Economy of Minds: Emerging Multi-Agent Intelligence with Economic Interactions

Zhenting Qi[Harvard,*] **Huangyuan Su**[Harvard,Kempner] **Ao Qu**[MIT] **Chenyu Wang**[Harvard]
Yu Yao[MIT] **Han Zheng**[MIT] **Kushal Chattopadhyay**[Harvard] **Guowei Xu**[Harvard]
Zihan Wang[2077AI] **Weirui Ye**[MIT] **Vijay Janapa Reddi**[Harvard] **Ju Li**[MIT] **Paul Pu Liang**[MIT]
Himabindu Lakkaraju[Harvard] **Sham Kakade**[Harvard,Kempner] **Yilun Du**[Harvard,Kempner,*]

Harvard MIT 2077AI Kempner Institute

*Corresponding to: zhentingqi@g.harvard.edu, ydu@seas.harvard.edu

GitHub Project Page

Abstract

How can a population of agents self-orchestrate and self-adapt into stronger collective intelligence without centralized control? Inspired by Friedrich Hayek's economic theory of decentralized coordination in markets, we study this question through an *agent economy* in which agents compete via auctions for the right to act, exchange payments, and accumulate wealth from environmental rewards. These simple local incentives induce decentralized credit assignment, driving global planning without explicit orchestration or communication protocols. The population evolves through economic selection: effective agents accumulate wealth and are mutated via exploitation, while ineffective ones go bankrupt and are replaced via exploration. Using modern Large Language Model (LLM) based agents, we show that, initialized with weak agents, the agent economy produces emergent multi-step reasoning strategies and outperforms stronger monolithic baselines across five agentic tasks, including mathematical reasoning, financial research, scientific research, accelerator design, and distributed-system optimization. Our results suggest a new path to multi-agent intelligence: rather than engineering coordination, we can design decentralized incentive structures under which it automatically emerges.

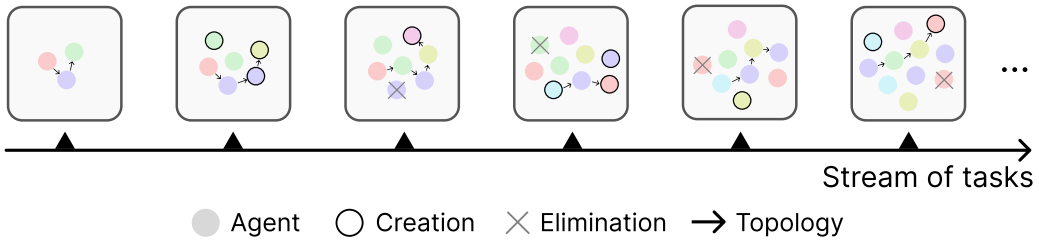


Figure 1: **Evolution of an agent society over a stream of tasks.** Each panel shows the population at a given stage, where agents are continuously created, selected, connected, and eliminated. As the society encounters more tasks, ineffective agents are removed and corrected, while useful ones persist and diversify, leading to an alive and increasingly structured population.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Imagine a world populated by a number of intelligent agents. Each agent may perform well on certain tasks, but it remains fundamentally limited: each operates with its own priors, partial observations of environments, and bounded computational resources. When faced with complex tasks that exceed individual capabilities, no single agent can reliably solve problems from start to finish. How, then, can such a population collectively solve these tasks?

A natural approach is to introduce a central orchestrator that creates agents, assigns specializations, and coordinates actions across agents [19, 33, 47]. However, such centralized systems suffer from two fundamental limitations. First, planning is bottlenecked at a single coordination gate: all information and decision-making must flow through the orchestrator, creating both a performance bottleneck and a single point of failure [6, 50]. Second, learning and adaptation become increasingly inefficient as the system scales. As the number of agents grows, the orchestrator must reason about and manage an ever-expanding set of agents, leading to coordination complexity that grows linearly with system size [50, 22]. These limitations motivate a shift in perspective: rather than asking how to build better centralized controllers, we ask whether a group of agents can form decentralized intelligence, which organizes itself into strong collective capability and naturally evolves through time.

These questions are especially compelling today. Advances in large foundation models such as large language models (LLMs) and world action models make it possible to instantiate intelligent agents that can reason, use tools, and operate in open-ended digital or embodied environments [51, 31, 21, 52]. However, such agents are typically carefully engineered and tailored to specific domains, introducing inductive biases that might hinder general utility. Also, constraints in the underlying models, such as finite context length, partial knowledge, and restricted inference budget, combined with environmental limitations like partial observability and constrained tool access, make it difficult for single agents to solve complex, compositional tasks alone [29, 20, 40]. Therefore, beyond improving these individual agents, another axis to scale up agentic intelligence is to organize these inherently constrained agents into an effective learnable system as a whole [7, 49, 34].

The economy in human society provides an intuition for such decentralized intelligence. In his seminal essay *The Use of Knowledge in Society* [14], Friedrich Hayek argued that the core problem of an economy is not optimization under known information, but the utilization of knowledge that is dispersed across individuals, which cannot be aggregated by any central authority. Hayek's key insight is that the free market solves this problem through the price system: prices serve as signals that aggregate and communicate dispersed information, enabling individuals to coordinate their actions without global awareness. As a result, large-scale social orders emerge from decentralized interactions through competition, specialization, and exchange. Building on this view, Baum further argued that economic organization provides a concrete model of intelligence, even though individual routines are simple [1, 2]. His Hayek machine shows that economic pressure serves as a natural mechanism for credit assignment, determining which routines act, how they are connected, and how they are renewed without centralized evaluation.

In this work, we show that simple economic incentives are sufficient for modern intelligent agents to self-orchestrate and self-evolve as a society. We present **Economy of Minds (EoM)**, a system where a population of agents compete via auctions for the right to act, exchange payments through peer-to-peer transactions, and accumulate or lose wealth based on an outcome reward. Agents that consistently contribute to successful trajectories naturally accumulate wealth, thus they survive and are periodically mutated through exploitation, while ineffective agents are eliminated through economic selection and replaced through exploration. Importantly, each agent operates purely locally: it only requires (1) a wake-up condition, and (2) executes its action accordingly. There is no need for global awareness, explicit coordination, engineered topology, or prescribed communication protocols.

We implement this framework using LLM-based agents and evaluate it on five digital agentic tasks, where EoM improves mathematical reasoning from 15.9% to 57.0%, raises financial research performance from 45.0% to 60.0%, increases scientific research accuracy over the baseline from 5.0% to 20.0%, reduces accelerator-design average energy-delay product to 39.3 versus 80.2 for a strong domain-specific method, and attains a best distributed-system optimization cost of 657 versus 930 for the baseline. Across these domains, we show with detailed cases that the agent society gradually self-organizes into effective workflows: competent agents persist, mutate, and specialize, while weaker agents are pruned and refined, yielding emergent structure and continual adaptation.

Our findings suggest that economic organization provides a simple, general, and scalable foundation for decentralized multi-agent intelligence. Rather than explicitly designing coordination mechanisms, we can define an incentive structure under which coordination, specialization, and cooperation naturally emerge between agents. This points toward a broader paradigm for multi-agent systems in the era of large foundation models: not as centrally engineered pipelines, but as evolving agent societies whose collective intelligence is shaped by the economies they inhabit.

2 An Economy of Language Agents

We model a society of language agents interacting through an economic mechanism. Each agent operates locally, making decisions based only on its own triggering condition and policy, while global coordination emerges from economic interactions. The system consists of two coupled processes: (1) *planning*, which governs how agents act and assigns credits to them within an episode, and (2) *adaptation*, which governs how the population evolves across episodes.

2.1 Problem Setup

We consider a task environment modeled as a partially observed Markov decision process $\mathcal{E} = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \mu_0)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s' \mid s, a)$ is the transition kernel, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in (0, 1]$ is the discount factor, and μ_0 is the initial-state distribution. At step t, the system observes $o_t \in \mathcal{O}$; in fully observed settings, $o_t = s_t$.

Each agent is implemented by a language model parametrized by θ. Here we adopt a simplified setting, where we use a shared frozen model backbone for all agents, with diversity arising entirely through system prompts. Formally, an agent is a tuple $a = (\phi_a, \pi_a, b_a, W_a)$, where $\phi_a : \mathcal{O} \to \{0, 1\}$ is a *triggering predicate* that determines whether the agent is eligible to act, $\pi_a : \mathcal{O} \to \Delta(\mathcal{A})$ is its action policy, $b_a \in \mathbb{R}_{\geq 0}$ is a fixed bid associated with the agent, and $W_a \in \mathbb{R}$ is its current wealth. Both ϕ_a and π_a are instantiated by the same frozen LLM with agent-specific prompts $p_a = (p_a^{\text{trig}}, p_a^{\text{act}})$:

$$\phi_a(o) = \text{LLM}_\theta(p_a^{\text{trig}}; o) \in \{0, 1\}, \quad \pi_a(\cdot \mid o) = \text{LLM}_\theta(p_a^{\text{act}}; o) \in \Delta(\mathcal{A}).$$

Thus, an agent is fully specified by its prompt pair together with its fixed bid and current wealth.[1] At episode e, the active population is denoted by $\mathcal{P}_e$.

2.2 Planning with Auctions and Transactions

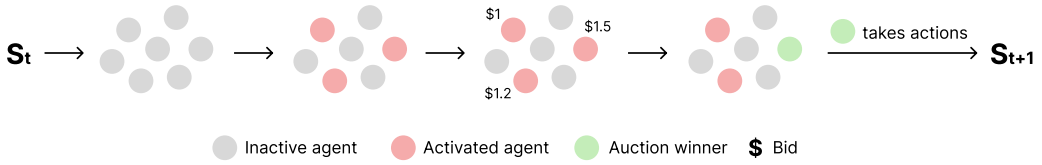


Figure 2: **Auctions.** Agents whose wake-up conditions are satisfied become eligible to bid; the highest bidder wins the auction, executes the action, and advances the environment from s_t to s_{t+1}.

At each environment step, agents compete for control through an auction (Figure 2). Given the current observation o_t, each agent evaluates its triggering predicate to determine eligibility. The eligible set is $E_t = \{a \in \mathcal{P}_e : \phi_a(o_t) = 1\}$. If $E_t = \emptyset$, no agent acts and the episode terminates or defaults to a null transition, depending on the environment. Otherwise, the auction winner is the highest-bidding eligible agent, $a_t^\star \in \arg\max_{a \in E_t} b_a$, with ties broken randomly.

The auction serves as a decentralized action-selection mechanism. Rather than relying on a centralized policy, control is allocated to the agent that bids the highest value to acting in the current context, as reflected by its bid. Once selected, the winning agent $a_t^\star$ samples an action from its policy $a_t^{\text{env}} \sim \pi_{a_t^\star}(\cdot \mid o_t)$, which advances the environment and produces the next observation o_{t+1} and

[1] This formulation generalizes Baum's Hayek machine [2, 3] from hand-specified condition-action rules to general agents: the condition is a model-based predicate, and the action is drawn from a model-based policy rather than being a fixed move.

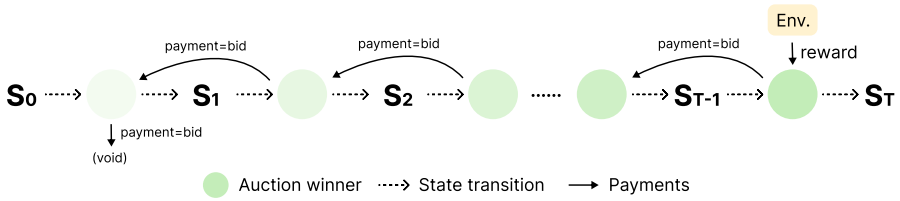


Figure 3: **Transactions.** Credit assignment naturally emerges as profits flow backward through the action sequence, rewarding agents whose actions enable successful downstream outcomes.

reward r_t. Let $a^\star_{t-1}$ denote the previous winning agent in the same episode. We then apply a bucket-brigade transfer rule (Figure 3):

$$W_{a^\star_t} \leftarrow W_{a^\star_t} - b_{a^\star_t} + r_t, \quad W_{a^\star_{t-1}} \leftarrow W_{a^\star_{t-1}} + b_{a^\star_t}. \tag{1}$$

That is, the winner pays its bid to the previously active agent while collecting any environmental reward r_t. For the first winner in an episode, the payment is made to the house.

This payment rule yields a decentralized form of credit assignment. An agent profits not only by directly receiving reward, but also by taking actions that place the system in states from which downstream agents are willing to pay highly for control. As a result, value flows backward along successful trajectories: agents whose actions enable productive continuations accumulate wealth, while agents that lead the system into unproductive states lose it.

2.3 Adaptation with Exploration and Exploitation

Beyond within-episode planning, the population evolves across episodes through economic selection. Let $\mathcal{G}$ denote a prompt-generation operator that is attached to the agents and proposes new agent prompts from the current one, for example, by mutating a successful agent's or amending a failed one's system prompt. Each newly created agent is initialized with wealth $W_0 \geq 0$, while existing agents may also incur a periodic rent $\rho \geq 0$. Examples of adaptations are provided in Appendix E and F, and theoretical motivations are stated in Appendix C.

Exploitation. Agents that consistently contribute to successful trajectories automatically accumulate wealth and persist in the population. Wealthy agents are periodically selected as parents and are mutated to produce new agents, allowing their successful patterns to be reused and refined. This mechanism biases the population toward high-performing behaviors, reinforces effective strategies, and promotes the emergence of specialized roles. Concretely, wealthy agents use their prompt generators $\mathcal{G}$ to propose prompt mutations that preserve their useful wake-up conditions or action policies while introducing small behavioral variations.

Exploration. Agents lose wealth through unhelpful actions or prolonged inactivity. When their wealth becomes negative, they are removed from the population, and new agents are introduced to replace them, typically by random or complementary variation of these bankrupt agents. This continual turnover enables learning from failures, discoveries of new behaviors, and prevents premature convergence. In this case, the prompt generator $\mathcal{G}$ is used by bankrupt agents to propose amended prompts that correct their failure modes or explore complementary regions of the behavior space.

Formally, between episodes the population update consists of three stages:

1. **Rent:** each agent pays ρ, so $W_a \leftarrow W_a - \rho$;
2. **Removal:** agents with $W_a < 0$ are deleted;
3. **Injection:** new agents are added according to the exploitation and exploration mechanisms until the population satisfies the prescribed maximum size constraints.

