Co-Evolving LLM Decision and Skill Bank Agents for Long-Horizon Tasks

Xiyang Wu
University of Maryland
United States
wuxiyang@umd.edu

Zongxia Li
University of Maryland
United States
zli12321@umd.edu

Guangyao Shi
University of Southern California
United States
shig@usc.edu

Alexander Duffy
Good Start Labs
United States
alex@goodstartlabs.com

Tyler Marques
Good Start Labs
Canada
tyler@goodstartlabs.com

Matthew Lyle Olson
Independent Researcher
United States
matthewlyleolson@gmail.com

Tianyi Zhou
Mohamed bin Zayed University of Artificial Intelligence
United Arab Emirates
tianyi.david.zhou@gmail.com

Dinesh Manocha
University of Maryland
United States
dmanocha@umd.edu

Abstract

Long-horizon interactive environments are a natural testbed for evaluating agents' skill-usage abilities, since they demand multi-step reasoning, the chaining of multiple skills over many timesteps, and robust decision-making under delayed rewards and partial observability. Games are a particularly diverse and reproducible instance of such environments, providing a controllable setting for studying multi-skill long-horizon behavior. Large Language Models (LLMs) offer a promising alternative as game-playing agents, but they often struggle with consistent long-horizon decision-making because they lack a mechanism to discover, retain, and reuse structured skills across episodes. We present **COS-PLAY**, a co-evolution framework in which an LLM decision agent retrieves skills from a learnable skill bank to guide action taking, while an agent-managed skill pipeline discovers reusable skills from the agent's unlabeled rollouts to form a skill bank. Our framework improves both sides jointly: the decision agent learns better skill retrieval and action generation, while the skill bank agent continually extracts, refines, and updates skills together with their contracts. Experiments across six game environments show that **COS-PLAY** with an 8B base model achieves over 25.1% average reward improvement against four frontier LLM baselines on single-player game benchmarks while remaining competitive on multi-player social reasoning games.

Keywords: LLM agents, agent skills, skill bank, long-horizon decision making, co-evolution, reinforcement learning, GRPO, game playing

Agent Skills '26, San Jose, CA, USA
2026. ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

A central goal of artificial general intelligence is to build autonomous agents that improve through interaction by discovering reusable *skills*, *i.e.*, reusable, temporally extended behavior protocols extracted from trajectories, and invoking them to solve increasingly complex tasks. This paradigm has a long history, spanning self-improving game agents in Atari and AlphaZero [29, 40] to modern LLM agents for coding, web interaction, and embodied exploration [7, 13, 43, 48]. Across these settings, self-learning has expanded from structured games to richer interactive environments, while the core explore–learn–reuse loop remains fundamental to autonomous improvement. As agents operate in increasingly complex domains, the challenge shifts from merely acquiring useful skills to organizing, refining, and retrieving them so they can be reused reliably across future tasks.

A natural way to address this challenge is to maintain an external *skill bank*: a structured library of reusable behavior protocols that the agent can acquire, refine, and compose over time [18]. Recent work increasingly treats *skills* as a distinct abstraction layer for agentic LLMs beyond one-off tool calls or prompts, since skills package reusable procedural knowledge together with execution guidance and applicability constraints [16, 50]. Rather than repeatedly retraining the backbone LLM, the system improves by updating this skill bank and learning to invoke the right skills at the right time. Yet these two components are tightly coupled: a skill bank is only useful if the decision agent can select and execute its skills effectively, and the decision agent is only as capable as the skills available to it. This interdependence motivates two questions: **(Q1)** Should the skill bank and decision agent be learned *jointly*, rather than separately? **(Q2)** What properties make a skill bank useful for improving decision-making?

To study these questions, we need an environment that reveals co-evolution dynamics: episodes must be long enough to require *multiple* skills in sequence, subgoals must *recur* across episodes so that reuse is meaningful, and rollouts must be cheap enough to support iterative skill-bank refinement. Games offer such a setting in a safe and reproducible sandbox [2, 36, 41, 51], while still demanding long-context understanding, memory, and adaptive decision-making [12]. At the same time, they pose challenges that make naive approaches insufficient: delayed rewards [25], compositional strategies, and limited high-quality demonstrations or skill annotations [22, 26]. Consequently, many existing game-playing agents still rely on curated human data [10] or prolonged training with external feedback, whereas we instead aim to learn a co-evolving skill bank directly from interaction.

In this paper, we propose a **multi-agent co-evolution** framework for long-horizon game playing. The framework comprises: (i) an LLM-based **decision agent** that maintains an intention and active skill, retrieves or switches skills as needed, and executes primitive actions conditioned on both; and (ii) a **skill bank agent** that performs unsupervised skill discovery and maintenance by segmenting trajectories, learning compact skill contracts, and refining the skill bank over time. The two components improve each other in a closed loop: the decision agent uses the current skill bank for multi-step, skill-guided decision-making, while the skill bank agent converts newly collected rollouts into better reusable skills. We optimize both agents with Group Relative Policy Optimization (GRPO) [38]: the decision agent is trained to improve skill retrieval and action execution, while the skill bank agent is trained for skill segmentation, contract learning, and bank curation. To our knowledge, this is one of the first frameworks to couple LLM-based game decision-making with an agentic skill-bank pipeline for unsupervised skill discovery and continual refinement in a unified co-evolution loop. Our main contributions are:

- We propose an agentic skill-bank pipeline for construction and maintenance that transforms unlabeled trajectories into reusable skills via boundary proposal, segmentation, contract learning, and bank curation, enabling skill-guided long-horizon decision making across diverse game environments;
- We propose a co-evolution framework for long-horizon gameplay that closes the loop between decision making and skill learning: an LLM-based decision agent interacts with the environment to collect trajectories, and a skill bank agent converts these rollouts into reusable skills that improve future decision making;
- We provide a comprehensive evaluation of COS-PLAY across six game environments requiring multi-hop skill usage, covering both task performance and skill reusability. Built on an 8B base model, COS-PLAY achieves over **25.1%** average gain against four frontier LLM baselines on single-player games while remaining competitive with SOTA LLMs on multi-player social reasoning tasks.

2 Related Work

Agents for Game Playing and Benchmarks. Game agents and benchmarks are useful for studying long-horizon decision-making because they stress memory, planning, and real-time action under delayed rewards and complex dynamics. However, despite rapid progress, relatively few works study game-playing agents at scale due to the difficulty of interactive environments and evaluation. BALROG [33], VS-Bench [52], VideoGameBench [58], and VisGym [45] introduce increasingly challenging benchmarks for long-horizon reasoning, strategic interaction, and multi-step visual decision-making, revealing large gaps between frontier models and human performance. On the method side, Optimus-1 [20], VARP [4], and AVA [27] develop agents for Minecraft, ARPGs, and StarCraft II, highlighting the importance of skill exploration, memory, and planning in complex gameplay [57]. These advances motivate our framework, which uses games as a controlled setting for co-evolving decision policies with a refineable skill bank. Rather than focusing only on benchmarks or game-specific agents, we study how reusable skills can be extracted from trajectories, refined over time, and fed back to improve long-horizon decision-making across diverse games.

Memory and Skill-Augmented Self-Improving Agents. Memory- and skill-augmented agents have gained attention because long-horizon decision-making benefits from reusing past experience and executable skills instead of solving each step from scratch, especially in complex multimodal tasks. Recent work improves self-evolving agents through memory and skill reuse [13, 18, 19, 21]. PolySkill [55] improves skill transfer by decoupling abstract goals from concrete implementations. SAGE [42], SkillRL [47], SCALAR [56], and XSkill [15] use skill-augmented learning to improve skill generation, grounding, composition, retrieval, and policy performance. Memweaver [54] introduces a hierarchical dual-memory framework combining graph-based retrieval with LLM-summarized cognitive memory. ProcMEM [28] and CASCADE [14] emphasize procedural memory and cumulative skill creation. UI-Mem [49] studies hierarchical experience memory for long-horizon online RL, while MemRL [60] and MemSkill [59] treat memory retrieval and operations as runtime mechanisms for improving frozen agents. These works motivate our framework, but most of them focus on memory augmentation, skill transfer, or runtime retrieval within largely fixed pipelines. In contrast, our method studies co-evolution, where a decision agent and skill bank improve each other over time by automatically extracting, refining,

and reusing skills from unlabeled trajectories to improve downstream decision-making.

