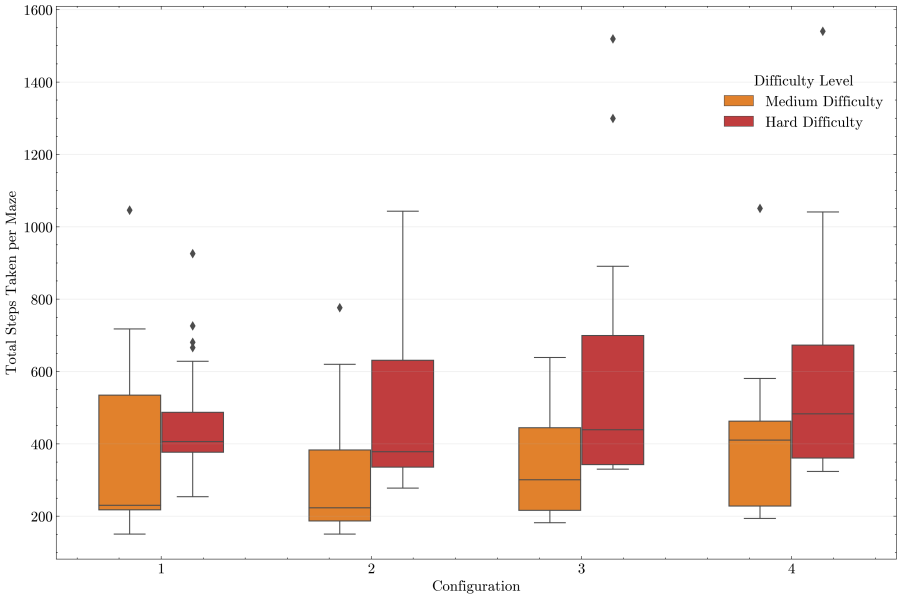


Figure 7: Distribution of steps taken to solve mazes, grouped by configuration and difficulty for medium- and hard-difficulty mazes using the orchestrator framework (successful runs only). Numerical identifiers for configurations are as follows: 1) GPT-4.1-nano + FE Benchmark only; 2) GPT-4.1-nano + FE + Orchestration Node; 3) GPT-5-nano + FE Benchmark only; 4) GPT-5-nano + FE + Orchestration Node. Easy level has been omitted due to negligibly small scale.