This induces a stochastic population process over finite sets of prompted agents, driven jointly by environment randomness, LLM stochasticity, and prompt generation.

A key design choice is that bids are not learned online; instead, each agent receives a bid when it is introduced, which is then frozen. For a newly injected agent a' (novice), let t denote its first step

at which it becomes eligible, and let $C_t = \{a \in \mathcal{P}_e \setminus \{a'\} : \phi_a(o_t) = 1\}$ be the set of competing eligible agents at that step. We assign its bid according to the novice rule

$$b_{a'} = \left(\max_{a \in C_t} b_a\right) + \varepsilon_{a'}, \qquad \varepsilon_{a'} \sim \mathcal{D}_\varepsilon, \tag{2}$$

with $\max \emptyset := 0$. In our experiments, $\mathcal{D}_\varepsilon$ is a small positive perturbation distribution. This rule guarantees that the new agent wins the first auction for which it is eligible, forcing the system to test it at least once before market selection determines whether it should survive (more discussion in Appendix A). Together, exploitation and exploration drive a self-evolving system. Exploitation preserves and sharpens useful behaviors, while exploration injects novelty and enables adaptation. Crucially, evolution is governed entirely by economic signals, i.e., wealth accumulation and loss, without centralized supervision or explicit global performance labeling.

2.4 Training and Evaluation

During optimization (Algorithm 1), each task episode uses the auction-based planning mechanism from Section 2.2: agents bid for control, winning agents act in the environment, bid transfers propagate wealth along the trajectory, and environment rewards update the final actor. These interactions provide an implicit evaluation signal, after which the adaptation step in Section 2.3 removes bankrupt agents, retains profitable ones, mutates successful templates, and replenishes the population to preserve diversity. For evaluation (Algorithm 2), we reuse the same planning structure but disable the evolutionary and economic updates. The trained population is frozen, bids are fixed, no payments, rewards, rent, births, or mutations are applied, and each test task runs on a thread-local snapshot to avoid shared-state effects. Thus, optimization uses auctions for both action selection and credit redistribution, while evaluation uses them only as a decentralized decision rule, measuring the behavior of the learned population without further learning or wealth dynamics.

3 Experiments

3.1 Setup

Partial vs. complete agents. We use *partial agent* to denote any agent whose capability is intentionally incomplete to solve the full task. This partiality can take different forms across domains: an agent may have a restricted action space, access to only one tool, a short generation budget, specialized roles, or only partial observation of the environment. A *complete agent*, by contrast, has access to the full task interface, thus can use the full action space and attempt to solve the task end-to-end. This distinction lets us test whether economic organization can compensate for, or even outperform, capability concentrated in a single complete agent.

Tasks. We instantiate EoM on five domains. On **mathematical reasoning**, we use MATH [16] to train on an easy-to-hard task stream from Level 1 to 5 and evaluate greedy pass@1 accuracy by difficulty level. The population is initialized with planner, executor, and verifier agents, each with short output budgets of 128 tokens on average. On **financial research**, we use Finance-Agent-Bench [5], where agents solve financial questions over company filings using four tools; each partial agent has access to only one tool. For **scientific research**, we use FrontierScience-Research [44], where agents are tasked to solve open-ended scientific questions with literature, planner, executor, and verifier roles. For **accelerator design**, we use the GEMMINI benchmark suite [13] and optimize mappings for 24 ResNet-50 convolution kernels, with minimizing energy-delay product (EDP) as the objective. The population is initialized with three role-specialized partial agents: *Historians* that summarize prior trial outcomes, *Planners* that propose mapping-level search directions, and *Executors* that carry out local mapping evaluations. For **distributed-system optimization**, we use Cloudcast, a task from ADRS [8], in which agents iteratively improve a program to minimize the total data-transfer cost. Full task protocols, splits, agent definitions, tools, and model backbones are provided in Appendix D.

Baselines. Complete-agent baselines include REACT [51], which solves each task end-to-end with the same backbone and full task interface; GEA [46], which performs self-improvement through experience sharing and agent evolution; and OPENEVOLVE [36, 28], an evolutionary monolithic coding-agent baseline. As a partial-agent baseline, we use Multi-Agent Debate [11], where several partial agents interact by communication but do not use market-driven population evolution. For

Table 1: **Performance across domains.** Left: MATH accuracy comparing constrained populations to complete agents (* denotes officially reported numbers). Right: Accelerator design results measured by average EDP (lower is better). In both settings, EoM achieves stronger performance than corresponding baselines.

Backend	Partial agents		Complete agent
	Initial	After training	
Llama-3.1-8B	15.9 (1.37)	**57.0** (3.36)	51.9*
Gemma-2-9B	4.2 (0.52)	**45.1** (4.12)	44.3*

Method	Avg. EDP (μJ·Mcyc) ↓
DOSA [18]	80.2
Complete agent (Gemma-4-31B-it)	43.1
EoM (Gemma-4-31B-it)	**39.3**

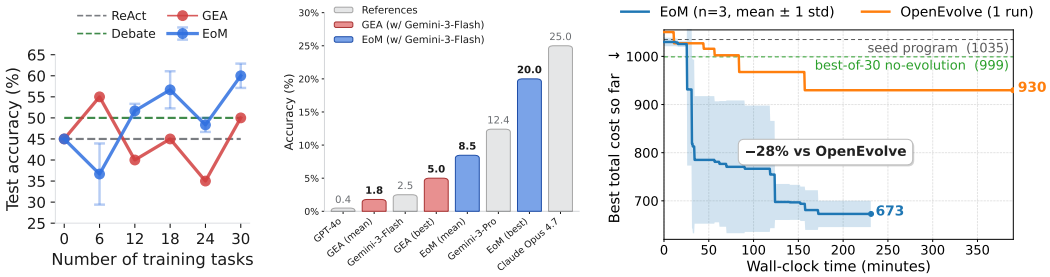
domain-specific comparisons, we also include DOSA [18] for accelerator design. Additional baseline details are in Appendix D.1.

3.2 Can Economics Turn Weak Individuals into Stronger Systems?

Across all five domains, we show that economic coordination turns individually partial agents into collective systems that match or outperform complete-agent baselines. On MATH, EoM improves Llama-3.1-8B agents from 15.9% to 57.0% and Gemma-2-9B agents from 4.2% to 45.1%, exceeding the corresponding complete-agent baselines of 51.9% and 44.3% in Table 1 (left). This is notable because each individual agent in the population is role-specialized and restricted to short outputs, while the complete-agent baseline can attempt the full problem end-to-end without restrictions. On accelerator design, EoM reduces average EDP to 39.3, compared with 43.1 for the same-backbone complete ReAct agent and 80.2 for the domain-specific DOSA baseline in Table 1 (right).

Figure 4 shows that, on Finance-Agent-Bench, EoM rises from 45.0% at initialization to 60.0% after 30 training tasks. This outperforms Multi-Agent Debate at 50.0%, ReAct at 45.0%, and GEA at 50.0%, even though each partial agent in EoM can access only one tool. On FrontierScience-Research, EoM reaches 8.5% mean accuracy and 20.0% best-run accuracy, compared with 1.8% mean and 5.0% best-run accuracy for GEA under the same Gemini-3-Flash backbone. Finally, on Cloudcast, EoM reaches an average total cost of 673 across three attempts, with the best attempt achieving 657, compared with 930 for OpenEvolve, corresponding to a 28% reduction in best cost while using fewer optimization episodes.

Together, these results show that the relevant advantage is not merely "many agents" over "one agent." Rather, EoM shows that a population of partial agents, when organized by economic interactions, can match or surpass complete agents with greater individual access to the task interface.



(a) Finance-Agent-Bench (b) FrontierScience (c) Cloudcast

Figure 4: **Performance across domains.** EoM consistently outperforms baselines, demonstrating the benefits of economic coordination among agents.

3.3 Beyond Multiple Agents: The Role of Economic Ingredients

The gains are not explained by merely having multiple agents: they depend on the economic dynamics that allocate control, transfer value, remove unproductive agents, and propagate successful ones. The ablations in Table 2 show that weakening these dynamics consistently reduces performance.

On MATH, the original system achieves the strongest partial-agent performance, with 43.9 mean accuracy and 57.0 best-run accuracy. Perturbing the economic parameters lowers performance: increasing rent, decreasing rewards, or increasing rewards reduces the mean to 39.0–41.8 and the best

run to 44.0–47.0. This indicates that performance depends on the balance between reward inflow, rent pressure, and agent survival.

On Finance-Agent-Bench, the full system again achieves the strongest overall result, with 52.5 mean accuracy and 65.0 best-run accuracy. Removing exploration causes a large drop to 26.0 mean and 40.0 best accuracy, while removing exploitation lowers the mean to 33.5. Removing auctions also underperforms the full system, reaching 48.0 mean and 58.5 best accuracy. These results suggest that the population needs both sides of the economic process: exploration introduces new candidate agents with awareness of past failures, while exploitation propagates successful ones.

Cloudcast provides a complementary comparison. In Figure 4 (c), EOM reaches a best cost of 673, while the best-of-N multi-agent baseline reaches only 999. Since this baseline uses multiple agents but does not evolve them through market selection, the gap shows that repeated multi-agent sampling alone is insufficient. Thus, economic dynamics are not incidental implementation details; they are the mechanism that turns a collection of partial agents into an adaptive society.

Table 2: **Ablations on MATH (left) and Finance-Agent-Bench (right).** Sensitivity of performance to economic parameters (left) and component removal (right). In both cases, the full/original system achieves the strongest overall results.

Agent Configuration		Mean (%)	Best (%)
Complete	/	51.9	51.9
Constrained	large rent ($\times 10$)	41.8	47.0
	small reward ($\times 0.2$)	39.0	44.0
	large reward ($\times 4$)	40.9	47.0
	original	**43.9**	**57.0**

Agent Configuration		Mean (%)	Best (%)
Complete	/	45.0	45.0
Constrained	w/o auction	48.0	58.5
	w/o exploration	26.0	40.0
	w/o exploitation	33.5	60.0
	full	**52.5**	**65.0**

3.4 How Does the Economy Improve Performance?

We show that the economy improves performance by selecting useful action chains, reproducing successful lineages, eliminating unproductive agents, and gradually sharpening both agent prompts and interaction topology.

What changes inside the society as performance improves? The training dynamics reveal that EOM improves by reshaping both the population and the agents themselves. On Finance-Agent-Bench, Figure 4 (a) shows a non-monotonic but improving trajectory: EOM starts at 45.0, dips during early exploration, then recovers and finishes at 60.0. This pattern is consistent with a market that first reallocates control and tests alternative specialists before converging to stronger coordination.

Accelerator design gives a more direct view of the economic mechanism. In Figure 5, per-agent wealth trajectories show how useful lineages survive while ineffective ones disappear. In panel (a), Historian children born during the run quickly lose wealth and are removed, indicating that the inherited mutation could not earn back its rent. In panel (b), a Planner lineage spawns two good-birth offspring and continues to dominate the auction, while a Historian-line bad-birth child eventually bankrupts. In panel (c), a successful Historian lineage and a failing Executor lineage evolve in parallel. Across these cases, wealth concentrates on agents that repeatedly contribute to record-breaking submissions, while low-value agents are pushed toward bankruptcy. In short, the economy improves performance by turning trajectory-level success into population-level selection.

Does the society learn reusable structure? We show that the society learns more than task-specific answer traces. Figure 7 (a) shows that EOM achieves a $2.2\times$ geometric-mean EDP gain over DOSA across all 24 ResNet-50 convolution kernels, with much larger gains on the hardest kernels: $37.5\times$, $26.3\times$, $17.3\times$, and $12.0\times$ on `Conv` 14, 16, 17, and 4. These gains are structured rather than uniform. The hardest kernels are the 1×1 convolutions inside ResNet-50's bottleneck blocks, which have large input- and output-channel counts but small spatial dimensions. For such shapes, an *output-stationary* dataflow, which holds each output partial sum in fast on-chip storage and accumulates contributions along the input-channel dimension, is a known effective design pattern. Importantly, EOM is not given this motif as a template. The auction rewards only EDP record-breaks, with no per-kernel labels, dataflow-specific reward shaping, or hand-coded output-stationary preference. Nevertheless, across the strongest solutions, the population repeatedly converges on the same tiling pattern, recovering a transferable hardware–software co-design heuristic that DOSA misses. Thus, market selection can

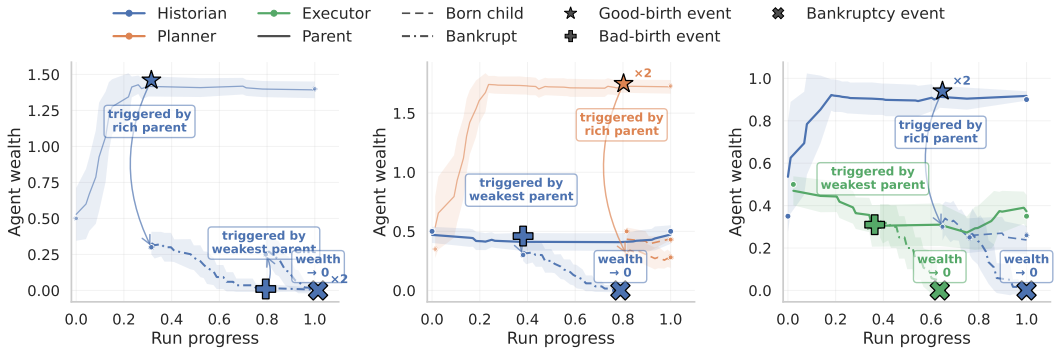


Figure 5: **Training dynamics in accelerator design.** Per-agent wealth on three representative ResNet-50 kernels. Wealth flow to agents that produce new EDP records; rent uniformly deducts wealth. Periodic births spawn *good-birth* children ($\star$, *exploitation*: mutated from the richest agent) and *bad-birth* children ($+$, *exploration*: amended from the weakest); wealth < 0 triggers bankruptcy ($\times$). Shaded bands are rolling $\pm 1\sigma$. **(a)** Both Historian descendants bankrupt—inherited bias fails market pressure. **(b)** A Planner lineage reproduces twice while a Historian bad-birth child eventually fails. **(c)** A strong Historian and a struggling Executor lineage co-exist.

discover reusable domain structure when successful agents are allowed to accumulate wealth and propagate through mutation.