3 Problem Formulation

Preliminary. We consider an interactive game environment $\mathcal{E} = (\mathcal{O}, \mathcal{A}, P, R, \gamma, T)$ with horizon T, where $\mathcal{O}$ is the observation space, $\mathcal{A}$ is the action space, $P(o_{t+1} \mid o_t, a_t)$ denotes the environment transition dynamics, $R(o_t, a_t, o_{t+1}) \in \mathbb{R}$ is the reward function, and $\gamma \in (0, 1]$ is the discount factor. At each timestep t, an agent observes $o_t \in \mathcal{O}$, takes an action $a_t \in \mathcal{A}$, receives reward $r_t \in \mathbb{R}$, and transitions to the next observation o_{t+1}. A trajectory (episode) is denoted by $\tau = (e_1, e_2, \ldots, e_T)$, where each step is represented as an experience

$$e_t = (o_t, a_t, r_t, o_{t+1}, d_t, z_t)$$

Here $d_t \in \{0, 1\}$ is the termination flag, and z_t is the agent's latent intention state at timestep t, representing its internal strategic interpretation of the environment [35, 46]. This latent intention may correspond to either a short-horizon action intention for the next step or a higher-level strategic skill that guides behavior over the next several steps. In our framework, z_t is generated from the agent's understanding of the current environment state and is iteratively updated throughout the trajectory as the agent interacts with the environment and refines its strategic objective.

Skills. A skill s_k is a reusable, temporally extended behavior abstraction extracted from trajectory segments. Each skill is stored in the skill bank as a structured *skill protocol* with the following components: *Summary*, which describes the skill's purpose; *Pre-condition*, which specifies when the skill is applicable; *Plan*, which outlines how the skill should be carried out; *Success/Abort Criteria*, which indicate when execution should terminate successfully or be abandoned; and *Contract*, which summarizes the state changes the skill is expected to produce. These components form the skill's externally queryable interface for retrieval and execution, while the skill itself remains grounded in the set of supporting trajectory segments from which the protocol and contract are learned and refined over time.

Refineable Skill Bank. A key property of our framework is that skills are not fixed after initialization. Instead, as interactive learning between the agents and the environment continues, the skill bank is continuously updated and refined with new trajectory evidence. As new rollouts are collected, the skill bank agent updates the bank by segmenting trajectories, refining existing skills, and storing emerging skills.

Skill-augmented Decision-making. Given the current skill bank $\mathcal{B}$, the decision agent maintains an intention state z_t that captures its current strategic focus and skill-level subgoal. At each timestep, it first retrieves candidate skills from the bank, selects one based on the current observation and intention, updates its intention conditioned on the selected skill, and finally executes an action. Formally,

$$\tilde{s}_t = \pi_\theta^{\text{skill}}(o_t, \mathcal{B}), \tag{1}$$

$$z_t = \pi_\theta^{\text{int}}(o_t, \tilde{s}_t), \tag{2}$$

$$a_t \sim \pi_\theta^{\text{act}}(\cdot \mid o_t, z_t, \tilde{s}_t). \tag{3}$$

We model the decision agent with three components serving distinct roles: $\pi_\theta^{\text{skill}}$ for skill retrieval-and-selection, π_θ^{int} for intention updating, and π_θ^{act} for action execution. Here, $\tilde{s}_t$ denotes the retrieved skill or skill set at timestep t, and z_t denotes the updated intention state conditioned on the current observation and retrieved skill. The agent is trained to maximize the expected cumulative reward:

$$\max_\theta \; J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=1}^{T} r_t\right]. \tag{4}$$

Skill Bank Update. We formulate skill-bank refinement iteratively because the skill bank and decision agent co-evolve over training: each round of newly collected trajectories reflects the current policy, while the updated bank shapes subsequent skill retrieval and action execution. The trajectory set $\mathcal{D}^{(u+1)}$ collected up to co-evolution iteration $u+1$ is passed to the skill bank agent to refine the current skill bank $\mathcal{B}^{(u)}$:

$$\mathcal{B}^{(u+1)} = \Phi_{\text{S}}\left(\mathcal{B}^{(u)}, \mathcal{D}^{(u+1)}\right), \tag{5}$$

where $\mathcal{B}^{(u)}$ and $\mathcal{B}^{(u+1)}$ denote the skill bank before and after the update, $\mathcal{D}^{(u+1)}$ denotes the accumulated rollout set, and Φ_{S} denotes the skill-bank update pipeline. This pipeline continually updates the skill bank so that retrieved skills remain reusable and aligned with the evolving decision policy.

4 Methodology

We propose a co-evolving multi-agent framework for long-horizon video-game decision-making via unsupervised trajectory decomposition and skill-bank refinement (Figure 1). The framework has two components: **(a) Decision Agent** A_D: an LLM-based agent that interacts with the game through primitive actions and skill retrieval. At each step, it summarizes the current state, retrieves relevant skill candidates from the skill bank, updates its intention, selects or switches skills when needed, and executes an action. **(b) Skill Bank Agent** A_S: an LLM-based pipeline that converts unlabeled trajectories into reusable protocol-based skills and learns compact effect contracts for them. It updates the skill bank by proposing new skill candidates, refining low-quality skills, and revising skill protocols over time. Together, the decision agent generates trajectories, and the skill bank agent transforms them into structured skills that support future decisions through skill retrieval and selection.

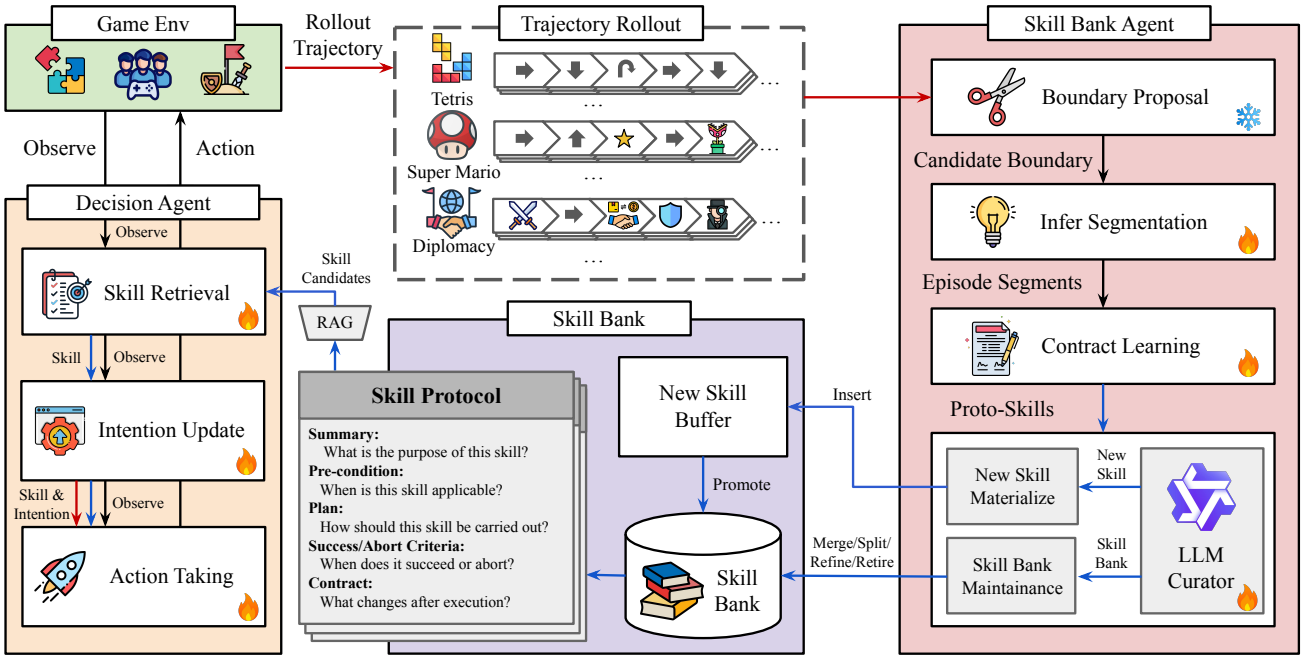

Figure 1. Overview of COS-PLAY. COS-PLAY is a multi-agent co-evolution framework that couples gameplay with skill learning. It consists of a decision agent (**Orange Box**), a skill bank agent (**Red Box**), and a skill bank (**Purple Box**). The decision agent interacts with the game by retrieving skills, updating intentions, and selecting actions. After each episode, the skill bank agent segments trajectories, learns skill contracts, adds new skills, and updates existing ones through refinement, merging, splitting, or retirement. This loop enables the agent to acquire reusable skills from prior experience and continually improve its policy.