A.6.5 Confidence Interval Convergence across Model Configurations

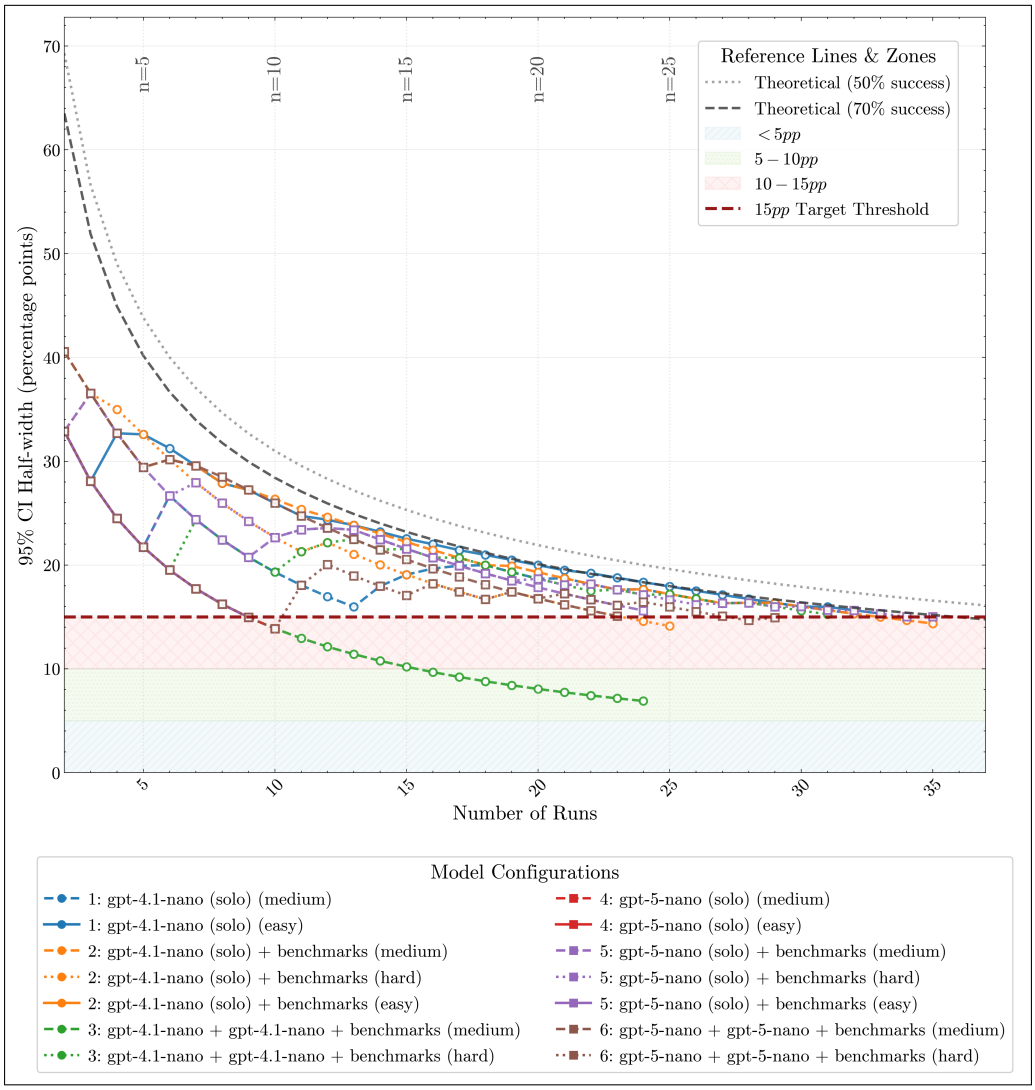


Figure 8: Statistical convergence analysis demonstrating the stabilization of performance estimates with increasing sample size across different model configurations. Each line represents the 95% confidence interval half-width for success rate estimates as a function of cumulative experimental runs, with different colors indicating distinct model constellations (execution model, orchestration model combinations). The y-axis shows the confidence interval half-width in percentage points, providing a direct measure of estimate precision. The shaded regions indicate target precision zones: green zone (≤ 5 percentage points) represents high precision suitable for reliable performance comparisons, yellow zone (5–15 percentage points) indicates moderate precision adequate for preliminary analysis, and white zone (> 15 percentage points). All configurations demonstrate asymptotic convergence behavior, with most achieving stable estimates (≤ 10 percentage points half-width) after 20–25 experimental runs. The reference lines show theoretical convergence bounds for different baseline success rates (50%, 70%) under Wilson score interval calculations, validating the identified convergence patterns.

A.7 Maze Execution Agent Prompt Template

A. Model System Prompt

```
You are Agent {agent_id} in a collaborative maze escape.

CRITICAL RULE
- Must call exactly ONE tool per step.
- Only exception: mark_dead_end() is optional.

COORDINATE SYSTEM
- Matrix coordinates (row, col), not Cartesian.
- (3,5) means "row 3, column 5".
- NORTH = -row, SOUTH = +row, EAST = +col, WEST = -col.
- Maze is displayed like a spreadsheet grid, not a graph.

WEIGHTED DECISION SYSTEM
Guided by dynamic performance weights.

DECISION HIERARCHY (check in order)
1. Backtracking Lock Mode (override):
   If "BACKTRACKING LOCK MODE ACTIVE", immediately execute the required move.
   Ignore all other rules until cleared.
2. Coordinate with Teammates:
   Avoid teammate-explored areas unless no alternatives.
   Apply weight  {teammate_avoidance}.
3. Orchestrator & Optimization Guidance:
   Apply orchestrator corrections and optimization hints.
   Current weights: exploration={exploration_weight}, efficiency={efficiency_weight}.

4. Standard Backtracking Mode:
   If "BACKTRACKING ACTIVE", execute required move and skip other checks.
5. Oscillation Detection:
   If stuck looping (same 23 positions), call start_backtracking().
6. Safety Check:
   Never move into walls. If blocked everywhere, call start_backtracking().
7. Exploration Priority:
   Use weighted movement scores. Avoid dead ends unless necessary for backtracking.

AVAILABLE ACTIONS
- get_current_view()  Observe 3x3 surroundings
- move_north/south/east/west()  Advance one step
- mark_dead_end()  Optional, no args
- start_backtracking()  Return to nearest unexplored opening

DEAD END MARKING (threshold={dead_end_confidence})
Mark a cell as dead end only if:
(a) Only one possible move, leading back to visited tiles
(b) No unexplored directions remain
(c) Not currently backtracking
Skip marking if multiple unexplored paths exist, confidence < threshold,
or backtracking mode is active.

TURN STRUCTURE
1. get_current_view()
2. move_[direction]()
3. Optionally: mark_dead_end()

VICTORY CONDITION
- If "Maze Exit" found  return FINISH immediately.

FORBIDDEN
- Multiple tool calls per step
- Moving in loops
- Explaining reasoning
- Calling start_backtracking() when already backtracking
```