Beyond this domain-specific behavior, we observe two broader forms of adaptation. The population evolves sharper prompt-level strategies, as illustrated in Appendix E.1, enables cross-domain transfer, as shown in Appendix E.2 and reorganizes its interaction topology, as shown in Appendix E.3. Appendix F traces both adaptations in a Cloudcast run. Together, these analyses suggest that economic dynamics shape both internal agent policies and macroscopic social structure.

3.5 Robustness and Generalization

We show that the evolving society is robust in three complementary senses: skills learned on easier tasks transfer to harder ones, performance depends on curriculum but does not collapse under reversed ordering, and adding a complete generalist does not destroy specialization.

Do learned behaviors transfer from easier tasks to harder ones? On MATH, training on an easy-to-hard stream improves not only the easier levels encountered earlier in training, but also harder levels that are initially beyond the agents' capabilities. As shown in Figure 6, both Llama-3.1-8B and Gemma-2-9B improve on every difficulty band. The largest gains appear on Levels 1–3, where Llama-3.1-8B rises to roughly 55–70% and Gemma-2-9B to roughly 45–65%. Importantly, improvement also transfers to Level 5: performance rises from around 10% at the beginning to about 20% by the end for both backbones. Thus, local reasoning routines learned on simpler problems can be recomposed on harder ones.

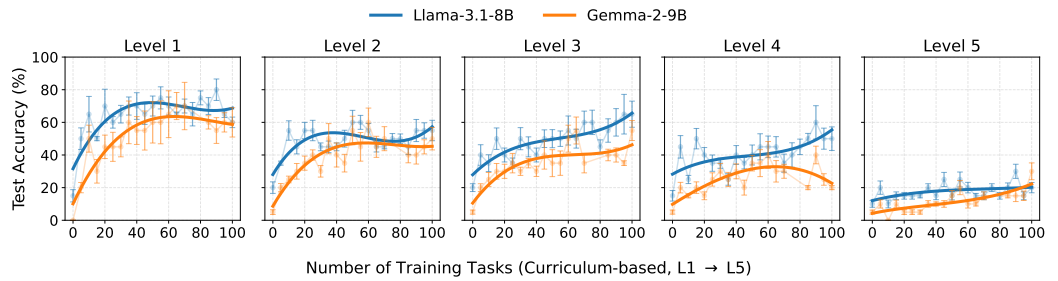
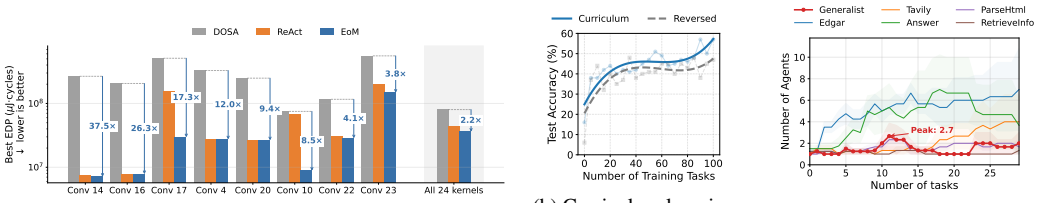


Figure 6: **Easy-to-hard generalization on MATH.** Test accuracy across MATH difficulty levels during training. The partial agent population improves not only on the easier levels encountered earlier, but also on harder levels that are initially beyond its capability, indicating that behaviors learned on simple problems can be reused on more difficult ones.

How sensitive is the society to curriculum order? We compare the default easy-to-hard curriculum with a reversed hard-to-easy schedule. Figure 7 (b) shows that both schedules improve quickly at the beginning, but the easy-to-hard curriculum stays ahead for most of training and finishes clearly higher, at roughly 57% versus about 47% for the reversed schedule. The reversed curriculum plateaus in the low 40s for much of training, while the easy-to-hard curriculum continues improving late in training. This indicates that partial specialists benefit from first mastering reusable local routines before confronting the hardest problems, although the system still improves under the reversed order.



(a) Per-kernel accelerator EDP

(b) Curriculum learning on MATH

(c) Finance research with a generalist

Figure 7: **Mechanism, robustness, and generalization analyses.** **(a)** Per-kernel EDP on ResNet-50. Best EDP found by DOSA, REACT, and EOM on log scale; lower is better. **(b)** Comparison between the default easy-to-hard curriculum and a reversed hard-to-easy curriculum. **(c)** Adding a strong generalist agent with access to all tools does not automatically dominate specialized agents.

Can a complete generalist monopolize the economy? A natural concern is that if a single agent is given complete access to the task interface, it may dominate the market and collapse the society into a monolithic system. We test this in Finance-Agent-Bench by adding a generalist agent with access to all tools alongside the partial specialist agents. As shown in Figure 7 (c), the generalist briefly expands around tasks 11–12, but then contracts and returns to a single agent for the remainder of training. Meanwhile, specialized populations such as Edgar and Tavily continue to grow, reaching roughly 5–8 agents late in training. This suggests that broader access does not automatically imply market dominance. The economy rewards local value: a specialist whose wake-up condition, tool use, and evidence standard are tuned to a narrow subproblem can outcompete a generalist whose prompt budget is spread across many heterogeneous responsibilities. We provide a detailed analysis in Appendix G, where we show that specialists evolve sharper local decision rules, while the generalist tends to accumulate broad but diluted procedural instructions. Thus, the society remains decentralized not because complete agents are forbidden, but because the market continues to favor locally more precise specialists.

4 Related Work

Self-Evolving Agents. Recent work studies how agents can improve through interaction, adaptation, and accumulated experience rather than relying on fixed centralized control. In the LLM setting, this includes co-evolution, iterative refinement, and experience sharing across agent populations [7, 46, 9]. Related work on reward-driven self-organization shows that structured behaviors can emerge from local feedback alone [55]. More broadly, rich ecological environments and evolutionary pressures have been shown to induce increasingly complex adaptive behaviors, suggesting that agent capabilities can emerge through continual interaction with other agents and the environment [4]. For more related work, please refer to Appendix B.

5 Limitations, Conclusions, and Future Work

This work shows that decentralized economic interactions can turn a population of agents into adaptive collective intelligence. Across diverse domains, auctions, transactions, and wealth-based selection induce specialization and coordination without centralized control, suggesting that designing incentives is a powerful alternative to designing individual agents.

A key limitation is that adaptation occurs only in prompt space with a frozen backbone, which may restrict capability growth in tasks requiring new skills or representations. Future work can extend this framework to parameter-space training, hybrid adaptation, and broader agent backbones, including multimodal or embodied systems.

Acknowledgments and Disclosure of Funding

This work was supported in part by a gift from the Chan Zuckerberg Initiative Foundation to establish the Kempner Institute at Harvard University.

References

[1] Eric B. Baum. Toward a model of mind as a laissez-faire economy of idiots. In *Proceedings of the 13th International Conference on Machine Learning*, pages 20–27, 1996. 2, 18

[2] Eric B Baum. Toward a model of intelligence as an economy of agents. *Machine Learning*, 35 (2):155–185, 1999. 2, 3, 17

[3] Eric B Baum and Igor Durdanovic. Evolution of cooperative problem solving in an artificial economy. *Neural Computation*, 12(12):2743–2775, 2000. 3, 18

[4] Joseph Bejjani, Chase Van Amburg, Chengrui Wang, Chloe Huangyuan Su, Sarah M Pratt, Yasin Mazloumi, Naeem Khoshnevis, Sham M Kakade, Kianté Brantley, and Aaron Walsman. The emergence of complex behavior in large-scale ecological environments. *arXiv preprint arXiv:2510.18221*, 2025. 9

[5] Antoine Bigeard, Langston Nashold, Rayan Krishnan, and Shirley Wu. Finance agent benchmark: Benchmarking llms on real-world financial research tasks. *arXiv preprint arXiv:2508.00828*, 2025. 5, 25

[6] Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, et al. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657*, 2025. 2

[7] Yixing Chen, Yiding Wang, Siqi Zhu, Haofei Yu, Tao Feng, Muhan Zhang, Mostofa Patwary, and Jiaxuan You. Multi-agent evolve: Llm self-improve through co-evolution. *arXiv preprint arXiv:2510.23595*, 2025. 2, 9

[8] Audrey Cheng, Shu Liu, Melissa Pan, Zhifei Li, Bowen Wang, Alex Krentsel, Tian Xia, Mert Cemri, Jongseok Park, Shuo Yang, et al. Barbarians at the gate: How ai is upending systems research. *arXiv preprint arXiv:2510.06189*, 2025. 5, 26

[9] X. Dai, Y. Zhu, et al. Geoevolver: A self-evolving multi-agent system for earth observation. *arXiv preprint arXiv:2602.02559*, 2026. 9

[10] M Bernardine Dias, Robert Zlot, Nidhi Kalra, and Anthony Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006. 17

[11] Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first international conference on machine learning*, 2024. 5, 17, 25

[12] Paul Duetting, Vahab Mirrokni, Renato Paes Leme, Haifeng Xu, and Song Zuo. Mechanism design for large language models. In *Proceedings of the ACM Web Conference 2024*, pages 144–155, 2024. 17

[13] Hasan Genc, Seah Kim, Alon Amid, Ameer Haj-Ali, Vighnesh Iyer, Pranav Prakash, Jerry Zhao, Daniel Grubb, Harrison Liew, Howard Mao, et al. Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 769–774. IEEE, 2021. 5, 26

[14] Friedrich A. Hayek. The use of knowledge in society. *The American Economic Review*, 35(4): 519–530, 1945. 2

[15] Friedrich A. Hayek. *Law, Legislation and Liberty*. University of Chicago Press, 1973. 18

[16] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021. 5, 25

[17] John H. Holland. Properties of the bucket brigade algorithm. In *Proceedings of the 1st International Conference on Genetic Algorithms*, 1985. 17, 18

[18] Charles Hong, Qijing Huang, Grace Dinh, Mahesh Subedar, and Yakun Sophia Shao. Dosa: Differentiable model-based one-loop search for dnn accelerators. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 209–224, 2023. 6, 25, 26

[19] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The twelfth international conference on learning representations*, 2023. 2

[20] Shariq Iqbal, Robby Costales, and Fei Sha. Alma: Hierarchical learning for composite multi-agent tasks. *Advances in neural information processing systems*, 35:7155–7166, 2022. 2

[21] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. 2

[22] Yubin Kim, Ken Gu, Chanwoo Park, Chunjong Park, Samuel Schmidgall, A Ali Heydari, Yao Yan, Zhihan Zhang, Yuchen Zhuang, Yun Liu, et al. Towards a science of scaling agent systems. *arXiv preprint arXiv:2512.08296*, 2025. 2

[23] Shalev Lifshitz, Sheila A McIlraith, and Yilun Du. Multi-agent verification: Scaling test-time compute with multiple verifiers. *arXiv preprint arXiv:2502.20379*, 2025. 17

[24] Yexiong Lin, Shanshan Ye, Yu Yao, Zhen Fang, Bo Han, and Tongliang Liu. Squidagent: Parallelize wisely, coordinate efficiently. *arXiv preprint arXiv:2610.08647*, 2026. 17

[25] Shuo Liu, Tianle Chen, and Christopher Amato. Improved multi-agent collaboration with multi-turn reinforcement learning. In *First Workshop on Multi-Turn Interactions in Large Language Models*. 17

[26] Carol Xuan Long, David Simchi-Levi, Feng Zhu, Huangyuan Su, Andre P. Calmon, and Flavio P. Calmon. Reliability and effectiveness of autonomous ai agents in supply chain management. *arXiv preprint arXiv:2605.17036*, 2026. 17

[27] Marvin Minsky. *The Society of Mind*. Simon and Schuster, 1988. 18

[28] Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025. 5, 24, 26

[29] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P How, and John Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *International conference on machine learning*, pages 2681–2690. PMLR, 2017. 2

[30] Chanwoo Park, Seungju Han, Xingzhi Guo, Asuman E Ozdaglar, Kaiqing Zhang, and Joo-Kyung Kim. Maporl: Multi-agent post-co-training for collaborative large language models with reinforcement learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30215–30248, 2025. 17

[31] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37: 126544–126565, 2024. 2

[32] Shvetank Prakash, Andrew Cheng, Arya Tschand, Mark Mazumder, Varun Gohil, Jeffrey Ma, Jason Yik, Zishen Wan, Jessica Quaye, Elisavet Lydia Alvanaki, et al. Quarch: A benchmark for evaluating llm reasoning in computer architecture. *arXiv preprint arXiv:2510.22087*, 2025. 26

[33] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 15174–15186, 2024. 2

[34] Ao Qu, Han Zheng, Zijian Zhou, Yihao Yan, Yihong Tang, Shao Yong Ong, Fenglu Hong, Kaichen Zhou, Chonghe Jiang, Minwei Kong, et al. Coral: Towards autonomous multi-agent evolution for open-ended discovery. *arXiv preprint arXiv:2604.01658*, 2026. 2

[35] Jürgen Schmidhuber. The neural bucket brigade: A local learning algorithm for dynamic feedforward and recurrent networks. *Connection Science*, 1(4):403–412, 1989. 18

[36] Asankhaya Sharma. Openevolve: an open-source evolutionary coding agent, 2025. URL https://github.com/algorithmicsuperintelligence/openevolve. 5, 24, 26

[37] Vighnesh Subramaniam, Yilun Du, Joshua B Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. Multiagent finetuning: Self improvement with diverse reasoning chains. *arXiv preprint arXiv:2501.05707*, 2025. 17

[38] Abhimanyu Pallavi Sudhir and Long Tran-Thanh. Market-based architectures in rl and beyond. *Accepted to AAMAS 2025 Blue Sky Track*, abs/2503.05828, 2025. 17

[39] Hanbo Sun, Chenyu Wang, Zhenhua Zhu, Xuefei Ning, Guohao Dai, Huazhong Yang, and Yu Wang. Gibbon: Efficient co-exploration of nn model and processing-in-memory architecture. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 867–872. IEEE, 2022. 26

[40] Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon llm agent via context-folding. *arXiv preprint arXiv:2510.11967*, 2025. 2