4.1 Skill-Augmented Decision Agent

We design a skill-augmented decision agent that unifies skill retrieval, intention updating, action execution, and reward-based learning for long-horizon gameplay. At each step, it maintains a state summary, intention, and active skill; either continues the current skill or retrieves a new one; and executes a primitive action conditioned on the observation, intention, and skill plan. The resulting observation and reward are used to update the agent state, while training is driven by a composite reward combining environment feedback, skill-following shaping, and switching cost.

Skill Retrieval and Intention Update. The decision agent maintains an active skill and a current intention. Given the current state summary, it either continues the active skill or retrieves a new one from the skill bank when the current skill is unavailable, exhausted, or ineffective. Each skill is a structured context prompt containing protocol steps, preconditions, and termination cues. After each primitive action, the agent updates a short natural-language intention tag summarizing its immediate subgoal. When this intention shifts sharply, it signals a subtask transition and triggers skill switching [17].

Action Execution. At each timestep, the agent executes a primitive environment action conditioned on the current state summary, task context, recent interaction history, the current intention, and, when available, the active skill plan. The active skill does not replace low-level control, but instead guides action generation toward coherent multi-step behavior. The executed action produces the next observation and reward, which are then used to update the agent state and intention.

4.2 Skill Bank Agent for Skill Discovery and Maintenance

With decision agent rollouts, a skill bank agent analyzes the trajectory to extract reusable skills and maintain the skill bank over time. This agent has a four-stage pipeline: proposing candidate skill boundaries, inferring a segmentation with skill labels, learning and verifying effect contracts, and finally updating the bank through refinement, materialization, merging, splitting, and retirement. In this way, the bank evolves from repeated trajectory evidence while remaining compact, stable, and useful for downstream decision making. Figure 2 walks through the full pipeline on a single Diplomacy episode.

Boundary Proposal. Boundary proposal is a heuristic candidate generation step that identifies plausible skill-transition

points in a trajectory. For each timestep, we compute a boundary score from lightweight local signals that often indicate a change of skill, including predicate flips between adjacent states, intention-tag changes, reward or event spikes, optional surprisal peaks, and transitions between primitive-action execution and skill-selection steps. Timesteps with high scores are retained as candidate cut points, after which nearby candidates are merged to remove redundancy. This produces a compact, high-recall boundary set that is later disambiguated by the segmentation module.

Infer Segmentation. Starting from the candidate boundaries, we choose the subset that best explains the trajectory as a sequence of skill segments. Each segment is summarized by its observed effects and compared against skills in the current bank. We first score candidate skills by overlap between the segment's added and deleted predicates and each skill's learned *effect contract*, *i.e.*, a compact specification of the state changes that the skill reliably produces when executed successfully, with a small bonus for stronger prior support, and then re-rank the top candidates using the skill bank agent. If a segment matches an existing skill with high confidence, we assign that label. Otherwise, we mark it as *new skill*, indicating behavior not yet represented in the bank. Its observed effects are kept as provisional evidence, and a reusable effect contract is learned later by aggregating and verifying multiple such segments in the contract learning stage.

Contract Learning. For each skill, we aggregate the added and deleted predicates across its decoded segments to learn an *effect contract* that captures its reliable state changes. We keep only consensus effects that appear consistently across instances, treating infrequent ones as noise. When enough evidence is available, the contract can be enriched by an LLM-based summarization module, whose suggestions are added to the statistical consensus. We then verify whether the contract effects are supported by observed state changes, and only contracts with sufficiently high pass rates are written back into the skill bank. These verified contracts also provide a compatibility prior for segment decoding, giving higher confidence to segments whose observed effects better match a skill's contract.

Skill Bank Maintenance. The final stage mutates the skill bank through five operations: *refine*, *materialize*, *merge*, *split*, and *retire*. Segments labeled as *new skill* are first evaluated for consistency and reuse potential rather than added immediately; promising ones enter a new-skill buffer, where similar candidates are grouped until sufficient evidence supports materializing a new bank entry. For existing skills, *refine* updates contracts when new verified evidence meaningfully changes their reliable effects, *merge* removes redundant near-duplicate skills with highly overlapping contracts, and *split* marks overly broad or inconsistent skills for later re-segmentation. We also *retire* skills that are no longer used or supported by recent trajectories, keeping the bank compact and relevant. The skill bank agent may approve or reject proposed mutations before they are committed to the bank.

4.3 Co-Evolution Framework

We adopt a *co-evolution* framework in COS-PLAY where the decision agent and skill bank agent are mutually dependent. The decision agent uses the current skill bank to interact with the environment and collect trajectories; the skill bank agent then segments these into reusable skills, infers contracts, and refines the bank. This closed loop is self-reinforcing: better skills improve decision making, and better rollouts improve skill learning. Both agents are updated via GRPO [38] using separate LoRA adapters [9]. The decision agent has two adapters: *action-taking*, which rewards progress on the active skill while discouraging unnecessary retrieval, and *skill-retrieval*, which is evaluated at episode end and rewards skills that led to useful, contract-satisfying outcomes. This decomposition lets the agent jointly improve *what* skill to retrieve and choose and *how* to execute it. The skill bank agent uses three adapters. The *segmentation* adapter maps high-value trajectory segments to existing skills while allowing new skills to appear. The *contract* adapter learns compact state-transition summaries that generalize across episodes. The *curator* adapter updates the bank through retention, exploration, and evidence-based merging, splitting, or rejection.

4.4 Reward Design

Although both agents are trained jointly under the co-evolution loop, each LoRA adapter plays a distinct functional role and is therefore optimized with its own reward. All reward terms are computed directly from collected trajectories on CPU and require no additional LLM inference, keeping the training loop lightweight and reproducible. We describe the signal used by each of the five adapters below.

Action-Taking. The action-taking adapter is trained with a per-step reward

$$r_t = r_t^{\text{env}} + \lambda_{\text{f}}\, r_t^{\text{follow}} + r_t^{\text{cost}}, \tag{6}$$

where r_t^{env} is the environment reward, r_t^{follow} is a skill-following shaping term that densifies sparse feedback by giving small bonuses for satisfying predicates in the active skill's effect contract, and r_t^{cost} penalizes skill switching, a common failure mode in early training. We set $\lambda_f = 0.1$ in all experiments.

Skill Retrieval. The skill-retrieval adapter receives a delayed reward evaluated at skill-switch time, computed from the normalized environment reward accumulated during the skill, a temporal-efficiency term, a contract-completion term, an abort penalty for violated preconditions, and a low-weight retrieval-confidence prior from the RAG module. This favors skills that are successful, efficient, and applicable in the current state.