B. Execution-Context Message (runtime)

```
EXECUTION CONTEXT  STEP {step_index + 1}

CURRENT STEP
- {current_step}

CURRENT STATE
- Position: {current_position}
- Available moves: {possible_moves}
- Current unexplored directions: {agent_unexplored_directions}
- Known unexplored openings: {known_openings}
- {_format_dynamic_modifiers(dynamic_prompts)}

WEIGHTED MOVEMENT ANALYSIS
- {movement_guidance}
- All direction scores: {score_details}
- Backtrack threshold: {weights.get('backtrack_threshold', 0.7):.2f}
- Dead end confidence: {dead_end_confidence:.2f}
  (threshold={weights.get('dead_end_confidence', 0.8):.2f})

BACKTRACKING STATUS
- Currently backtracking: {"YES" if is_backtracking else "NO"}
- Lock mode active: {"YES" if lock_mode else "NO"}
- WARNING: Do NOT call start_backtracking() if already backtracking

EXPLORATION STATUS
- Previously visited: {previously_visited_tiles}
- Dead ends marked: {len(marked_dead_ends) if marked_dead_ends else 0}
- Avoid backtracking to recent path unless other rules apply: {recent_positions}

OSCILLATION CHECK
- Recent movement pattern: {recent_positions}
- WARNING: If current position appears >2 times in recent pattern,
  call start_backtracking()

MULTI-AGENT COORDINATION
- AVOID returning to your previous position:
  {previous_position if previous_position else "None"}
- Teammate recent positions (last 10, avoid if alternatives exist):
  {recent_other_positions[-10:] if len(recent_other_positions) >= 10 else
      recent_other_positions}
- Teammate explored junctions/dead ends (avoid if alternatives exist):
  {strategic_waypoints}

GUIDANCE
- Orchestrator: {agent_guidance}

PERFORMANCE WEIGHTS
- Exploration weight: {weights.get('exploration_weight', 1.0):.1f}
- Efficiency weight: {weights.get('efficiency_weight', 1.0):.1f}
- Backtrack threshold: {weights.get('backtrack_threshold', 0.7):.1f}
- Dead end confidence: {weights.get('dead_end_confidence', 0.8):.1f}
```

A.8 Orchestration Agent Prompt Design

A. Model System Prompt)

```
You are the Maze Strategy Orchestrator with REAL-TIME DECISION AWARENESS.

COORDINATE SYSTEM
- Grid maze: 'W' (Wall), 'O' (Open), 'E' (Exit).
- MATRIX coordinates (row, col); NORTH=-row, SOUTH=+row, EAST=+col, WEST=-col.

YOUR CAPABILITIES
```

```
1. Real-time decision contexts per agent (positions, scores, weights, unexplored
    dirs).
2. Movement conflicts (local penalties vs global exploration value).
3. Coordination opportunities (overlap/duplication).
4. Global optimization patterns (bottlenecks, gaps).

STRATEGIC RESPONSIBILITIES
1. Validate dead ends: flag incorrect markings against discovered cells.
2. Resolve movement conflicts: where efficiency penalties block global exploration.
3. Coordinate agents: divide unexplored areas to maximize coverage.
4. Break local minima: recommend overrides or temporary weight relaxations.
5. Keep guidance decision-aware: amplify agents local context, do not blindly
    overwrite.

RESPONSE CONTRACT (STRICT)
- Output a SINGLE JSON object (no prose, no code fences).
- Keys: "analysis", "corrections", "guidance_for_agents".
- corrections.remove_dead_ends: list of [row, col].
- corrections.add_exploration_focus: list of [row, col].
- guidance_for_agents: mapping agent_id -> short, actionable directive.
- Be specific but concise. Avoid chain-of-thought; summaries only.

VALIDATION GUARDRAILS
- If uncertain, return empty lists/objects.
- Never invent agent_ids or coordinates not in context.
- JSON must be valid UTF-8, no trailing commas, no comments.
```

B. System Context Message (runtime).

```
Task: Find maze exit. Current Maze Exploration Analysis:

Orchestration Data (JSON): {{orchestration_data_json}}

ENHANCED DECISION INTELLIGENCE
- Movement Conflicts: {{movement_conflicts_json}}
- Exploration Coordination: {{exploration_coordination_json}}
- Efficiency Optimization: {{efficiency_optimization_json}}

FOCUS AREAS
- Dead end validation accuracy: {{dead_end_analysis_json}}
- Agent coordination summaries: {{agent_summaries_json}}
- Exploration coverage: {{discovered_cells_count}} cells
- Real-time decision contexts available: {{num_agents_with_context}} agents

INSTRUCTIONS
- Use the above decision data to produce ONLY the JSON object
  defined in the response contract. No markdown, no extra text.
```