[41] Arya Tschand, Chenyu Wang, Zishen Wan, Andrew Cheng, Ioana Cristescu, Kevin He, Howard Huang, Alexander Ingare, Akseli Kangaslahti, Sara Kangaslahti, Theo Lebryk, Hongjin Lin, Jeffrey Jian Ma, Alexandru Meterez, Clara Mohri, Depen Morwani, Sunny Qin, Roy Rinberg, Paula Rodriguez-Diaz, Alyssa Mia Taliotis, Pernille Undrum Fathi, Rosie Zhao, Todd Zhou, and Vijay Janapa Reddi. Genai for systems: Recurring challenges and design principles from software to silicon, 2026. URL https://arxiv.org/abs/2602.15241. 26

[42] Chenyu Wang, Zishen Wan, Hao Kang, Emma Chen, Zhiqiang Xie, Tushar Krishna, Vijay Janapa Reddi, and Yilun Du. Slm-mux: Orchestrating small language models for reasoning. *arXiv preprint arXiv:2510.05077*, 2025. 17

[43] Jian Sheng Wang. Aesp: A human-sovereign economic protocol for ai agents with privacy-preserving settlement. *arXiv preprint arXiv:2603.00318*, 2026. 18

[44] Miles Wang, Robi Lin, Kat Hu, Joy Jiao, Neil Chowdhury, Ethan Chang, and Tejal Patwardhan. Frontierscience: Evaluating ai's ability to perform expert-level scientific tasks. *arXiv preprint arXiv:2601.21165*, 2026. 5, 25

[45] Yiping Wang, Shao-Rong Su, Zhiyuan Zeng, Eva Xu, Liliang Ren, Xinyu Yang, Zeyi Huang, Xuehai He, Luyao Ma, Baolin Peng, et al. Thetaevolve: Test-time learning on open problems. *arXiv preprint arXiv:2511.23473*, 2025. 26

[46] Zhaotian Weng, Antonis Antoniades, Deepak Nathani, Zhen Zhang, Xiao Pu, and Xin Eric Wang. Group-evolving agents: Open-ended self-improvement via experience sharing. *arXiv preprint arXiv:2602.04837*, 2026. 5, 9, 24

[47] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First conference on language modeling*, 2024. 2

[48] Minghui Xu. The agent economy: A blockchain-based foundation for autonomous ai agents. 2026. NeurIPS 2026. 17

[49] Xiangyuan Xue, Yifan Zhou, Guibin Zhang, Zaibin Zhang, Yijiang Li, Chen Zhang, Zhenfei Yin, Philip Torr, Wanli Ouyang, and Lei Bai. Comas: Co-evolving multi-agent systems via interaction rewards. *arXiv preprint arXiv:2510.08529*, 2025. 2

[50] Yingxuan Yang, Huacan Chai, Shuai Shao, Yuanyi Song, Siyuan Qi, Renting Rui, and Weinan Zhang. Agentnet: Decentralized evolutionary coordination for llm-based multi-agent systems, 2025. URL https://arxiv.org/abs/2504.00587. 2, 17

[51] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022. 2, 5, 24

[52] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026. 2

[53] Mert Yuksekgonul, Daniel Koceja, Xinhao Li, Federico Bianchi, Jed McCaleb, Xiaolong Wang, Jan Kautz, Yejin Choi, James Zou, Carlos Guestrin, and Yu Sun. Learning to discover at test time. *arXiv preprint*, 2026. 26

[54] Miao Zhang, Junsik Kim, Siyuan Xiang, Jian Gao, and Cheng Cao. Dynamic role assignment for multi-agent debate. *arXiv preprint arXiv:2601.17152*, 2026. 17

[55] Heng Zhou, Hejia Geng, Xiangyuan Xue, Li Kang, Yiran Qin, Zhiyong Wang, Zhenfei Yin, and Lei Bai. Reso: A reward-driven self-organizing llm-based multi-agent system for reasoning tasks. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 15990–16009, 2025. 9

[56] Andy Zhu and Yingjun Du. A role-aware multi-agent framework for financial education question answering with llms. *arXiv preprint arXiv:2509.09727*, 2025. 17

[57] Mingchen Zhuge, Haozhe Liu, Francesco Faccio, Dylan R Ashley, Róbert Csordás, Anand Gopalakrishnan, Abdullah Hamdi, Hasan Abed Al Kader Hammoud, Vincent Herrmann, Kazuki Irie, et al. Mindstorms in natural language-based societies of mind. *arXiv preprint arXiv:2305.17066*, 2023. 18

Appendix - Table of Contents

A Extended Implementation Details **16**

B Extended Related Works **17**

C Theoretical Motivations **18**

C.1 Market selection drives bids toward value 18

C.2 Outcome-only reward is sufficient for a strong population 18

C.3 Regret to an oracle coordinator 19

C.4 Bucket-brigade payments recover structured credit assignment 19

C.5 Malicious agents and collusion 20

C.6 Proofs for Section C.1 20

C.7 Proofs for Section C.2 22

C.8 Proofs for Section C.3 22

C.9 Proofs for Section C.4 23

C.10 Proofs for Section C.5 23

D Extended Experiment Details **24**

D.1 Baseline Details 24

D.2 Mathematical Reasoning 25

D.3 Financial Research 25

D.4 Scientific Research 25

D.5 Accelerator Design 26

D.6 Distributed-System Optimization 26

D.7 Evaluation Protocol 27

E Case Study: Prompt and Topology Evolution in Scientific Research **27**

E.1 Prompt-Level Evolution: From Generic Execution to Reusable Scientific Reasoning 27

E.2 Cross-Domain Transfer of the Evolved Reasoning Routine 28

E.3 From Local Reasoning Evolution to Topology Evolution 29

E.4 Early Regime: Explicit Multi-Role Auditing 29

E.5 Late Regime: Compact Specialist Execution 29

E.6 Training-Wide Topology Trend 30

E.7 Population-Level Selection 30

F Case Study: Prompt and Topology Evolution in CloudCast **30**

F.1 Auction-Derived Topology 31

F.2 Prompt Evolution: Checks Before Costly Actions 31

G Why the Generalist Does Not Monopolize **32**

G.1 Initial Generalist Prompt 33

G.2 Evolved Generalist Prompt 33
G.3 Comparison to Specialized Agent Evolution 34
G.4 Generalist Failure as Evidence for Specialization 34

A Extended Implementation Details

Training and evaluation pseudo code for EOM are shown below:

Algorithm 1 The Training Loop

Require: task stream $\mathcal{D}$, initial population $\mathcal{P}_0$, wake-up oracle ω
Require: base bid b_0, novice premium ϵ, step cap S, trial cap T
Require: bounds $[N_{\min}, N_{\max}]$; rent ρ every K_r tasks
Require: birth probs p_a, p_b (bankruptcy), p_g (periodic); period K_b, batch B

$\mathcal{P} \leftarrow \mathcal{P}_0$
for episode $e = 1, 2, \ldots$ over $\tau \sim \mathcal{D}$ **do**
  $\sigma \leftarrow \{a \mapsto \textsc{Snapshot}(a) : a \in \mathcal{P}\}$; $\mathcal{B}_e \leftarrow \emptyset$ ▷ e.g. wealth, capability, ...
  for trial $t = 1, \ldots, T$ **do** ▷ replay until no bankruptcy
    $a^{\text{last}}, H \leftarrow \textsc{Planning}(\tau, \mathcal{P})$; a^{last}.wealth += env.reward() ▷ outcome reward to final actor
    $\mathcal{B} \leftarrow \{a : a.\text{wealth} \le 0\}$
    if $\mathcal{B} = \emptyset$ **then break**
    end if
    $\mathcal{P} \leftarrow \mathcal{P} \setminus \mathcal{B}$; $\mathcal{B}_e \cup= \mathcal{B}$
    for all $a \in \mathcal{P}$ **do** $\textsc{Restore}(a, \sigma(a))$
    end for ▷ rollback survivors
  end for
  if $e \bmod K_r = 0$ **then** ▷ periodic rent
    a.wealth $-= r$ for all $a \in \mathcal{P}$
    $\mathcal{B}' \leftarrow \{a \in \mathcal{P} : a.\text{wealth} \le 0\}$; $\mathcal{P} \leftarrow \mathcal{P} \setminus \mathcal{B}'$; $\mathcal{B}_e \cup= \mathcal{B}'$
  end if
  $\textsc{Adaptation}(\mathcal{P}, \mathcal{B}_e, e)$ ▷ bankruptcy + periodic births, replenish to $N_{\min}$
end for

procedure $\textsc{Planning}(\tau, \mathcal{P})$
  init env on τ; $H \leftarrow \emptyset$; $a^- \leftarrow \perp$
  for $s = 1, \ldots, S$ **do**
    $\mathcal{E} \leftarrow \{a \in \mathcal{P} : \omega(a, \text{env})\}$; **if** $\mathcal{E} = \emptyset$ **break**
    $\textsc{SetBids}(\mathcal{E})$ ▷ pluggable: e.g. novices enter at $\max_{\text{vet}} b + \epsilon$ and lock
    $a^* \leftarrow \arg\max_{a \in \mathcal{E}} b_a$ (random tie-break)
    a^*.wealth $-= b_{a^*}$; **if** $a^- \neq \perp$: a^-.wealth $+= b_{a^*}$
    env.apply(a^*.act(env)); $H \leftarrow H \cup \{a^*\}$; $a^- \leftarrow a^*$
    if env terminated **break**
  end for
  return a^-, H
end procedure

procedure $\textsc{Adaptation}(\mathcal{P}, \mathcal{B}_e, e)$
  for all $a \in \mathcal{B}_e$ with $|\mathcal{P}| < N_{\max}$ **do**
    $u \sim \mathcal{U}(0, 1)$
    if $u < p_a$: $\mathcal{P} \leftarrow \mathcal{P} \cup \{\textsc{Mutate}(\text{richest}(\mathcal{P}))\}$
    elif $u < p_a + p_b$: $\mathcal{P} \leftarrow \mathcal{P} \cup \{\textsc{Amend}(a)\}$
  end for
  if $e \bmod K_b = 0$ **then**
    for $i = 1, \ldots, B$ with $|\mathcal{P}| < N_{\max}$ **do**
      $a' \leftarrow \textsc{Mutate}(\text{richest}(\mathcal{P}))$ with prob. p_g, else $\textsc{Amend}(\text{poorest}(\mathcal{P}))$
      $\mathcal{P} \leftarrow \mathcal{P} \cup \{a'\}$
    end for
  end if
  while $|\mathcal{P}| < N_{\min}$ **do**
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{\textsc{Mutate}(a_0)\}$ for some template $a_0 \in \mathcal{P}_0$
  end while
end procedure

Algorithm 2 The Evaluation Loop

Require: trained population $\mathcal{P}^*$, test stream $\mathcal{D}_{\text{test}}$, wake-up oracle ω
Require: step cap S; thread-pool size W

$\mathcal{P} \leftarrow \mathcal{P}^*$; freeze $\{w_a, b_a, \text{status}_a\}_{a \in \mathcal{P}}$ ▷ no payments, no rewards, no rent, no births
for task $\tau \in \mathcal{D}_{\text{test}}$ **in parallel** (workers W) **do**
  $\mathcal{P}_\tau \leftarrow \textsc{Snapshot}(\mathcal{P})$ ▷ thread-local deep copy; no shared mutation
  $a^{\text{last}}, H \leftarrow \textsc{Planning}(\tau, \mathcal{P}_\tau)$ ▷ no a^{last}.wealth update
  record $\{\tau, H, \text{env.terminated}, \text{env.score}()\}$
end for

procedure $\textsc{Planning}(\tau, \mathcal{P})$
  init env on τ; $H \leftarrow \emptyset$; $a^- \leftarrow \perp$
  for $s = 1, \ldots, S$ **do**
    $\mathcal{E} \leftarrow \{a \in \mathcal{P} : \omega(a, \text{env})\}$; **if** $\mathcal{E} = \emptyset$ **break**
    (skip $\textsc{SetBids}(\mathcal{E})$*: bids* b_a *are frozen from training; no novice → veteran promotion)*
    $a^* \leftarrow \arg\max_{a \in \mathcal{E}} b_a$ (random tie-break)
    (skip wealth transfer: a^**.wealth* $-= b_{a^*}$ *and* a^-*.wealth* $+= b_{a^*}$ *are no-ops)*
    env.apply(a^*.act(env)); $H \leftarrow H \cup \{a^*\}$; $a^- \leftarrow a^*$ ▷ no reward credited to a^*
    if env terminated **break**
  end for
  return a^-, H
end procedure

Setting the bids. In the released EOM, bids are never updated online. A new agent a' receives its bid once, at the first step at which it is eligible, via the novice rule (2): $b_{a'} = \max_{a \in C_t} b_a + \varepsilon_{a'}$. The bid is then frozen for the rest of training and at evaluation (Algorithms 1 and 2). The premium $\varepsilon_{a'}$ only ensures that the newcomer wins its first auction and is thus tested at least once. We also experimented with other bid-setting schemes, including LLM-generated bids, a single constant bid shared by all agents, and Holland-style wealth-proportional bids $b_a = \alpha W_a$ [17]. The fixed novice rule above worked best in practice, which is consistent with Baum's finding that adaptive bid rules can produce inflationary speculative bubbles [2].

Bids approximate values. Although no individual bid ever changes, the bids of the surviving population approximate the value of acting. By transfer rule (1), an agent a that wins at context x pays b_a and receives the immediate reward plus the next winner's bid, so its expected profit per win is $V(a, x) - b_a$, where $V(a, x)$ is the market value of the state its action reaches (Appendix C.1). Agents that overbid ($b_a > V(a, x)$) go bankrupt. Agents that underbid are profitable and tend to be mutated; their offspring re-enter at the incumbent's bid plus ε and take over its wins. Bids therefore increase until further entry is unprofitable, so the surviving bids approach the value from below, $V^\star(x) - \varepsilon_{\max} \leq \beta_\infty(x) \leq V^\star(x)$ (Appendix 1). The approximation is coarse: its resolution is $\varepsilon_{\max}$, rent keeps bids slightly below value, and each bid averages over instances on which the agent wins.

B Extended Related Works

Multi-Agent Systems. A parallel line of work examines how multiple agents coordinate, specialize, and solve tasks collectively. Recent work in the LLM era frames multi-agent orchestration as an "society of minds," where specialization and coordination emerge through local interactions rather than centralized planning [10, 38, 48, 24]. Existing approaches include reinforcement-learning-based collaboration [25, 30], dynamic role assignment and role-aware specialization [54, 56, 26], and decentralized evolutionary coordination, where agents dynamically adapt their graph connectivity, specialize through retrieval-augmented memory, and route tasks without a central orchestrator [50]. Other directions include multi-agent debate and verification for improving reasoning and test-time scaling [11, 23, 42], and multi-agent finetuning with diverse reasoning chains for self-improvement [37], alongside broader studies of collaborative and self-organizing multi-agent systems. This perspective also connects to mechanism design for LLMs, which studies how incentives and allocation rules shape agent behavior in strategic environments [12].