Skill Discovery Pipeline — Diplomacy (Austria), one episode

(a) Raw Trajectory

20-step rollout. Shaded rows mark detected transitions.

t	State Predicates	Action	Intention	r_t
0	`3SC, nbr_intent={???}`	A BUD→TRI	"scout Italian border"	0
1	`italy_in_VEN (neutral)`	F TRI→ALB	"secure Adriatic"	0
4	`russia_in_GAL (hostile)`	A VIE→GAL	"block Russia"	0
5	`galicia_held, serbia_open`	A BUD→SER	"take Serbia"	+1
6	`4SC, turkey_passive`	F ALB→GRE	"push south"	+1
9	`5SC, italy_allied`	F GRE→ION	"enter Mediterranean"	+1
10	`turkey_attacks_BUL`	A BUL S A SER	"hold Bulgaria"	0
11	`5SC, east_front_stable`	A SER S A BUL	"consolidate east"	0

(b) Boundary Proposal

Three candidates scored; t=3 falls below threshold.

Candidate	Score	Signals
t=5	0.83	intention flip ("scout" → "take") + predicate flip (`+serbia_open`)
t=10	0.76	reward drop (+1 → 0) + predicate flip (`+turkey_attacks_BUL`)
t=3	0.41	~~minor intention reword~~ ← discarded

(c) Infer Segmentation

Four segments; three match existing bank skills.

Segment	Skill Label	Conf.
[0–4]	Explore (matched)	0.87
[5–9]	Setup (matched)	0.82
[10–16]	Defend (matched)	0.79
[17–19]	*new skill* (none)	—

(d) Contract Learning (Explore)

Consensus effects over 28 episodes; one sporadic effect dropped.

Add: `+border_info(IT)`, `+border_info(RU)`, `+gal_status_known`
Del: `-unk_intent(IT)`, `-unk_intent(RU)`
Verify: 26/28 pass → rate = 0.93 ✓
`+fleet_adriatic` (5/28) dropped.

(e) Skill Bank Maintenance

Bank mutations triggered by this episode.

Refine Explore: +1 clause `gal_status`
Mat. [17–19] buffered; 12 instances → Negotiate
Merge Press._Ser. ≈ Balk._Push → Setup
Retire Scatter_Units: 0 uses → removed

Stored Skill Entry: Explore (v2, 28 inst.)

The protocol the decision agent queries.

Summary : Scout borders, probe neighbor intentions
Pre-cond : Game start or awareness lost
Plan : Observe → Assess → Commit
Success : ≥3 neighbor intents classified
Abort : Attack on home centers
Contract : `+border_info(`N`)`, `−unk_intent(`N`)`

Figure 2. Skill bank agent pipeline on one Diplomacy episode (Austria). (a) Raw Trajectory. A decision-agent rollout; shaded rows mark skill transitions. **(b) Boundary Proposal.** We score each timestep for transition signals and discard low scorers. **(c) Infer Segmentation.** We select true boundaries and label each segment with a bank skill or *new skill*. **(d) Contract Learning.** We aggregate state deltas across all instances of Explore and drop sporadic effects to obtain a verified contract. **(e) Skill Bank Maintenance.** Four operations (refine, materialize, merge, retire) update the bank. The bottom-right box shows the stored protocol the decision agent queries.

Segmentation. The segmentation adapter is trained to assign high-reward trajectory segments to reusable skills in the current bank. Its reward combines (i) the fraction of positive episode reward covered by existing skills, (ii) a value-matching term between skill quality and segment importance, (iii) a Viterbi decode score for global consistency, and (iv) a decode-margin term for confident assignments. Assignments to *new skill* receive partial credit to prevent systematic overuse of existing skills when novel behavior appears.

Contract Learning. The contract adapter learns compact effect contracts. Without held-out instances, the reward uses F1 between predicted effects and observed start/end predicate changes, with additional terms for predicate coverage, sparsity, and specificity. With held-out instances, it emphasizes generalization through a reward-weighted holdout pass rate, augmented by precision/recall against the consensus effects.

Skill Bank Curator. The curator adapter evaluates proposed bank updates, including *materialize*, *merge*, *split*, and *retire* operations. Its reward balances (i) a quality-alignment term from the continuous skill-quality score, (ii) an exploration term for approving promising new skills with adequate evidence, which prevents systematic rejection of new behaviors, and (iii) a reason-quality term that favors evidence-based decisions citing pass rates, quality scores, and instance counts.

5 Experiments

Evaluations. We evaluate on six game environments spanning single-player puzzle and interactive control tasks, including 2048, Candy Crush, Tetris, and Super Mario Bros. [11, 34], as well as multi-player social reasoning games, Avalon and Diplomacy [57]. Avalon is a team-based hidden-role game in which only one side can win. It is especially challenging for the Good side, which must infer hidden roles from sparse signals such as proposals, votes, and quest outcomes, while Evil players begin fully coordinated and can strategically conceal or sabotage [23]. Diplomacy further increases the difficulty by requiring long-context reasoning, negotiation, alliance tracking, and multi-phase planning. Smaller LLMs have recently been shown to struggle with both Avalon and Diplomacy due to the games' highly social and interactive nature, making them an excellent testbed for COS-PLAY [5, 37].

Across all environments, observations are converted into structured textual states, and the agent interacts through discrete text actions under a unified Gym-style API [3]. Full environment details, including observations, actions, horizons, and rewards, are provided in Appendix B.

Cold Start Initialization. We use GPT-5.4 as a teacher model to generate 60 seed trajectories per game. We then apply supervised fine-tuning (SFT) on these trajectories to train a Qwen3-8B [53] model, which serves as the shared initialization for both the decision agent and the skill bank agent.

Co-Evolving Training Setting. The co-evolving training proceeds in three stages: (1) the decision agent interacts with the environment to collect rollouts; (2) the skill bank agent segments skill candidates from these rollouts and updates the skill bank; and (3) GRPO updates both agents. Starting from the cold-start model, we attach separate LoRA adapters [9] for each trainable function: two for the decision agent (skill retrieval and action taking) and three for the skill bank agent (infer segmentation, contract learning, and skill bank maintenance). We discuss this multi-adapter design further in Appendix E. For social reasoning games (Avalon and Diplomacy), we use GPT-5-mini [31] as opponents to provide strong supervision signals during co-evolution training. Full training details and training curves are in Appendix C and G.

Reward Design and Metrics. For single-player games, we report the native game rewards provided by the benchmark environments [11, 34]. For social games, we use team win rate for Avalon and the number of occupied supply centers for Diplomacy as the main evaluation metrics, following [57], to provide a consistent basis for comparison across baselines. The training rewards used by each LoRA adapter are defined in Section 4.4.

5.1 Main Results

We compare COS-PLAY with strong frontier LLMs, including GPT-5.4 [32], Gemini-3.1-Pro [6], Claude-4.6-Sonnet [1], and gpt-oss-120b [30]. Results are shown in Table 1. Qualitative results appear in Appendix H.

COS-PLAY matches or exceeds frontier LLMs with efficient, few-shot adaptation. Overall, COS-PLAY achieves a substantial average improvement of 25.1% over GPT-5.4 on single-player games over 16 runs. More importantly, COS-PLAY requires only few-shot adaptation to transfer to new benchmarks: each game needs at most 25 iterations to reach strong performance, compared with the hundreds of training steps often required by prior RL-based game agents [39], suggesting the strong data efficiency, learnability, and cross-domain adaptability of our framework.

A small model with structured state tracking and reusable skills can approach frontier LLMs on social reasoning tasks that require long-context memory and multi-turn consistency. For multi-player social games, we use GPT-5.4 as the opponent model, against which most non-GPT baselines underperform (Table 1). In Avalon, COS-PLAY is comparable to Gemini-3.1-Pro and gpt-oss-120b, trailing by only 1% in win rate, while in Diplomacy it outperforms Gemini-3.1-Pro by 8.8%. Role-wise and power-wise breakdowns are provided in Table 4 and Table 5 in Appendix D. These social games expose a core weakness of baseline LLM agents: they often fail to maintain consistent beliefs, track intent across rounds, and act coherently on prior evidence, and tend to fall back to defensive, non-cooperative behavior when facing strong opponents that exploit ambiguity. COS-PLAY alleviates this in Avalon by converting raw histories into structured states and applying phase-specific skills, enabling more consistent suspicion modeling and multi-round commitment despite the asymmetric information available to the Good side. In Diplomacy, where long-context reasoning, negotiation, alliance tracking, and multi-phase planning are all essential, COS-PLAY benefits from iterative skill-bank refinement that yields reusable strategic patterns and improves few-shot adaptation over repeated interactions. Together, these results suggest that structured state tracking and reusable skills provide an effective inductive bias for long-horizon social reasoning, even with a much smaller backbone model.