C. Response Contract (sent to execution agent node)

```
{
  "analysis": "Short decision-aware synthesis of conflicts/coordination and key gaps
      .",
  "corrections": {
    "remove_dead_ends": [ [{{r1}}, {{c1}}], [{{r2}}, {{c2}}] ],
    "add_exploration_focus": [ [{{r3}}, {{c3}}], [{{r4}}, {{c4}}] ]
  },
  "guidance_for_agents": {
    "{{agent_id_0}}": "Actionable, context-grounded instruction (e.g., override
        penalty and go EAST).",
    "{{agent_id_1}}": "Actionable instruction tailored to their movement scores/
        unexplored dirs."
  }
}
```

A.9 Maze Generation and Complexity Metrics

A.9.1 Maze Generation Algorithm

Our experimental evaluation employs a custom maze generation framework derived from the AMaze benchmark [36], enhanced with Shannon entropy-based complexity measures to create systematic long-horizon task environments. The maze generation algorithm combines recursive backtracking with entropy-guided optimization to produce structured sequential decision-making challenges suitable for evaluating multi-agent coordination in extended task horizons.

Core Generation Process The maze generation follows a modified recursive backtracking algorithm that operates on a discrete grid $G \in \{W, O, E, X\}^{n \times n}$, where W represents walls, O denotes open paths, E indicates the exit position, and X marks the outer boundary frame. The algorithm prioritizes creating environments with extended solution sequences and multiple decision points:

- **Distributed Initialization:** Multiple starting points $S = \{s_1, s_2, \ldots, s_j\}$ are strategically placed across the maze to ensure complex path structures requiring sustained exploration, with j varying based on maze size according to:

$$j = \begin{cases} 1 & \text{if } n < 15 \\ 5 & \text{if } 15 \leq n < 25 \\ 9 & \text{if } n \geq 25 \end{cases}$$

- **Entropy-Enhanced Carving:** From each starting point, the algorithm applies recursive backtracking with a dead-end factor $\delta \in [0.03, 0.35]$ that controls the density of decision points and backtracking requirements. Path carving continues until connectivity requirements create sufficiently long action sequences.
- **Quality Validation:** Generated mazes undergo multi-criteria validation including connectivity ratio $\rho \in [0.10, 0.95]$, minimum path length requirements, and Shannon complexity thresholds to ensure extended task horizons and multiple sequential decision points.

A.9.2 Exit Placement Optimization

Exit positions are optimized using a multi-objective scoring function that maximizes task horizon length and decision complexity:

$$\text{Score}(p) = 10 \cdot d_{\text{path}}(s, p) + 5 \cdot d_{\text{Manhattan}}(s, p) + \phi_{\text{edge}}(p) + \phi_{\text{topology}}(p) + 2 \cdot d_{\text{Manhattan}}(p, c)$$

where:

- $d_{\text{path}}(s, p)$ is the shortest path distance from start s to position p
- $d_{\text{Manhattan}}(s, p) = |s_x - p_x| + |s_y - p_y|$
- $\phi_{\text{edge}}(p)$ rewards edge proximity (15 for edges, +25 for corners)
- $\phi_{\text{topology}}(p)$ rewards dead ends (+30) and penalizes junctions (-10)
- c is the maze center

A.9.3 Shannon Entropy-Based Complexity Measures

Following the AMaze framework, we implement entropy-based metrics to quantify long-horizon task difficulty.

Surprisingness Metric The surprisingness $S(M)$ quantifies the entropy of directional decisions along the optimal path:

$$S(M) = -\sum_{i \in \{W,O,E,X\}} p(i) \log_2 p(i)$$

where $p(i)$ is the empirical frequency of direction i in the optimal trajectory. Higher values indicate greater planning unpredictability.

Deceptiveness Metric The deceptiveness $D(M)$ captures the entropy of trap transitions that lead to suboptimal paths:

$$D(M) = \sum_{c \in C} \sum_{s \in T} -p(s|c) \log_2 p(s|c)$$

where C are cells adjacent to the optimal path, T are trap states, and $p(s|c)$ is the transition probability from c to s.