Economics and Applications to Agents. The motivation for decentralized agent systems draws on a long tradition in economics and distributed intelligence. Smith's "invisible hand" and Hayek's "spontaneous order" emphasize how coordinated outcomes can emerge from decentralized actors with only local information [15]. Related ideas appear in Minsky's *Society of Mind*, which views intelligence as arising from the interaction of many simple agents rather than a single centralized reasoner [27, 57]. These ideas influenced computational models of coordination and learning, including Holland's bucket brigade for distributed credit assignment [17], Schmidhuber's neural bucket brigade [35], and Baum's Hayek Machine, which introduced property rights, wealth accumulation, and competitive selection among simple agents [1–3]. Recent proposals for economic protocols and settlement layers for autonomous AI agents continue this line by formalizing exchange, sovereignty, and privacy-preserving settlement in modern agent ecosystems [43].

C Theoretical Motivations

We use the notation and auction dynamics from Section 2. Agents are prompted LLM policies with fixed bids and wealths, and wealth evolves through the bucket-brigade transfer rule (1). This section summarizes why the resulting market dynamics select useful specialists, why outcome-only reward can be sufficient once the population is strong, and how bucket-brigade payments provide a structured credit-assignment signal.

C.1 Market selection drives bids toward value

The first result formalizes the basic market-selection mechanism. If an agent repeatedly wins at a recurrent context and its bid is below its expected resale-adjusted payoff, then its wealth has positive drift and it can survive. Conversely, agents whose bids exceed their expected payoff eventually become insolvent. Thus bankruptcy and replacement push the surviving bid frontier toward the value of the best available specialist.

For a recurrent context x, let $V(a, x)$ denote the expected resale-adjusted payoff of agent a, including the immediate reward and the downstream bucket-brigade payment. Let

$$V^\star(x) := \sup_{a:\phi_a(x)=1} V(a, x)$$

be the value of the best eligible specialist, and let $\beta_\infty(x)$ denote the largest bid among agents that win at x infinitely often and are never removed.

Theorem 1 (Selection of bids toward expected payoff)**.** *Suppose context x is recurrent, payoffs are stationary and bounded, and newly injected agents use the novice bidding rule with resolution $\varepsilon_{\max}$. Suppose also that whenever the current solvent frontier at x is more than $\varepsilon_{\max}$ below $V^\star(x)$, there is a positive probability of injecting an eligible near-optimal specialist. Then, almost surely,*

$$V^\star(x) - \varepsilon_{\max} \le \beta_\infty(x) \le V^\star(x).$$

Thus the long-run solvent bid frontier tracks the value of the best prompted specialist, up to the novice-resolution error.

This result explains the role of wealth and bankruptcy: the mechanism does not require explicitly fitting value functions. Instead, agents with overpriced bids lose wealth and are removed, while underpriced useful agents have a chance to accumulate wealth and persist.

C.2 Outcome-only reward is sufficient for a strong population

A natural question is whether the system requires dense process rewards, or whether final outcome reward alone is enough. The next result shows that outcome-only reward is sufficient whenever the auction population is already organized so that the winning agent is approximately the best eligible specialist at each reachable history.

Let $J^{\mathrm{out}}(\pi)$ denote the expected discounted outcome return of a policy π, using only the task-aligned reward r_t^{out}. Let $Q_t^\star(h_t, u)$ be the optimal outcome-value obtained by selecting agent u at history h_t and acting optimally thereafter.

Theorem 2 (Outcome reward suffices for strong agents)**.** *Suppose there exists $\varepsilon \geq 0$ such that for every reachable history h_t with nonempty eligible set E_t, the auction winner $a_t^\star$ satisfies*

$$Q_t^\star(h_t, a_t^\star) \geq \max_{u \in E_t} Q_t^\star(h_t, u) - \varepsilon.$$

Then with the usual interpretation that the sum equals H when $\gamma = 1$, the induced auction policy π^{auc} satisfies

$$J^{\mathrm{out}}(\pi^{\mathrm{auc}}) \geq \sup_\pi J^{\mathrm{out}}(\pi) - \varepsilon \, \frac{1 - \gamma^H}{1 - \gamma},$$

The theorem separates two questions. Process rewards may help train or shape the population, but they are not required for correctness once the market reliably selects near-best specialists under the outcome objective.

C.3 Regret to an oracle coordinator

We next compare the decentralized auction to a centralized oracle that directly selects the best eligible specialist at every history. The result shows that if market bids approximate the oracle action-values, then the auction policy has vanishing regret relative to this coordinator.

Theorem 3 (Vanishing regret of the market auction policy)**.** *Suppose there exists a nonincreasing sequence $\{\beta_e\}_{e \geq 1}$ such that for every episode e, reachable history h_t, and eligible agent $u \in E_{e,t}(h_t)$,*

$$\left| b_{e,t}(h_t, u) - Q_{e,t}^\star(h_t, u) \right| \leq \beta_e.$$

Let r_e be the episode regret of the market policy relative to the oracle coordinator. Then $r_e \leq 2\beta_e \sum_{t=0}^{H-1} \gamma^t$. Consequently, $\mathrm{Reg}(E) := \sum_{e=1}^{E} r_e \leq 2\Big(\sum_{t=0}^{H-1} \gamma^t\Big) \sum_{e=1}^{E} \beta_e$*. In particular, if $\beta_e \leq Be^{-1/2}$, then*

$$\frac{\mathrm{Reg}(E)}{E} = O(E^{-1/2}).$$

This bound makes explicit that the market need not be globally centralized: as long as bids become calibrated to specialist value, the auction tracks the oracle coordinator.

C.4 Bucket-brigade payments recover structured credit assignment

Bucket-brigade payments also provide a mechanism for assigning credit to earlier agents. The next two results show that, under additional structure, the payment received from the next winner can be interpreted as either a Bellman continuation value or an ordered marginal contribution in a workflow.

Proposition 1 (Bellman-like credit under continuation pricing)**.** *If the expected payment received from the next winner equals the discounted continuation value,*

$$\mathbb{E}[Y_t \mid h_t, a_t^\star] = \gamma \mathbb{E}[V^{\mathrm{bb}}(h_{t+1}) \mid h_t, a_t^\star],$$

then the bucket-brigade target satisfies

$$\mathbb{E}[r_t + Y_t \mid h_t, a_t^\star] = \mathbb{E}[r_t + \gamma V^{\mathrm{bb}}(h_{t+1}) \mid h_t, a_t^\star].$$

Thus the winner's wealth update uses a Bellman-like one-step target.

Theorem 4 (Shapley-like credit in DAG workflows)**.** *Consider an acyclic workflow in which subtasks are executed in topological order, and suppose the next winner's bid prices the continuation value of the remaining feasible subgraph. Then the bucket-brigade payment to the current subtask agent coincides, up to a downstream baseline, with that agent's ordered marginal contribution to the final outcome value.*

Hence bucket-brigade transfers do more than redistribute wealth: they can serve as a local credit-assignment signal that propagates downstream value backward through the executed chain of agents.

Throughout the proofs, we write x for a decision context. In a fully observed Markov environment, $x = o_t = s_t$; in partially observed settings, x may denote the history

$$h_t = (o_0, a_0^{\mathrm{env}}, r_0, \ldots, o_t).$$

For an agent a, let b_a denote its fixed bid and W_a its wealth. When a wins at context x, define its resale-adjusted economic payoff as

$$G_a(x) := r_t + \mathbf{1}\{t+1 < H\} b_{a^\star_{t+1}},$$

where r_t is the immediate environment reward and $b_{a^\star_{t+1}}$ is the next winning bid paid back to a under the bucket-brigade transfer rule. At a terminal step, the resale term is zero. Thus the net wealth increment of a winning agent, ignoring rent, is

$$G_a(x) - b_a.$$

Let

$$V(a,x) := \mathbb{E}[G_a(x)], \qquad V^\star(x) := \sup_{a:\phi_a(x)=1} V(a,x).$$

Let $\mathsf{Surv}_\infty(x)$ denote the set of agents that win at context x infinitely often and are never removed, and define the limiting surviving bid frontier

$$\beta_\infty(x) := \sup_{a \in \mathsf{Surv}_\infty(x)} b_a,$$

with $\sup \varnothing := 0$.

C.5 Malicious agents and collusion

Finally, the bankruptcy rule discourages agents that bid aggressively but do not create commensurate value. A malicious agent whose bid exceeds its expected resale-adjusted payoff has negative wealth drift and is removed in finite time.

For colluding agents, the relevant object is not individual wealth but coalition wealth, because internal transfers cancel. A cartel can survive only if some surviving sub-coalition has nonnegative aggregate external wealth drift.

Thus collusion changes the unit of selection from an individual agent to a surviving cartel. Bankruptcy eliminates colluding agents only when every possible surviving cartel is externally loss-making; otherwise a cartel with nonnegative aggregate drift may persist or even form a monopoly.

C.6 Proofs for Section C.1

Lemma 1 (Survival of underpriced agents)**.** *Suppose context x is recurrent and the payoff sequence of agent a at x is stationary and bounded. If*

$$V(a,x) - b_a \geq \delta > 0,$$

then agent a survives forever after its first win at x with positive probability.

Proof of Lemma 1. Fix an agent a and context x. Let $G_{a,k}(x)$ be the resale-adjusted payoff obtained on the k-th win of agent a at x. Define

$$X_k := b_a - G_{a,k}(x).$$

By stationarity and boundedness, $(X_k)_{k\geq 1}$ is i.i.d. and bounded. Moreover,

$$\mathbb{E}[X_k] = b_a - V(a,x) \leq -\delta < 0.$$

Since X_k is bounded and has strictly negative mean, there exists $\lambda > 0$ small enough such that

$$m_\lambda := \mathbb{E}\big[e^{\lambda X_k}\big] < 1.$$

Then

$$M_n := \exp\left(\lambda \sum_{k=1}^{n} X_k\right)$$

is a nonnegative supermartingale.

Let T be the bankruptcy time measured in wins of a at context x:

$$T := \inf\left\{ n \geq 1 : \sum_{k=1}^{n} X_k > W_a(0) \right\},$$

where $W_a(0)$ is the wealth immediately before its first win at x. On $\{T \le n\}$,

$$M_T \ge e^{\lambda W_a(0)}.$$

By optional stopping applied to $T \wedge n$,

$$1 = M_0 \ge \mathbb{E}[M_{T\wedge n}] \ge e^{\lambda W_a(0)}\mathbb{P}(T \le n).$$

Thus

$$\mathbb{P}(T \le n) \le e^{-\lambda W_a(0)}.$$

Letting $n \to \infty$ gives

$$\mathbb{P}(T < \infty) \le e^{-\lambda W_a(0)} < 1.$$

Therefore

$$\mathbb{P}(T = \infty) \ge 1 - e^{-\lambda W_a(0)} =: q_\delta > 0.$$

□

Proof of Theorem 1. We prove the upper and lower bounds separately.

Upper bound. Fix any $a \in \mathsf{Surv}_\infty(x)$. After its first n wins at x, its wealth is bounded above by

$$W_a(n) \le W_a(0) + \sum_{k=1}^{n}\big(G_{a,k}(x) - b_a\big),$$

ignoring nonpositive rent terms. By the strong law of large numbers,

$$\frac{1}{n}\sum_{k=1}^{n}\big(G_{a,k}(x) - b_a\big) \xrightarrow{\text{a.s.}} V(a,x) - b_a.$$

If $b_a > V(a,x)$, then $W_a(n) \to -\infty$ almost surely, so a would eventually be removed. This contradicts $a \in \mathsf{Surv}_\infty(x)$. Hence every forever-surviving recurrent winner satisfies

$$b_a \le V(a,x) \le V^\star(x).$$

Taking the supremum gives

$$\beta_\infty(x) \le V^\star(x).$$

Lower bound. Fix $\delta > 0$. Suppose, toward contradiction, that with positive probability

$$\beta_\infty(x) < V^\star(x) - \varepsilon_{\max} - 2\delta.$$

On this event, the recurrent solvent frontier at x is eventually below $V^\star(x) - \varepsilon_{\max} - 2\delta$. Since x is recurrent, there are infinitely many later opportunities for injection. By the entry condition, with probability at least $\lambda_\delta > 0$, a new eligible agent a' is injected with

$$V(a',x) \ge V^\star(x) - \delta.$$

Under the novice rule,

$$b_{a'} = \max_{a\in C_t} b_a + \varepsilon_{a'}, \qquad \varepsilon_{a'} \in (0, \varepsilon_{\max}],$$

where C_t is the competing eligible set when a' first competes. Since the competing frontier is below $V^\star(x) - \varepsilon_{\max} - 2\delta$, we have

$$b_{a'} \le V^\star(x) - 2\delta.$$

Therefore

$$V(a',x) - b_{a'} \ge \delta.$$

By Lemma 1, each such entrant survives forever after its first win at x with probability at least $q_\delta > 0$. Across infinitely many independent injection opportunities, the probability that no such profitable entrant survives is zero. Hence, almost surely, one such entrant eventually survives, contradicting the assumed upper bound on $\beta_\infty(x)$. Therefore

$$\beta_\infty(x) \ge V^\star(x) - \varepsilon_{\max} - 2\delta.$$

Letting $\delta \downarrow 0$ gives

$$\beta_\infty(x) \ge V^\star(x) - \varepsilon_{\max}.$$

Combining the two bounds proves the theorem. □

C.7 Proofs for Section C.2

Proof of Theorem 2. Let

$$V_t^{\mathrm{auc}}(h_t) := \mathbb{E}_{\pi^{\mathrm{auc}}}\left[\sum_{u=t}^{H-1} \gamma^{u-t} r_u^{\mathrm{out}} \,\middle|\, h_t\right],$$

and define

$$\Delta_t(h_t) := V_t^\star(h_t) - V_t^{\mathrm{auc}}(h_t).$$

At any reachable history with $E_t \neq \varnothing$, the assumption gives

$$Q_t^\star(h_t, a_t^\star) \geq \max_{u \in E_t} Q_t^\star(h_t, u) - \varepsilon.$$