Model	Single-Player					Multi-Player	
	2048 Reward ↑	Tetris Reward ↑	CandyCrush Reward ↑	Super Mario Bros Reward ↑	Avg. Reward ↑	Avalon Win Rate ↑	Diplomacy Mean SC ↑
GPT-5.4	**1126.6 ± 150.2**	**458.2 ± 203.5**	**532.6 ± 24.8**	752.0 ± 35.7	717.4	**65.0 ± 14.2**	**4.70 ± 0.35**
Gemini-3.1-Pro	813.3 ± 143.6	372.7 ± 157.7	334.3 ± 59.4	436.8 ± 86.1	489.3	42.0 ± 13.2	2.72 ± 0.26
Claude-4.6-Sonnet	945.0 ± 134.5	444.2 ± 182.6	328.6 ± 23.8	399.5 ± 53.4	529.3	40.0 ± 13.1	3.16 ± 0.19
GPT-OSS-120B	1029.5 ± 122.0	358.1 ± 139.7	334.4 ± 40.5	**968.5 ± 175.0**	672.6	40.0 ± 13.1	2.46 ± 0.25
Qwen3-8B	131.0 ± 102.6	32.0 ± 8.5	519.9 ± 37.8	835.5 ± 161.6	379.6	30.0 ± 9.9	2.64 ± 0.18
SFT w/o Skill	516.7 ± 172.3	28.2 ± 9.7	356.1 ± 30.1	736.8 ± 130.4	409.5	28.7 ± 9.7	2.75 ± 0.13
SFT + 1st Skill	385.5 ± 239.7	35.8 ± 7.0	569.6 ± 29.5	871.9 ± 126.2	465.7	21.2 ± 8.9	2.89 ± 0.20
SFT + Final Skill	64.8 ± 46.0	24.4 ± 7.0	554.4 ± 24.3	794.4 ± 112.9	359.5	25.0 ± 9.3	2.65 ± 0.25
GRPO w/o Skill	510.0 ± 249.5	96.7 ± 30.3	163.3 ± 71.4	669.4 ± 130.1	359.9	36.2 ± 10.3	2.76 ± 0.19
GRPO + 1st Skill	152.0 ± 107.9	93.7 ± 37.7	353.8 ± 20.2	621.2 ± 130.2	305.2	31.2 ± 10.0	2.56 ± 0.22
COS-PLAY (Base Qwen3-8B)	**1589.0 ± 192.4**	**510.9 ± 199.5**	**648.8 ± 38.8**	**948.9 ± 153.2**	**924.4**	**39.0 ± 9.4**	**2.96 ± 0.20**

Table 1. Performance across game categories. We report **reward** for 2048, Tetris, Candy Crush, and Super Mario Bros, **overall win rate** for Avalon, and **overall mean supply centers** for Diplomacy. All results are reported with **95% confidence intervals**, based on **16 evaluation rollouts** for single-player games and **10 rollouts per player** for multi-player games. **Conclusion:** COS-PLAY achieves a significant **25.1%** average improvement over GPT-5.4 on single-player games, while remaining competitive with SOTA LLMs on multi-player social reasoning tasks.

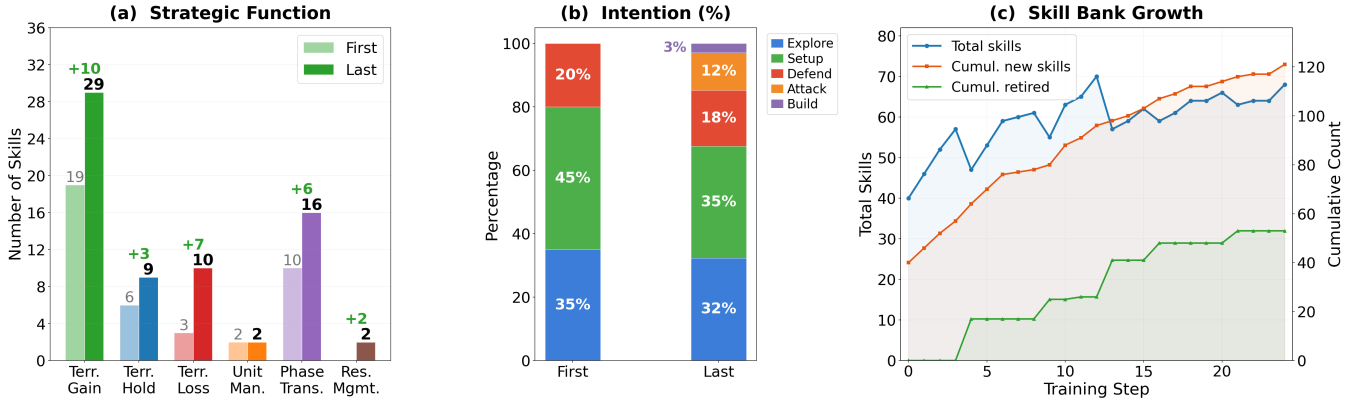

Figure 3. Skill bank evolution over Diplomacy training. (a) Development of Strategic Function Categories from the first to the last training step. Compared with the initial skill bank, the final bank shows notable increases in phase transition and territory loss skills, indicating a broader tactical repertoire. **(b) Changes in Intention Composition** between the first and last training steps, suggesting increasingly goal-directed behavior and greater diversity across skill categories. **(c) Skill Bank Growth** during training: the active bank stays at roughly 55–70 skills, while 121 are discovered overall. Periodic curation removes 53 redundant skills via merge and split, keeping the bank compact and effective.

5.2 Ablation Study

No single component is sufficient. We compare our full pipeline against several variants in Table 1: the base Qwen3-8B [53]; the SFT decision agent without a skill bank (SFT w/o Skill); the SFT model with the first skill bank from the initial co-evolution iteration (SFT + 1st Skill, no co-evolution); the SFT model with the final skill bank (SFT + Final Skill, no co-evolution); the GRPO decision agent without a skill bank (GRPO w/o Skill); and the GRPO decision agent with the first skill bank (GRPO + 1st Skill). While some variants improve specific games, their gains are inconsistent across domains. SFT w/o Skill improves action formatting but lacks reusable long-horizon structure, whereas GRPO w/o Skill improves behavior yet remains unstable under sparse rewards. Variants with mismatched skill banks perform worse, as the policy and skill bank are optimized for different state distributions, making retrieved skills less aligned and often harmful. These findings suggest that the main advantage comes not from skills or RL alone, but from co-evolution, which keeps the skill bank aligned with the policy's changing behavior and task distribution.