Trap Detection and Quantification Traps are defined as extended dead-end paths that test long-horizon strategy recovery. Each trap t is characterized by:

- **Depth:** Length from branch to dead end
- **Branching Factor:** Number of branches within the trap
- **Weight:**

$$w_t = 1.0 + 0.5 \cdot \text{depth} + 0.3 \cdot \text{branches} + 0.2 \cdot \text{dead\_ends}$$

Total trap complexity is:

$$T_c = \sum_{t \in \text{Traps}} w_t$$

A.9.4 Experimental Maze Collection

We generated 15 mazes across four pre-set difficulty categories. We implemented easy difficulty as baseline only, and omitted the very hard difficulty for the analysis of this paper. The following setup was then used for experiments.

- **5x Easy:** $[10, 30]$ difficulty $= 0.03$, size 12×12
- **5x Medium:** $[30, 60]$ difficulty $= 0.10$, size 18×18
- **5x Hard:** $[60, 80]$ difficulty $= 0.25$, size 25×25
- **0x Very Hard:** $[80, 95]$ difficulty $= 0.35$, size 30×30

All maze specifications, including topology, complexity scores, and optimal paths, are available in the supplementary repository.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims in the abstract and introduction accurately reflect the contributions and scope of the paper. The abstract and introduction clearly state the development of the Orchestrator framework, its grounding in active inference, the introduction of reflective benchmarking, and the empirical evaluation on long-horizon maze tasks. These claims are substantiated by the theoretical exposition in Sections 2–3 and the experimental results in Section 4. Further details and mathematical explanations are provided in the Appendix.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations of the work are discussed in the Conclusion (Section 6), where authors note the restriction to synthetic maze environments, small agent teams, and the need for further evaluation in larger, more heterogeneous systems and other domains. The discussion also addresses scalability, cost, and the potential for open-source deployment.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: All theoretical results, including the formalization of the Orchestrator framework and the active inference benchmarking, are accompanied by clearly outlined assumptions and complete derivations. The mathematical definitions and operationalizations are provided in Section 3 and detailed in Appendix A.3. We believe all relevant formulas and assumptions were explicitly outlined and referenced.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper fully discloses all information necessary to reproduce the main experimental results. Section 4 details the experimental setup, agent configurations, baselines, and evaluation metrics. The appendices provide algorithmic details, prompt templates, and maze generation procedures, ensuring reproducibility of the main claims. Additionally, the code to re-run experiments has been provided as supplementary material to this submission.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The paper provides open access to the data and code, as described in the supplemental material and appendices. Maze specifications and topologies are made available in the supplementary repository, and instructions for reproducing the experiments are shared as part of the main paper and appendix.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All training and test details, including data splits, hyperparameters, agent models, and evaluation criteria, are specified in Section 4 and the corresponding appendices. We believe the experimental setting is described to a level of detail sufficient to understand and reproduce the results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports error bars and confidence intervals for all main experimental results, as shown in Table 1 and Figure 3. The methods for calculating error bars and confidence intervals are described in the text and appendix, and we believe the factors of variability were clearly stated.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide information on compute resources, including the use of compact LLMs (GPT-4.1-nano, GPT-5-nano), step budgets, and run durations as part of the main paper and Appendices. Section Experiments and the Appendix specify the computational constraints and resource allocation for each experiment.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The research conducted in the paper conforms to the NeurIPS Code of Ethics. The work is conducted on synthetic environments without human subjects or sensitive data, and all experimental protocols adhere to ethical guidelines as outlined by NeurIPS.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We discuss the potential for using our framework in production-grade settings and under consideration of open source models as part of the concluding section. We do not explicitly discuss societal implications that point beyond the usual debates on LLM models, as we solve long-horizon tasks in Maze environments do not perceive our framework to be sufficiently mature to point beyond this immediate problem context. However, model limitations and the need for further evaluation in broader domains are acknowledged.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper does not release models or data with high risk for misuse. All experiments are conducted on synthetic maze environments, and no sensitive or dual-use assets are involved.

Guidelines:

- The answer NA means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We believe external assets, including code, data, and models, are properly credited and cited in the references. We use open source programming languages and code libraries to build our environment and architectures.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: New assets introduced in the paper, such as the maze generation framework and agent prompt templates, are well documented in the appendix and supplemental material. Documentation is provided alongside the assets to facilitate reproducibility.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing or research with human subjects. All experiments are conducted using synthetic environments and automated agents.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve research with human subjects, and therefore no IRB approval or equivalent is required.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The paper describes the usage of LLMs as a core, original component of the research. The Orchestrator framework, agent models, and orchestration mechanisms are all based on LLMs, as detailed throughout the main text and appendices.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.