After the auction winner acts, the remaining loss is the future auction suboptimality. Therefore

$$\Delta_t(h_t) \leq \varepsilon + \gamma \mathbb{E}[\Delta_{t+1}(h_{t+1}) \,|\, h_t, a_t^\star].$$

Since $\Delta_H(h_H) = 0$, backward induction yields

$$\Delta_0(h_0) \leq \varepsilon \sum_{t=0}^{H-1} \gamma^t.$$

Taking expectation over the initial history proves the theorem. □

C.8 Proofs for Section C.3

Proof of Theorem 3. Fix episode e and reachable history h_t. Let $u_{e,t}^{\mathrm{orc}}$ be the oracle-selected eligible agent, and let $u_{e,t}^{\mathrm{auc}}$ be the auction winner. By bid concentration,

$$Q_{e,t}^\star(h_t, u_{e,t}^{\mathrm{orc}}) \leq b_{e,t}(h_t, u_{e,t}^{\mathrm{orc}}) + \beta_e.$$

Because the auction maximizes bids,

$$b_{e,t}(h_t, u_{e,t}^{\mathrm{orc}}) \leq b_{e,t}(h_t, u_{e,t}^{\mathrm{auc}}).$$

Applying bid concentration again,

$$b_{e,t}(h_t, u_{e,t}^{\mathrm{auc}}) \leq Q_{e,t}^\star(h_t, u_{e,t}^{\mathrm{auc}}) + \beta_e.$$

Combining the three inequalities gives

$$Q_{e,t}^\star(h_t, u_{e,t}^{\mathrm{auc}}) \geq \max_{u \in E_{e,t}(h_t)} Q_{e,t}^\star(h_t, u) - 2\beta_e.$$

Thus, in episode e, the auction winner is $2\beta_e$-optimal at every reachable history. Applying Theorem 2 with $\varepsilon = 2\beta_e$ gives

$$r_e \leq 2\beta_e \sum_{t=0}^{H-1} \gamma^t.$$

Summing over $e = 1, \ldots, E$ proves the cumulative regret bound. If $\beta_e \leq Be^{-1/2}$, then

$$\sum_{e=1}^{E} \beta_e \leq B \sum_{e=1}^{E} e^{-1/2} \leq 2B\sqrt{E},$$

which gives

$$\mathrm{Reg}(E) \leq 4B\Big(\sum_{t=0}^{H-1} \gamma^t\Big)\sqrt{E}.$$

Dividing by E yields the average-regret rate. □

C.9 Proofs for Section C.4

Proof of Proposition 1. By assumption,

$$\mathbb{E}[Y_t \mid h_t, a_t^\star] = \gamma \mathbb{E}[V^{\mathrm{bb}}(h_{t+1}) \mid h_t, a_t^\star].$$

Substituting this identity into

$$\Delta W^{\mathrm{win}}_{a_t^\star,t} = r_t + Y_t - b_{a_t^\star}$$

and taking conditional expectations gives

$$\mathbb{E}[\Delta W^{\mathrm{win}}_{a_t^\star,t} \mid h_t] = \mathbb{E}[r_t + \gamma V^{\mathrm{bb}}(h_{t+1}) \mid h_t, a_t^\star] - b_{a_t^\star}.$$

□

Proof sketch of Theorem 4. Because the workflow is acyclic, any execution induces a topological order. Completing subtask u enlarges the completed prefix from S to $S \cup \{u\}$. Under the continuation-pricing assumption, the successor's willingness to pay equals the continuation value of the expanded feasible subgraph, up to a downstream baseline. Therefore the payment flowing back to the current agent measures the additional downstream value unlocked by executing u. This is precisely the ordered marginal contribution

$$v(S \cup \{u\}) - v(S),$$

up to the baseline term. Summing over the realized topological order yields a telescoping decomposition of the final outcome value. □

C.10 Proofs for Section C.5

Lemma 2 (Extinction of malicious agents)**.** *Suppose payoffs are recurrent, stationary, and bounded, and suppose rent $\rho > 0$ is charged infinitely often. If an agent a is malicious in the sense that there exists $\eta > 0$ such that*

$$V(a, x) \le b_a - \eta$$

at every recurrent context x where a is eligible, then a is removed in finite time almost surely.

Proof of Lemma 2. Fix an agent a. Suppose first that a wins infinitely often. Let $G_{a,k}$ be the resale-adjusted payoff on its k-th win. Its wealth after n wins satisfies

$$W_a(n) = W_a(0) + \sum_{k=1}^{n} \big(G_{a,k} - b_a\big) - \rho N_a^{\mathrm{ep}}(n),$$

where $N_a^{\mathrm{ep}}(n)$ is the number of elapsed rent-charging episodes by the time of its n-th win. For a malicious agent,

$$\mathbb{E}[G_{a,k} - b_a] \le -\eta.$$

By the strong law of large numbers,

$$\frac{1}{n} \sum_{k=1}^{n} \big(G_{a,k} - b_a\big) \to \mathbb{E}[G_{a,1} - b_a] \le -\eta \qquad \text{a.s.}$$

The rent term is nonpositive, so $W_a(n) \to -\infty$ almost surely. Thus bankruptcy occurs in finite time.

If a wins only finitely many times, then after its final win it no longer receives downstream payments. Since $\rho > 0$ and rent is charged infinitely often, rent alone eventually drives W_a below zero. Hence a is removed in finite time almost surely. □

Before proving the collusion result, we record the coalition wealth decomposition. For a nonempty set of agents S, define

$$W_S(t) := \sum_{a \in S \cap \mathcal{A}_t} W_a(t).$$

Bucket-brigade transfers between two agents in S cancel in $W_S(t)$. Therefore, over an episode segment τ, the only changes to W_S come from environment rewards, payments entering S, and

payments leaving S. If p_t denotes the predecessor of the winner at step t, with $p_t = \varnothing$ at the first winning step of an episode, then

$$\begin{aligned}\Delta W_S(\tau) = &\sum_{t\in\tau} \mathbf{1}\{a_t^\star \in S\}\, r_t \\ &+ \sum_{t\in\tau} \mathbf{1}\{a_t^\star \notin S,\ p_t \in S\}\, b_{a_t^\star} \\ &- \sum_{t\in\tau} \mathbf{1}\{a_t^\star \in S,\ p_t \notin S \cup \{\varnothing\}\}\, b_{a_t^\star} \\ &- \sum_{t\in\tau} \mathbf{1}\{a_t^\star \in S,\ p_t = \varnothing\}\, b_{a_t^\star}.\end{aligned}$$

Lemma 3 (Extinction of externally loss-making collusions)**.** *Let S be a coalition of agents that may coordinate bids, prompts, and internal transfers. If every nonempty surviving sub-coalition $T \subseteq S$ has strictly negative expected aggregate external wealth drift on recurrent episode segments, then all agents in S are removed in finite time almost surely.*

Proof of Lemma 3. Consider the currently surviving colluding set $T \subseteq S$. By assumption, whenever T is active on recurrent episode segments,

$$\mathbb{E}[\Delta W_T(\tau) \mid \mathcal{F}_{\tau^-}] \le -\eta_T$$

for some $\eta_T > 0$. The increments are bounded, so the coalition wealth W_T has strictly negative drift on recurrent segments. By the strong law of large numbers for the corresponding bounded adapted increments, W_T eventually falls below zero unless some member of T is removed earlier. Once a member is removed, the argument is restarted with the smaller surviving sub-coalition. The assumption holds for every nonempty sub-coalition that can remain active after bankruptcies, so this induction continues until no member of S survives. Hence all agents in S are removed in finite time almost surely. □

D Extended Experiment Details

This appendix provides task-level and baseline-level details omitted from the main experiment section. In the main text, we use the term *partial agent* for any agent whose capability is incomplete relative to the full task interface. Partiality can arise from explicit constraints, such as limited tools or output length, or from specialization, such as role-specific prompting. This section specifies how partiality is instantiated in each domain.

We compare EOM with complete-agent, partial-agent, and domain-specific baselines. The experiments were conducted primarily for inference-time evaluation. We use NVIDIA H200 GPUs for running local models, with at most a single H200 GPU required per experiment. In addition to local computation, we leveraged several commercial APIs, including the official APIs from OpenAI, Google Gemini, and Anthropic Claude, to evaluate model performance across different providers.

D.1 Baseline Details

Complete-agent baselines. REACT [51] is the primary complete-agent baseline. In each domain where it is used, a single agent with the same backbone model is allowed to solve the task end-to-end with access to the full action or tool space. This baseline tests whether economic organization among partial agents can compensate for capability that is concentrated in a single complete agent.

GEA [46] is also instantiated as a complete-agent baseline. Like REACT, it uses complete agents with access to the full task interface, but additionally provides a self-improvement mechanism based on experience sharing and agent evolution. This baseline tests whether the gains of EOM come specifically from market-based execution-time coordination and wealth-driven population evolution, rather than from self-improvement alone.

OPENEVOLVE [36, 28] is used for distributed-system optimization. It performs iterative program improvement with a monolithic evolutionary coding agent. In Cloudcast, OPENEVOLVE follows its standard setup with 300 iterations, while EOM runs for 30 episodes.

Partial-agent baselines. For partial-agent comparison, we use Multi-Agent Debate [11]. This baseline uses multiple agents that interact with one another, but it does not include auctions, bid-based control allocation, peer-to-peer payments, rent, bankruptcy, or wealth-based exploration and exploitation. It isolates the effect of economic coordination from the effect of merely using multiple partial agents.

We also compare against a best-of-N multi-agent baseline on Cloudcast, where N is set to the number of episodes used by EoM. This baseline tests whether improvements can be explained by repeated multi-agent sampling without market-driven evolution.

Domain-specific baselines. For accelerator design, we compare against DOSA [18], a strong non-LLM method for differentiable model-based one-loop search in DNN accelerator optimization.

D.2 Mathematical Reasoning

We evaluate on MATH [16], a competition-level mathematics benchmark whose problems are annotated from Level 1 to Level 5 by difficulty. We randomly sample 20 training problems from each difficulty level and train on an easy-to-hard stream from Level 1 to Level 5. Evaluation reports greedy pass@1 accuracy separately by difficulty level.

The population is initialized with a planner–executor–verifier decomposition. The planner proposes the immediate next step, the executor carries out the proposed step, and the verifier judges whether the executed step is correct. These agents are partial in two ways. First, each agent is role-specific and therefore sees the task through a limited functional interface. Second, each agent is given a short maximum output budget of 64–256 tokens, preventing any individual agent from solving the entire problem with a long chain-of-thought style response.

We intentionally use relatively weak LLM backbones, Llama-3.1-8B and Gemma-2-9B, to test whether economic organization can amplify partial individual capabilities. The complete-agent baseline is a single end-to-end agent with the corresponding backbone and full output/action capability.

D.3 Financial Research

We evaluate on Finance-Agent-Bench [5], a benchmark of real-world financial research tasks over company filings. The environment provides four tools. We randomly select 30 tasks for training and use the remaining 20 for testing.

Each partial agent is restricted to exactly one tool, so no individual partial agent can complete the full research task alone. The population therefore must coordinate across tool-specialized agents, including agents responsible for filing search, web search, HTML parsing, retrieval, and answer submission. We compare against Multi-Agent Debate under the same partial-agent setting, as well as complete-agent ReAct and GEA baselines that can access all tools.

For the generalist-v.s.-specialist study in Figure 7, we introduce a complete generalist agent with access to all four tools, alongside the ordinary partial specialist agents. Appendix G analyzes why this generalist does not monopolize the market.

D.4 Scientific Research

We evaluate on the *Research* subset of FrontierScience [44], which assesses open-ended scientific problem-solving. We adopt a 40/20 train–test split.

The society is initialized with four role-specialized partial agents: literature, planner, executor, and verifier. The literature agent lists possible background knowledge required for the scientific question. The planner decomposes the problem into subgoals. The executor advances one unresolved sub-question with algebraic or scientific reasoning. The verifier checks whether intermediate reasoning is valid and flags errors. These agents are partial because each is prompted to operate only within one stage of the problem-solving pipeline.

We use Gemini-3-Flash as the backbone model and compare against GEA under the same backbone. We report both mean accuracy and best-run accuracy across evaluated checkpoints during the evolution

Table 3: **Representative discovered mappings.** Example accelerator mappings discovered by EOM on representative kernels, with EDP reduction reported *relative to* DOSA. The learned solutions repeatedly recover effective output-stationary motifs. **Notation:** `L0`–`L3` index memory-hierarchy levels (`L0`=innermost registers, `L3`=outermost DRAM); `K`,`C` are output- and input-channel loops; `P`,`Q` are output spatial dimensions; `R`,`S` are filter spatial dimensions; `N` is the batch loop; suffix `X` marks a spatially parallelized loop; `[O]` marks the output-stationary level.

Kernel	EOM best mapping	EDP reduction
Kernel 1	`L3 K8 - L2 C2 K128X - L1[O] P14 Q14 C128X - L0 N1`	36.5×
Kernel 2	`L3 K2 - L2 C8 K128X - L1[O] P14 Q14 C128X - L0 N1`	23.4×
Kernel 3	`L3 K2 R3 S3 - L2 C2 K128X - L1[O] P14 Q14 C128X - L0 N1`	11.9×
Kernel 4	`L3 K32 - L2 C8 K64X - L1[O] P7 Q7 C64X - L0 N1`	8.7×
Kernel 5	`L3 R3 S3 - L2 K128X - L1[O] P28 Q28 C128X - L0 N1`	4.9×
Kernel 6	`L3 K8 P2 Q2 - L2 C2 K32X - L1[O] P28 Q28 C32X - L0 N1`	3.1×
Kernel 7	`L3 K4 R3 S3 - L2 C4 K128X - L1[O] P7 Q7 C128X - L0 N1`	2.2×
Kernel 8	`L3 K2 R3 S3 - L2 C2 K128X - L1[O] P14 Q14 C128X - L0 N1`	0.4×

process. Appendix E.1 and Appendix E.3 provide case studies showing how scientific reasoning prompts and auction-selected topologies evolve over training.

D.5 Accelerator Design

We evaluate EOM on accelerator design, following the task description in Section 3.1. The task is a discrete search problem over hardware mappings for deep learning accelerators. We use the standardized GEMMINI benchmark suite [13], which provides a hardware simulation environment for 24 ResNet-50 convolution kernels. For each kernel, the goal is to find a hardware mapping that minimizes energy-delay product (EDP), with lower EDP indicating a better design. Broader context on generative models applied across the systems stack, including DSE, can be found in [32, 41, 39].