Splitting LoRAs by function is essential. A natural alternative to our five stage-specific LoRA adapters is to merge

Game	#Skills	#Cats	Avg. Clauses	Tot. Sub-Ep	Max Inst	Avg. Inst	Gini	Avg. Ver
2048	13	6	0.92	593	236	45.6	0.718	3.1
Tetris	6	2	4.67	295	190	49.2	0.634	4.2
CandyCrush	6	4	13.67	203	78	33.8	0.498	2.0
Super Mario Bros	20	9	7.55	281	74	14.1	0.543	2.5
Avalon	16	3	4.40	479	125	29.9	0.564	6.4
Diplomacy	64	5	6.52	814	45	12.7	0.524	2.4

Table 2. Cross-run skill reusability across games. We report the number of discovered skills (#Skills), number of categories (#Cats), average number of clauses per skill (Avg. Clauses), total number of segmented sub-episodes (Tot. Sub-Ep), the maximum number of instances for the most reused skill (Max Inst), the average number of instances per skill (Avg. Inst), reuse concentration (Gini), and the average number of contract refinement versions per skill (Avg. Ver). These results suggest that our framework learns reusable and adaptable skill abstractions rather than memorizing isolated trajectories.

them into a single adapter per agent. As shown in Table 7 (Appendix E), this merged variant performs worse on Candy Crush than even the base Qwen3-8B, while the stage-specific variant outperforms all baselines by a clear margin. Compressing diverse objectives, including skill retrieval, action generation, segmentation, contract learning, and curation, into a single adapter forces the learned capacity to be shared across incompatible functions and induces interference. Stage-specific LoRAs instead enable cleaner specialization, reduce optimization conflict, and better preserve functional capabilities throughout co-evolution.

5.3 Skill Reusability Analysis

Apart from decision performance, we also evaluate the *reusability* of the learned skills. Unlike prior approaches that manually specify skills or distill them from strong LLMs with human supervision, our skill bank agent automatically extracts skills from unlabeled rollout trajectories. A useful skill should both improve downstream rewards and remain reusable across episodes, since repeated reuse suggests stable preconditions, effects, and execution protocols rather than memorization of isolated trajectories. As summarized in Table 2 and illustrated in Figure 3, our skill bank learns compact and adaptable skill abstractions. In the Diplomacy case study, the bank grows in strategic diversity over training while remaining compact through periodic merge-and-split curation. The x-axis in Figure 3 (left) corresponds to co-evolution iterations, where each point reflects the skill bank after one round of rollout collection and bank update.

5.4 Generalization to General Reasoning Tasks

A natural concern when adapting an LLM to a narrow domain is whether gains in that domain come at the cost of the base model's general reasoning ability. To check this, we evaluate COS-PLAY on two standard LLM reasoning benchmarks: Math-500 [8, 24], which targets mathematical reasoning, and MMLU-Pro [44], which covers broad knowledge and reasoning across diverse domains. As shown in Table 3, COS-PLAY matches Qwen3-8B closely on both benchmarks, with absolute drops of only 0.8% on MMLU-Pro and 1.8% on Math-500. Combined with the substantial gains on game-playing benchmarks in Table 1, this shows that COS-PLAY reaches strong game performance within only 25 co-evolution steps without sacrificing the base model's general reasoning capabilities.

Model	MMLU-Pro Acc. ↑	Math-500 EM ↑
Qwen3-8B	61.99%	46.40%
COS-PLAY	61.15%	44.60%

Table 3. Performance on general reasoning benchmarks. COS-PLAY stays comparable to the base model on both benchmarks, with drops of only 0.8% on MMLU-Pro and 1.8% on Math-500.

5.5 Discussion

Why Reusable Skills Matter in Long-Horizon Tasks. Table 2 suggests that COS-PLAY learns reusable skill abstractions rather than memorizing isolated trajectories. This is especially important in long-horizon and socially strategic settings, where coherent multi-step behavior is required. By retrieving only a small set of relevant skills, COS-PLAY provides compact guidance that improves action selection with low overhead, while continual refinement supports stable co-evolution and efficient adaptation.

Co-evolution Matters More than Any Single Component. Ablations in Table 1 show that the main advantage of COS-PLAY comes not from skill retrieval alone or RL fine-tuning alone, but from jointly updating the decision policy and skill bank. As the policy improves, it encounters new states, behaviors, and failure modes; if the skill bank stays fixed, retrieved skills become mismatched and may even hurt performance. By updating the bank from fresh rollouts, our framework keeps skills relevant and allows them to be refined, split, merged, or retired over time, while the updated

bank in turn provides more reliable guidance and denser feedback for policy learning. This closed loop makes optimization more stable and data efficient, showing that the main benefit comes from co-evolving skills and policy rather than improving either alone.

6 Conclusion

We present COS-PLAY, a co-evolution framework in which an LLM decision agent retrieves skills from a skill bank to guide action taking, while an agent-managed skill pipeline discovers reusable skills from unlabeled rollouts for future decisions. Our framework improves both sides jointly: the decision agent learns better skill retrieval and action generation, while the skill bank agent continually extracts, refines, and updates skills and their contracts. By keeping the policy and skill bank aligned over time, COS-PLAY improves long-horizon control and adaptation. Experiments across diverse game environments show that COS-PLAY yields strong gains over frontier LLM baselines on single-player games while remaining competitive on multi-player social reasoning tasks.

Limitation and Future Work. A key limitation of COS-PLAY is its reliance on compact text-based state summaries, which limits grounded multi-modal understanding and can miss critical evidence in raw visual observations or long temporal dependencies. Over long trajectories, summarization errors can accumulate, reducing skill relevance. An important direction for future work is to extend COS-PLAY to multi-modal interactive environments, where learned skills can capture both interaction dynamics and visual reasoning. We also aim to improve cross-domain transfer, so that learned skills generalize across multi-modal games, agentic environments, and broader visual reasoning tasks.

References

[1] Anthropic. 2026. Introducing Claude Sonnet 4.6.

[2] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. 2019. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680* (2019).

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[4] Peng Chen, Pi Bu, Jun Song, Yuan Gao, and Bo Zheng. 2024. Can vlms play action role-playing games? take black myth wukong as a study case. *arXiv preprint arXiv:2409.12889* (2024).

[5] Alexander Duffy, Samuel J Paech, Ishana Shastri, Elizabeth Karpinski, Baptiste Alloui-Cros, Tyler Marques, and Matthew Lyle Olson. 2026. Democratizing diplomacy: A harness for evaluating any large language model on full-press diplomacy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 40. 37350–37359.

[6] Google DeepMind. 2026. Gemini 3.1 Pro Model Card.

[7] Yicheng He, Chengsong Huang, Zongxia Li, Jiaxin Huang, and Yonghui Yang. 2025. Visplay: Self-evolving vision-language models from images. *arXiv preprint arXiv:2511.15661* (2025).

[8] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring Mathematical Problem Solving With the MATH Dataset. *arXiv preprint arXiv:2103.03874* (2021).

[9] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *Iclr* 1, 2 (2022), 3.

[10] Hengyuan Hu and Dorsa Sadigh. 2023. Language instructed reinforcement learning for human-ai coordination. In *International Conference on Machine Learning*. PMLR, 13584–13598.

[11] Lanxiang Hu, Mingjia Huo, Yuxuan Zhang, Haoyang Yu, Eric P Xing, Ion Stoica, Tajana Rosing, Haojian Jin, and Hao Zhang. 2025. lmgame-Bench: How Good are LLMs at Playing Games? *arXiv preprint arXiv:2505.15146* (2025).

[12] Sihao Hu, Tiansheng Huang, Gaowen Liu, Ramana Rao Kompella, Fatih Ilhan, Selim Furkan Tekin, Yichang Xu, Zachary Yahn, and Ling Liu. 2024. A survey on large language model-based game agents. *arXiv preprint arXiv:2404.02039* (2024).

[13] Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. 2025. R-zero: Self-evolving reasoning llm from zero data. *arXiv preprint arXiv:2508.05004* (2025).

[14] Xu Huang, Junwu Chen, Yuxing Fei, Zhuohan Li, Philippe Schwaller, and Gerbrand Ceder. 2025. Cascade: Cumulative agentic skill creation through autonomous development and evolution. *arXiv preprint arXiv:2512.23880* (2025).

[15] Guanyu Jiang, Zhaochen Su, Xiaoye Qu, et al. 2026. XSkill: Continual Learning from Experience and Skills in Multimodal Agents. *arXiv preprint arXiv:2603.12056* (2026).

[16] Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. 2026. SoK: Agentic Skills–Beyond Tool Use in LLM Agents. *arXiv preprint arXiv:2602.20867* (2026).