The population contains three role-specialized partial agent types. *Historians* summarize previous trajectory trials and maintain memory of promising or failed design directions. *Planners* propose architectural or mapping-level search directions. *Executors* carry out fine-grained local evaluations. These roles are partial because each agent is responsible for only one component of the long-horizon design-space exploration loop.

We compare against the non-LLM DOSA baseline [18] and a complete REACT agent. Both LLM-based methods use the Gemma-4-31B backbone. The main metric is average EDP across the 24 kernels, with lower values indicating better designs. Figure 5 analyzes wealth dynamics during search, and Figure 7 reports per-kernel EDP improvements.

D.6 Distributed-System Optimization

We evaluate on the Cloudcast task from the ADRS benchmark [8], which minimizes total data-transfer cost in a multi-region, multi-cloud environment. We formulate Cloudcast as iterative code optimization. Each episode starts from the last successfully verified program, regardless of its score, and receives a reward at the end. This resembles test-time reinforcement learning [45, 53], where improvement attempts are made from a parent solution and updated using reward signals.

The population uses role-specialized partial agents commonly used in coding tasks. The Planner sets subgoals; the Reader inspects the codebase and summarizes findings; the Implementer edits code; the Builder compiles and tests; the Evaluator calls the verifier and records score changes; and the Finalizer submits the run. These agents are partial because each is responsible for only one stage of the coding and verification workflow.

We compare EOM with OPENEVOLVE [36, 28] and a best-of-N multi-agent baseline. All methods use GPT-5-mini. EOM runs for 30 episodes, while OPENEVOLVE uses its standard 300-iteration setup, which takes longer in wall-clock time. Appendix F traces how prompt-level action discipline and auction-selected topology evolve over a representative Cloudcast run.

D.7 Evaluation Protocol

During training, EOM alternates between within-episode planning and across-episode population adaptation. Within an episode, agents wake up according to their local predicates, compete via auctions, execute actions, and exchange payments through the bucket-brigade transaction rule. Across episodes, agents pay rent, agents with negative wealth are removed, and new agents are injected through exploration and exploitation.

During evaluation, the trained population is frozen. Bids, prompts, wealth values, and agent status are fixed; no payments, rewards, rent, births, or bankruptcies are applied. Each test task is evaluated using a thread-local copy of the trained population, and the auction mechanism is used only for action selection. Thus, reported test results measure the learned society at a fixed checkpoint rather than continued adaptation during testing.

E Case Study: Prompt and Topology Evolution in Scientific Research

This appendix analyzes the scientific research run at two coupled levels. At the micro level, economic selection evolves reusable reasoning strategies inside agents' prompts. At the macro level, these local strategies reshape the auction-selected collaboration topology. The central observation is that topology evolution is not independent of prompt evolution: once an EXECUTER internalizes checks that previously required separate VERIFIER or LITERATURE intervention, the wakeup landscape changes, and the same auction rules select a different execution path.

We focus on FrontierScience-Research, where the EXECUTER as an example, advances one unresolved sub-question with explicit algebra or scientific reasoning. Across the 40-episode training run, nine of eleven successful episodes are carried by descendants of a single evolving EXECUTER agent. Since each child inherits its parent's trainable prompt and receives only a small mutation, this agent family provides a traceable view of prompt-level evolution. The evidence is mechanistic rather than a controlled single-sentence intervention: we do not claim that a specific prompt line deterministically causes a specific answer. Instead, we show that economic selection accumulates reusable, falsifiable reasoning operators, and that these operators change both local behavior and system-level routing.

E.1 Prompt-Level Evolution: From Generic Execution to Reusable Scientific Reasoning

The initial prompt is a generic execution policy. It asks the agent to complete the earliest unfinished sub-question, show intermediate algebra, track signs and dimensions, and repair verifier-flagged errors. This is a reasonable starting point, but it does not yet specify how to expose abstract relations as auditable scalar systems, how to check whether a problem is well-posed, or how to falsify intermediate results against the original equations.

```
Identify the earliest sub-question in the plan that is not yet completed,
then carry out its derivation cleanly:
- State the sub-part label (e.g. 'Part (b):') at the top of your <step>.
- Show the intermediate algebra or physical reasoning; keep equations
  compact but include justification for non-trivial substitutions
  (limits, approximations, linearisations).
- Track signs, unit/dimension consistency, and small-parameter regimes.
- If a prior Executer turn has an error flagged by a Verifier, fix only
  that error in this turn instead of starting a new sub-part.
```

Table 4 summarizes the main prompt-level changes. The prompt does not merely become longer. It becomes more operational: each added motif specifies a concrete check or decomposition procedure that can be reused on future tasks.

The first major mutation replaces generic algebraic exposition with coordinate-based and component-wise execution. This change is distilled from a CMB trajectory in which treating C_ℓ^{TT}, C_ℓ^{EE}, and C_ℓ^{TE} as separate scalar relations proved more reliable than manipulating a single high-level covariance object. The resulting prompt requires the agent to expose all relevant variables and scalar equations, making intermediate reasoning easier for VERIFIER agents to audit.

Subsequent mutations add increasingly explicit self-auditing. The agent learns to state the governing principle before algebraic manipulation, test limiting cases or symmetries, verify boundary and

Table 4: Reusable reasoning motifs learned by the evolving EXECUTER agent. Each mutation adds a falsifiable operation rather than a task-specific fact.

Stage	Learned reasoning motif and local effect	Representative transfer
Gen. 0	Generic derivation with unit/sign checks; performs local algebra with limited self-auditing.	Initial prompt
Gen. 1	Component-wise expansion; converts abstract relations into auditable scalar systems.	Josephson junction; dark-matter detection
Gen. 2–3	Governing-principle identification, limiting cases, and constraints; anchors derivations and checks feasibility.	Gravimetry; pharmacology
Gen. 4	Equation-vs-DOF counting; detects underdetermined or overclaimed solutions.	Protein purification; constrained inference
Gen. 5	Symmetry scouting and substitute-back checks; internalizes verifier-like falsification.	^{195}Pt NMR; biology; synthesis

feasibility constraints, count independent equations against unknown degrees of freedom, and, when possible, substitute the derived expression back into the governing equations. These edits turn the EXECUTER from a generic algebraic operator into a compact scientific reasoning module.

By generation 5, the prompt has grown from four generic bullets to eight operational checks:

```
Identify the earliest sub-question in the plan that is not yet completed,
then carry out its derivation cleanly:
- State the sub-part label (e.g. 'Part (b):') and the core logic bridge
  or governing principle connecting the known variables to the target.
- Identify any symmetries, invariants, or simplified regimes that can
  be used to streamline the derivation or cross-check the result.
- Verify that the system is well-defined by comparing the number of
  independent equations to the degrees of freedom before proceeding
  with the math.
- Prioritize explicit coordinate-based or component-wise derivations;
  define all basis coefficients and variables clearly.
- Show full algebraic expansions and provide the complete system of
  scalar equations for all degrees of freedom.
- Track signs, unit/dimension consistency, and small-parameter regimes;
  sanity-check results via limiting cases, symmetries, or functional
  scaling.
- Confirm the final expression satisfies all relevant constraints,
  boundary conditions, and logical bounds; when feasible, verify by
  substituting the result back into the governing equations.
- If a prior Executer turn has an error flagged by a Verifier, fix only
  that error in this turn instead of starting a new sub-part.
```

This final prompt is not simply a more verbose instruction to "execute carefully." It specifies a reusable scientific procedure: identify the principle, check symmetries, verify well-posedness, expand explicitly, enforce constraints, and falsify the result by substitution.

E.2 Cross-Domain Transfer of the Evolved Reasoning Routine

Table 5 shows that the evolved routine transfers across domains. The agent is shaped by physics tasks, refined on chemistry and pharmacology, and later reused in spectroscopy, biology, and synthesis.

The clearest transfer occurs in episode 32, the ^{195}Pt NMR spectroscopy task. The prompt had been shaped by CMB cosmology, dark-matter detection, electrolyte chemistry, and nAChR pharmacology, none of which directly concerns hyperfine NMR shifts. Nevertheless, the inherited routine applies directly: identify the governing principle, count equations versus unknowns, check symmetry, expand the scalar relation, and substitute the result back into the observation. This supports the interpretation that the economy evolves content-independent reasoning strategies rather than memorized task solutions.

Table 5: Successful scientific-research episodes associated with the evolving EXECUTER agent. The transferable object is not factual content, but a reusable reasoning routine.

Ep.	Subject	Main reusable operation	Score
8	CMB parameter inference	Component-wise scalar decomposition	0.70
11	Josephson junction	Auditable component-wise execution	0.75
14	Dark-matter direct detection	Symmetry reduction and normalization checks	0.80
21	NaCl/KCl gravimetry	Feasibility and constraint checking	1.00
25	$\alpha4\beta2$ nAChR pharmacology	Subunit symmetry and counting	0.70
32	^{195}Pt NMR spectroscopy	Governing-principle identification and substitution	0.90
33	Protein purification	Equation-vs-DOF counting	1.00
36	$\alpha7$ nAChR homopentamer	Stoichiometric and symmetry checks	0.90
37	Pd-catalysed C–N bond	Constraint-aware synthesis reasoning	0.80
39	$\alpha4\beta2$ nAChR stoichiometry	Reuse of subunit-counting routine	0.80

E.3 From Local Reasoning Evolution to Topology Evolution

The prompt-level changes above have a system-level consequence. As the EXECUTER begins to perform dimensional checks, equation-vs-DOF counting, constraint verification, and substitute-back falsification internally, the marginal value of separate VERIFIER turns decreases in some states. Similarly, once the EXECUTER prompt begins by naming the governing principle, separate LITERATURE turns become less necessary for some tasks. Since wakeup decisions are local LLM judgments over the current workspace and each agent's evolved prompt, these internalized routines directly change who wakes up, who bids, and who acts.

We define an episode's topology as the ordered sequence of auction winners, represented at both the role level and the agent-identity level. This topology is not a pre-specified execution graph. It is reselected at every step through local wakeup decisions and bid competition.

E.4 Early Regime: Explicit Multi-Role Auditing

Early in training, successful trajectories often rely on explicit multi-role auditing. Episode 11, the Josephson-junction task, is the longest successful trajectory in the run. It obtains a score of 0.75 with ten auction steps and all five roles:

$$\text{LIT.} \to \text{PLAN} \to \text{EXE.} \to \text{VER.} \to \text{EXE.} \to \text{VER.} \to \text{PLAN} \to \text{EXE.} \to \text{VER.} \to \text{ANS.}$$

This trajectory is long, but it is structured. Literature supplies background, Planner decomposes the problem, Executer advances sub-parts, Verifier audits intermediate executions, Planner re-scopes after the audits, and Answer synthesizes the final response. No agent designs this workflow globally. It emerges from local wakeup judgments, novice bidding, same-role blocking, and the terminal answer restriction.

This early topology compensates for immature local reasoning. Because the EXECUTER has not yet internalized enough verification, the society externalizes uncertainty through repeated execute–verify loops.

E.5 Late Regime: Compact Specialist Execution

Later in training, the same auction mechanism can select a much shorter specialist workflow. Episode 33, the protein-purification task, receives a score of 1.0 with only three steps:

$$\text{PLAN} \to \text{EXE.} \to \text{ANS.}$$

This contraction is not caused by a smaller population. The population contains 14 agents, including living LITERATURE and VERIFIER agents. These agents still run their wakeup judges, but return NO. The reason is not a change in the auction code; it is the evolved prompt state. `Executer` already performs dimensional checks, equation-vs-DOF counting, constraint verification, and substitute-back falsification. As a result, separate verifier intervention has lower marginal value for this episode.

E.6 Training-Wide Topology Trend

Table 6 summarizes the successful trajectories. Later episodes often use compact 3–4 step paths despite larger populations. Long paths remain when the task still benefits from explicit literature or verifier intervention, showing that topology is adaptive rather than simply pruned.

Table 6: Auction-derived topology for successful scientific-research episodes. Compact paths become common later in training, even though the population remains large.

Ep.	Subject	Auction-derived path	Roles	Pop.	Score
8	CMB inference	Plan → Exe. → Ans.	3	10	0.70
11	Josephson junction	Lit. → Plan → Exe. → Ver. → Exe. → Ver. → Plan → Exe. → Ver. → Ans.	5	12	0.75
14	Dark-matter detection	Lit. → Exe. → Ans.	3	12	0.80
21	NaCl/KCl gravimetry	Exe. → Ver. → Exe. → Ver. → Ans.	3	17	1.00
25	$\alpha4\beta2$ nAChR	Exe. → Ans. → Exe. → Ans.	2	15	0.70
32	^{195}Pt NMR	Lit. → Plan → Exe. → Ans.	4	12	0.90
33	Protein purification	Plan → Exe. → Ans.	3	14	1.00
36	$\alpha7$ nAChR homopentamer	Lit. → Ver. → Lit. → Plan → Exe. → Ver. → Plan → Lit. → Exe. → Ans.	5	14	0.90
37	Pd-catalysed C–N bond	Lit. → Plan → Exe. → Ans.	4	14	0.80
39	$\alpha4\beta2$ nAChR redux	Lit. → Exe. → Ans.	3	16	0.80

The key point is that shorter paths are not a consequence of having fewer available agents. The market often has more agents to choose from, but selects fewer active roles because competence has concentrated in load-bearing agents. Conversely, when residual uncertainty remains high, as in episode 36, the auction still selects longer literature and verification chains. Thus, topology evolution is conditional and task-sensitive.

E.7 Population-Level Selection

The population graph evolves in parallel with the execution graph. Births are not uniformly distributed across roles. The EXECUTER agent deepens through descendants because it repeatedly appears on reward-bearing trajectories. The ANSWER agent family also broadens as final synthesis becomes valuable once upstream reasoning becomes reliable. In contrast, LITERATURE and PLANNER remain comparatively shallow in this run.

This reflects the economic feedback loop. Agents that create downstream value accumulate wealth, survive longer, and become more likely to seed future mutations. Therefore, the economy selects not only who acts in the current episode, but also where future evolutionary capacity is allocated. The central evidence of evolution is therefore not merely that later agents solve more tasks. It is that the market converts repeated successes and failures into reusable reasoning routines, and these routines reorganize the society's workflow without centralized orchestration.