[17] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. 2010. Constructing skill trees for reinforcement learning agents from demonstration trajectories. *Advances in neural information processing systems* 23 (2010).

[18] Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, Yubo Li, Bingran You, Haotian Shen, Jiankai Sun, et al. 2026. SkillsBench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670* (2026).

[19] Zongxia Li, Hongyang Du, Chengsong Huang, Xiyang Wu, Lantao Yu, Yicheng He, Jing Xie, Xiaomin Wu, Zhichao Liu, Jiarui Zhang, et al. 2026. Mm-zero: Self-evolving multi-model vision language models from zero data. *arXiv preprint arXiv:2603.09206* (2026).

[20] Zaijing Li, Yuquan Xie, Rui Shao, Gongwei Chen, Dongmei Jiang, and Liqiang Nie. 2024. Optimus-1: Hybrid multimodal memory empowered agents excel in long-horizon tasks. *Advances in neural information processing systems* 37 (2024), 49881–49913.

[21] Zongxia Li, Wenhao Yu, Chengsong Huang, Rui Liu, Zhenwen Liang, Fuxiao Liu, Jingxi Che, Dian Yu, Jordan Boyd-Graber, Haitao Mi, et al. 2025. Self-rewarding vision-language model via reasoning decomposition. *arXiv preprint arXiv:2508.19652* (2025).

[22] Yi Liao, Yu Gu, Yuan Sui, Zining Zhu, Yifan Lu, Guohua Tang, Zhongqian Sun, and Wei Yang. 2025. Think in games: Learning to reason in games via reinforcement learning with large language models. *arXiv preprint arXiv:2508.21365* (2025).

[23] Jonathan Light, Min Cai, Sheng Shen, and Ziniu Hu. 2023. Avalonbench: Evaluating llms playing the game of avalon. *arXiv preprint arXiv:2310.05036* (2023).

[24] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's Verify Step by Step. *arXiv preprint arXiv:2305.20050* (2023).

[25] Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Junge Zhang, and Jianbin Jiao. 2025. Agentic reinforcement learning with implicit step rewards. *arXiv preprint arXiv:2509.19199* (2025).

[26] Wenxuan Lu, Jiangyang He, Zhanqiu Zhang, Steven Y Guo, and Tianning Zang. 2025. Cultivating Gaming Sense for Yourself: Making VLMs Gaming Experts. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 13132–13152.

[27] Weiyu Ma, Yuqian Fu, Zecheng Zhang, Guohao Li, and Bernard Ghanem. 2025. VLMs Play StarCraft II: A Benchmark and Multimodal Decision Method. *arXiv preprint arXiv:2503.05383* (2025).

[28] Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. 2026. ProcMEM: Learning Reusable Procedural Memory from Experience via Non-Parametric PPO for LLM Agents. *arXiv preprint arXiv:2602.01869* (2026).

[29] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[30] OpenAI. 2025. Introducing gpt-oss.

[31] OpenAI. 2026. GPT-5 mini Model (gpt-5-mini).

[32] OpenAI. 2026. Introducing GPT-5.4.

[33] Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wolczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, et al. 2024. Balrog: Benchmarking agentic llm and vlm reasoning on games. *arXiv preprint arXiv:2411.13543* (2024).

[34] Dongmin Park, Minkyu Kim, Beongjun Choi, Junhyuck Kim, Keon Lee, Jonghyun Lee, Inkyu Park, Byeong-Uk Lee, Jaeyoung Hwang, Jaewoo Ahn, et al. 2025. Orak: A foundational benchmark for training and evaluating llm agents on diverse video games. *arXiv preprint arXiv:2506.03610* (2025).

[35] Siyuan Qi and Song-Chun Zhu. 2018. Intent-aware multi-agent reinforcement learning. In *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 7533–7540.

[36] Maria Abi Raad, Arun Ahuja, Catarina Barros, Frederic Besse, Andrew Bolt, Adrian Bolton, Bethanie Brownfield, Gavin Buttimore, Max Cant, Sarah Chakera, et al. 2024. Scaling instructable agents across many simulated worlds. *arXiv preprint arXiv:2404.10179* (2024).

[37] Shahab Rahimirad, Guven Gergerli, Lucia Romero, Angela Qian, Matthew Lyle Olson, Simon Stepputtis, and Joseph Campbell. 2025. Bayesian Social Deduction with Graph-Informed Language Models. *arXiv preprint arXiv:2506.17788* (2025).

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

[39] Tianye Shu, Jialin Liu, and Georgios N Yannakakis. 2021. Experience-driven PCG via reinforcement learning: A Super Mario Bros study. In *2021 IEEE Conference on Games (CoG)*. IEEE, 1–9.

[40] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815* (2017).

[41] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291* (2023).

[42] Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. 2025. Reinforcement learning for self-improving agent with skill library. *arXiv preprint arXiv:2512.17102* (2025).

[43] Qinsi Wang, Bo Liu, Tianyi Zhou, Jing Shi, Yueqian Lin, Yiran Chen, Hai Helen Li, Kun Wan, and Wentian Zhao. 2025. Vision-zero: Scalable vlm self-improvement via strategic gamified self-play. *arXiv preprint arXiv:2509.25541* (2025).

[44] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems* 37 (2024), 95266–95290.

[45] Zirui Wang, Junyi Zhang, Jiaxin Ge, Long Lian, Letian Fu, Lisa Dunlap, Ken Goldberg, XuDong Wang, Ion Stoica, David M Chan, et al. 2026. VisGym: Diverse, Customizable, Scalable Environments for Multimodal Agents. *arXiv preprint arXiv:2601.16973* (2026).

[46] Xiyang Wu, Rohan Chandra, Tianrui Guan, Amrit Bedi, and Dinesh Manocha. 2023. Intent-aware planning in heterogeneous traffic via distributed multi-agent reinforcement learning. In *Conference on Robot Learning*. PMLR, 446–477.

[47] Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, et al. 2026. SkillRL: Evolving Agents via Recursive Skill-Augmented Reinforcement Learning. *arXiv preprint arXiv:2602.08234* (2026).

[48] Zhishang Xiang, Chengyi Yang, Zerui Chen, Zhimin Wei, Yunbo Tang, Zongpei Teng, Zexi Peng, Zongxia Li, Chengsong Huang, Yicheng He, et al. 2025. A Systematic Survey of Self-Evolving Agents: From Model-Centric to Environment-Driven Co-Evolution. *Researchgate* (2025).

[49] Han Xiao, Guozhi Wang, Hao Wang, Shilong Liu, Yuxiang Chai, Yue Pan, Yufeng Zhou, Xiaoxin Chen, Yafei Wen, and Hongsheng Li. 2026. UI-Mem: Self-Evolving Experience Memory for Online Reinforcement Learning in Mobile GUI Agents. *arXiv preprint arXiv:2602.05832* (2026).

[50] Renjun Xu and Yang Yan. 2026. Agent skills for large language models: Architecture, acquisition, security, and the path forward. *arXiv preprint arXiv:2602.12430* (2026).

[51] Zhongwen Xu, Xianliang Wang, Siyi Li, Tao Yu, Liang Wang, Qiang Fu, and Wei Yang. 2025. Agents play thousands of 3D video games. *arXiv preprint arXiv:2503.13356* (2025).

[52] Zelai Xu, Zhexuan Xu, Xiangmin Yi, Huining Yuan, Xinlei Chen, Yi Wu, Chao Yu, and Yu Wang. 2025. VS-Bench: Evaluating VLMs for Strategic Reasoning and Decision-Making in Multi-Agent Environments. (2025).

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

[54] Shuo Yu, Mingyue Cheng, Daoyu Wang, Qi Liu, Zirui Liu, Ze Guo, and Xiaoyu Tao. 2025. Memweaver: A hierarchical memory from textual interactive behaviors for personalized generation. *arXiv preprint arXiv:2510.07713* (2025).

[55] Simon Yu, Gang Li, Weiyan Shi, and Peng Qi. 2025. Polyskill: Learning generalizable skills through polymorphic abstraction. *arXiv preprint arXiv:2510.15863* (2025).