F Case Study: Prompt and Topology Evolution in CloudCast

This appendix provides evidence that the same economic mechanism also works in a code-optimization environment. In Scientific Research, the main learned object is a reusable reasoning routine. In CloudCast, the main learned object is an action discipline: agents learn when *not* to spend an expensive action. The topology adapts accordingly. When the workspace is close to a new high score, the auction selects a short read–edit–evaluate–commit path; when the workspace still contains uncertain regressions or unresolved implementation choices, the auction selects longer edit–build–evaluate loops. Thus, the economy does not simply shorten workflows. It selects a workflow whose shape matches the current workspace state.

CloudCast asks the society to evolve a single Python file, `initial_program.py`, implementing a multi-cloud broadcast routing algorithm. Given source and destination regions and a network graph with edge costs and throughputs, the program must produce a `BroadCastTopology`. The simulator scores the program by total egress cost across five inter- and intra-cloud scenarios. The seed single-path Dijkstra implementation costs approximately $\$1035$, and the reported score is the

fractional reduction $\max(0, 1 - \text{cost}/1035)$. The workspace persists across episodes, so each episode continues from the previous program rather than starting from scratch.

F.1 Auction-Derived Topology

The CloudCast implementation contains six roles: READER, PLANNER, IMPLEMENTER, BUILDER, EVALUATOR, and FINALIZER. The READER inspects files, the PLANNER proposes sub-goals, the IMPLEMENTER edits code, the BUILDER runs a build or import check, the EVALUATOR calls `request_eval()`, and the FINALIZER submits the program via `final_answer`. At each step, every living agent runs a wakeup judge; eligible agents submit fixed bids, and the highest bidder acts. CloudCast does not use same-role blocking, so consecutive same-role turns are emergent rather than forbidden. The only structural restriction is terminal: only FINALIZER-tagged agents may bid near the end.

Table 7 summarizes the completed checkpoint episodes. Unlike the scientific-research case, path length does not monotonically decrease. This is expected: because the codebase persists, some episodes are one edit away from improvement, while others require multi-edit search or regression repair.

Table 7: Auction-derived topology for completed CloudCast checkpoint episodes. Superscripts denote consecutive same-role turns. Short paths occur when the persisted workspace is close to a new high; long paths occur during multi-edit or regression-repair phases.

Ep.	Score	Auction-derived topology	Steps	Roles
0	0.015	$R \to P \to I \to E \to F$	5	5
4	0.015	$R \to I \to B^2 \to I^4 \to B^3 \to E \to F$	13	5
9	0.107	$R \to I \to E \to F$	4	4
12	0.107	$R \to E \to I \to B \to E \to I \to B \to I \to B \to E^2 \to F$	12	4
15	0.119	$P \to R \to E \to F \to I^2 \to B^2 \to E \to I \to B \to E \to I^6 \to B \to E \to F^2$	22	5
20	0.154	$R \to E \to P \to I \to B^2 \to E \to F$	8	5
21	0.267	$P \to R \to I \to B \to E \to I \to E \to F^6$	13	5
26	0.328	$P \to R^2 \to I^3 \to B \to E \to F$	9	5
27	0.365	$R \to I \to B \to E \to I^2 \to E \to I \to B \to I \to E \to F$	12	5
28	0.365	$R \to I \to E \to B \to I \to E \to F^3$	9	5

The table shows two patterns. First, topology length tracks residual uncertainty rather than training time. Episode 9 is a clean one-edit improvement and collapses to four steps. Episode 15 is a partition-sweep phase and stretches to the full step budget. Second, the role mix becomes more selective. Late episodes often rely on a READER/IMPLEMENTER/EVALUATOR/ FINALIZER core, while PLANNER and BUILDER appear when the workspace state calls for them. The auction is unchanged; the local wakeup landscape changes as prompts and the workspace evolve.

Episode 27 is the peak checkpoint, with score 0.365, or 36.5% cost reduction relative to the Dijkstra seed ($657 vs. $1035). Its topology is:

$$R \to I \to B \to E \to I^2 \to E \to I \to B \to I \to E \to F$$

This path is not manually scripted. The READER stops after summarizing the relevant file, the IMPLEMENTER continues while verifier feedback names unresolved scenarios, the BUILDER wakes up when static validity is uncertain, the EVALUATOR wakes up when a score-changing edit is plausible, and the FINALIZER commits only when further iteration appears unlikely to help.

F.2 Prompt Evolution: Checks Before Costly Actions

The topology changes because the agents' prompts change. Across the run, mutations occur in four roles: EVALUATOR, BUILDER, IMPLEMENTER, and FINALIZER. The unifying pattern is simple: each role learns a cheap structural check before its expensive action. The EVALUATOR checks markers before `request_eval()`, the BUILDER checks symbols before running a build command, the IMPLEMENTER states intent and re-reads after `write_file`, and the FINALIZER rechecks invariants before `final_answer`.

Table 8: Prompt mutations in CloudCast. Each mutation adds an operational check at the point where the previous failure surfaced. The learned content is not a new routing heuristic, but a better discipline for deciding when to act.

Gen.	Born	Mutated agent	New behavior	Trigger
0	ep. 0	6 founders	—	Initial population
1	ep. 2	`Evaluator-9163`	Cheap pre-eval checks before `request_eval()`	Eval budget spent on $\Delta = 0$ calls
2	ep. 5	`Evaluator-8866`	Check markers, signatures, and required classes	Refactor broke structural markers
3	ep. 8	`Builder-4001`	Static symbol-presence check before build	Broken-import builds wasted steps
4	ep. 13	`Builder-3586`	Targeted test selection and actionable diagnostics	Generic build checks missed semantic regressions
5	ep. 19	`Evaluator-5245`	Per-scenario sign prediction before evaluation	Aggregate $\Delta \approx 0$ hid scenario regressions
6	ep. 22	`Finalizer-1195`	Re-audit structural invariants before commit	Broken invariant was committed
7	ep. 25	`Implementer-8828`	State edit intent and re-read after writing	Good behavior observed in recent thoughts
8	ep. 26	`Implementer-9766`	Prefer smaller edits when uncertain	Combined edits were hard to debug

The initial EVALUATOR prompt is mostly a wakeup rule:

```
Wake up whenever Implementer has written new code since the last eval.
If the task uses pure-Python code, you can eval directly without waiting
for a Builder green. Do not eval twice in a row without new code in
between, and do not eval on a build/import failure.
```

After early $\Delta = 0$ evaluations, the first mutation adds a pre-evaluation validation layer:

```
Do: run quick local checks (syntax/import, lightweight tests, and a
schema/validator smoke test), use inexpensive heuristics to estimate
whether the change is likely to yield a meaningful improvement, and
only call request_eval when those checks and the estimate indicate a
real chance of improvement.
```

A later mutation makes this check more structural:

```
Before calling request_eval():
- Do: run a quick compile/import check, run lightweight unit/smoke
  tests or static sanity checks, verify required interfaces/markers
  remain intact, and run a deterministic local heuristic/estimate
  that justifies an expected improvement; document the rationale for
  why an eval is likely to change the score.
```

This illustrates the general mechanism. The EVALUATOR does not learn a new optimization algorithm. It learns to avoid wasting expensive evaluation calls. The same motif then appears in other roles: BUILDER learns static checks before build commands, IMPLEMENTER learns intent-and-verify editing, and FINALIZER learns to recheck invariants before committing. Thus, prompt evolution changes the wakeup decisions, and the changed wakeup decisions reshape the auction-selected topology.

G Why the Generalist Does Not Monopolize

This appendix studies a potential failure mode of the economy: if one agent is given access to all tools, will it monopolize the market and eliminate specialization? In Finance-Agent-Bench, we introduce a *full-tool generalist* agent with access to all four tools (`edgar_search`, `web_search`,

parse_html_page, retrieve_information), alongside the ordinary specialized agents. At first glance, one might expect such an agent to dominate the society. It has the broadest action space, the least tool-level restriction, and in principle can subsume filing search, web search, document parsing, retrieval, and answer submission. Yet empirically it does not monopolize control. Specialized agents remain persistently active and frequently win auctions.

This failure to monopolize is informative: in our economy, being more general is not the same as being more competitive. The generalist must spread its prompt budget across heterogeneous requirements: tool formatting, decomposition, temporal coverage, accounting consistency, numerical verification, cross-source reconciliation, and final answer submission. As a result, its evolved prompt becomes broad and procedurally cautious rather than sharply discriminative. This is useful for coverage, but economically costly. A generalist that tries to be prepared for everything is often outbid by a specialist whose wakeup condition, search pattern, and evidence standard are tuned to one narrower subproblem. The generalist does not fail because it is weak; it fails to monopolize because it is too general. The society rewards agents that are locally precise, not globally omnibus.

G.1 Initial Generalist Prompt

The initial generalist prompt is intentionally broad. It enforces tool-call correctness and a basic search-first workflow, but it does not prescribe domain-specific decomposition, numerical discipline, or an evidence hierarchy beyond a generic sequence of actions.

```
TOOL-CALL FORMAT RULES (do not violate):
- edgar_search: pass CIK numbers as strings, e.g. ciks=['0000002488'].
- web_search: call web_search(search_query="..."), NOT web_search(query="...").
- parse_html_page: pass a valid URL string.
- retrieve_information: query stored documents by key.
- final_answer: call final_answer(answer="...") to submit.
Work methodically: search first, retrieve details, then answer. Do ONE
tool call per turn. When evidence is sufficient, submit immediately.
```

This initial version is a broad operator's manual. It ensures syntactic correctness and a sensible high-level order of operations, but it does not yet tell the agent how to reason under financial query structure. In particular, it does not force explicit decomposition of the question, distinguish intermediate from final evidence, align fiscal periods, or guard against subtle financial errors such as mixing segment-level values with consolidated totals.

G.2 Evolved Generalist Prompt

After evolution, the generalist prompt becomes substantially more elaborate. The added instructions do not specialize it around one micro-skill. Instead, they accumulate global requirements for acting as a competent all-purpose financial analyst.

```
TOOL-CALL FORMAT RULES (do not violate):
- edgar_search: pass CIK numbers as strings, e.g. ciks=['0000002488'].
- web_search: call web_search(search_query="..."), NOT web_search(query="...").
- parse_html_page: pass a valid URL string.
- retrieve_information: query stored documents by key.
- final_answer: call final_answer(answer="...") to submit.
Work methodically: Decompose the query into discrete requirements. For
multi-year trends, report data for every intermediate interval and
prioritize consolidated/total figures for the primary entity over
segmented or regional data. Trace every numerical value directly to raw
tool outputs; do not adopt figures from dialogue summaries without
verification. Ensure consistent accounting definitions across all periods.
Cross-reference sources to resolve discrepancies. Perform a sanity check
on the magnitude and trend before submitting. Do ONE tool call per turn.
Submit immediately once the full temporal and categorical scope is
satisfied.
```

Relative to generation 0, the mutation improves rigor but does not make the agent sharper about one task family. It adds a broad bundle of global cautionary rules: decompose the query into discrete requirements, cover every relevant time interval, prefer consolidated totals over segment or regional

figures, trace numbers back to raw tool outputs, enforce accounting consistency, cross-check source discrepancies, and sanity-check trends and magnitudes.

These additions make the generalist safer, but they also explain why it does not monopolize. The prompt becomes a general compliance layer rather than a highly tuned specialist heuristic. The evolved generalist is taught to be careful everywhere, rather than exceptional in one recurring local decision.

G.3 Comparison to Specialized Agent Evolution

The contrast is clearest when compared with the specialized EDGAR and TAVILY agents. Their initial prompts are narrow: the EDGAR agent is only asked to find relevant filings, and the TAVILY agent is only asked to perform one targeted web search. After evolution, both remain narrow, but become much more exacting about the specific failure modes of their own tool domains.

Edgar agent. The evolved EDGAR prompt does not try to become a universal analyst. It sharpens around filing-specific correctness: identifying the exact entity, filing type, and fiscal period; distinguishing aggregate totals from plan- or segment-specific values; checking filing date and fiscal year; locating future-dated projections inside the latest official filing; and matching the numerical answer to the exact qualifiers of the query. This is a specialization trajectory: the prompt is refined to avoid a small set of high-value, filing-specific mistakes.

Tavily agent. Likewise, the evolved TAVILY prompt becomes more disciplined about source reliability and arithmetic. It learns to identify the knowledge gap before search, trace numerical claims back to primary evidence, avoid substituting non-matching periods or metrics, cross-reference sources, and recompute arithmetic from raw values. Again, this is not a move toward generality. It is a move toward narrower epistemic discipline within one tool domain.

By contrast, the full-tool generalist must absorb all of these concerns at once. Its prompt therefore grows by accumulating heterogeneous global obligations rather than repeatedly repairing one local competence. Economically, this matters because auctions reward agents that are locally high-value under a particular state. A specialist can be the best agent precisely when its narrow prompt applies. The generalist is broad, but its local edge is diluted.

G.4 Generalist Failure as Evidence for Specialization

This case study illustrates a general principle of the economy: prompt evolution is most effective when repeated feedback can be compressed into a small number of role-specific decision rules. When an agent has a narrow tool interface and a narrow evidentiary responsibility, the mutator can improve it by writing concrete instructions that target the same class of mistakes repeatedly. The resulting prompt becomes sharper, more falsifiable, and more economically competitive.

The generalist does not enjoy this advantage. Because it is responsible for too many heterogeneous subtasks, each mutation tends to add another layer of generic caution rather than strengthening one specific inference pattern. Its prompt becomes longer, broader, and more procedurally safe, but not more economically dominant. Thus, the generalist does not monopolize: over-generality is not specialization, and in this market, specialization is what wins control.

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: Section 3

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: Section 5

   Guidelines:

   - The answer [N/A] 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: Appendix C

Guidelines:

- The answer [N/A] 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: Section 3, Appendix D

   Guidelines:

   - The answer [N/A] 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]

Links available on the first page.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 3, Appendix D

   Guidelines:

   - The answer [N/A] 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: Section 3

   Guidelines:

   - The answer [N/A] 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: Appendix D

   Guidelines:

   - The answer [N/A] 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, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] 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: Section 5

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our paper poses no such risks.

    Guidelines:

    - The answer [N/A] 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: Section 3

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: Our paper does not release new assets.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: We do not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: We do not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.