[56] Renos Zabounidis, Yue Wu, Simon Stepputtis, Woojun Kim, Yuanzhi Li, Tom Mitchell, and Katia Sycara. 2026. SCALAR: Learning and Composing Skills through LLM Guided Symbolic Planning and Deep RL Grounding. *arXiv preprint arXiv:2603.09036* (2026).

[57] Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, et al. 2025. Agentevolver: Towards efficient self-evolving agent system. *arXiv preprint arXiv:2511.10395* (2025).

[58] Alex L Zhang, Thomas L Griffiths, Karthik R Narasimhan, and Ofir Press. 2025. Videogamebench: Can vision-language models complete popular video games? *arXiv preprint arXiv:2505.18134* (2025).

[59] Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. 2026. MemSkill: Learning and Evolving Memory Skills for Self-Evolving Agents. *arXiv preprint arXiv:2602.02474* (2026).

[60] Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, et al. 2026. MemRL: Self-Evolving Agents via Runtime Reinforcement Learning on Episodic Memory. *arXiv preprint arXiv:2601.03192* (2026).

A Reproducibility

We will release the full COS-PLAY implementation, including the decision agent, skill bank agent, co-evolution training loop, GRPO training scripts with multi-adapter LoRA configurations, skill bank pipeline, and all prompt templates used in this paper. We will also provide the environment wrappers, reward definitions, and configuration files needed to reproduce the experiments across all six game environments.

We document the full evaluation protocol in the paper and appendix, including environment settings (Appendix B), hyperparameters, the number of evaluation rollouts per setting (Appendix C), and reward definitions (Section 4.4). For single-player games, we report results over 16 rollouts; for multi-player games, over 10 rollouts per player. Because multi-player experiments use black-box LLM APIs as opponents (GPT-5-mini for training, GPT-5.4 for evaluation), exact numerical replication may vary across API versions and infrastructure conditions. To reduce this variance, we use fixed prompts, deterministic environment seeds where supported, and repeated runs under the same protocol. We therefore expect the main qualitative conclusions to remain reproducible even when exact per-run metrics vary slightly.

B Game Environment Settings

We evaluate on six game environments spanning puzzle solving, platform control, and multi-agent social reasoning. In all cases, observations are converted into textual or natural-language state descriptions, and the agent interacts through discrete text actions. Below we summarize the observation space, action space, horizon, and reward for each environment.

2048. 2048 is a single-player sliding-tile puzzle on a 4×4 grid. Equal-valued tiles merge on contact, and the game ends when no valid moves remain. The agent observes a text-rendered board with tile values, score, maximum tile, empty-cell count, and directional move lookahead. The action space has four actions: `up`, `down`, `left`, and `right`. Episodes are capped at 200 steps, with reward given by the merge score.

Candy Crush. Candy Crush is a single-player match-3 puzzle on an 8×8 board with four candy colors. The player swaps adjacent candies to form matches, after which matched candies are cleared and new candies fall. The agent observes a text-based board and the currently valid swap actions. The action space is dynamic and consists of valid coordinate-pair swaps at each step. Episodes last at most 50 moves, and rewards are based on points from matches and cascades.

Tetris. Tetris is a single-player tile-stacking game on a 10×20 board with the standard seven tetrominoes and a four-piece preview queue. The agent receives an ASCII board, the current piece, upcoming pieces, and summary statistics such as stack height, holes, cleared lines, and level. Instead of primitive controls, the environment provides macro actions corresponding to valid placements defined by rotation and target column. Episodes are capped at 200 steps, and reward is the game score.

Super Mario Bros. Super Mario Bros. is a single-player side-scrolling platform game where Mario must traverse the level, avoid hazards, and reach the flag. Observations are converted into natural-language descriptions of Mario's position, nearby objects and enemies, and relevant game-state information. The action space contains seven discrete actions: `noop`, `right`, `right+A`, `A`, `left`, `left+A`, and `down`. Episodes are capped at 200 steps. Rewards combine progress and in-game signals, including distance, coins, and time bonus.

Avalon. Avalon is a five-player social deduction game with hidden roles: Merlin, two Servants, one Minion, and one Assassin. Play proceeds through quest rounds with team proposal, voting, quest execution, and assassination phases. The agent observes a natural-language state description containing the phase, private role information, quest progress, leader identity, proposed team, discussion history, and vote outcomes. Actions depend on the phase and include team proposals, votes, quest choices, and assassination targets. Each game contains up to five quests, and rewards are based mainly on whether the agent's side wins.

Diplomacy. Diplomacy is a seven-player grand-strategy board game on the standard European map. The powers are Austria, England, France, Germany, Italy, Russia, and Turkey, and the game cycles through movement, retreat, and adjustment phases. The agent observes a natural-language state including the current phase, controlled power, unit locations, supply center counts, valid orders, recent negotiations, and phase history. Actions are standard Diplomacy order strings issued jointly for all controlled units in a phase. Episodes are capped at 20 phases, and rewards are based on the number of controlled supply centers normalized by the victory target of 18.

C Key Hyperparameters

We summarize the main hyperparameters used in co-evolution training for the six game environments in the main paper. Table 6 lists the game-specific settings used in our main experiments. All training runs are conducted on an 8×A100 GPU cluster. For games without explicit GRPO overrides, we report the default values directly: GRPO clip ratio 0.2, maximum 4 epochs, no advantage clipping, learning rate 5×10^{-5}, and KL coefficient 0.05.

D Breakdown of Role-wise Performance on Multi-player Games

For multi-player social games, we use GPT-5.4 as the opponent model. We report the overall results in Table 1 of the main paper. Here we provide finer-grained breakdowns in Table 4 (per-role win rates for Avalon) and Table 5 (per-power mean supply centers for Diplomacy). The discussion

Model	Avalon						Overall
	Good			Evil			
	Merlin Win Rate ↑	Servant Win Rate ↑	Good Avg. Win Rate ↑	Assassin Win Rate ↑	Minion Win Rate ↑	Evil Avg. Win Rate ↑	Avg. Win Rate ↑
GPT-5.4	**62.5 ± 27.9**	**47.4 ± 20.5**	**51.9 ± 17.7**	**100.0 ± 19.5**	**85.7 ± 24.4**	**92.3 ± 15.9**	**65.0 ± 14.2**
Gemini-3.1-Pro	10.0 ± 19.3	36.4 ± 18.7	28.1 ± 14.9	66.7 ± 30.2	66.7 ± 23.6	66.7 ± 20.0	42.0 ± 13.2
Claude-4.6	30.0 ± 24.8	40.9 ± 19.0	37.5 ± 15.9	33.3 ± 30.2	50.0 ± 24.6	44.4 ± 20.9	40.0 ± 13.1
GPT-OSS-120B	30.0 ± 24.8	31.8 ± 18.2	31.2 ± 15.3	50.0 ± 31.2	58.3 ± 24.4	55.6 ± 20.9	40.0 ± 13.1
Qwen3-8B	18.8 ± 18.2	18.4 ± 12.1	18.5 ± 10.2	**66.7 ± 23.6**	42.9 ± 23.0	53.8 ± 17.9	30.0 ± 9.9
SFT w/o Skill	18.8 ± 18.2	18.4 ± 12.1	18.5 ± 10.2	41.7 ± 24.4	57.1 ± 23.0	50.0 ± 17.9	28.7 ± 9.7
SFT + 1st Skill	6.2 ± 13.6	13.2 ± 10.8	11.1 ± 8.5	33.3 ± 23.6	50.0 ± 23.2	42.3 ± 17.8	21.2 ± 8.9
SFT + Final Skill	25.0 ± 19.7	**26.3 ± 13.5**	25.9 ± 11.4	8.3 ± 17.0	35.7 ± 22.4	23.1 ± 15.5	25.0 ± 9.3
GRPO w/o Skill	**37.5 ± 21.4**	15.8 ± 11.5	22.2 ± 10.9	58.3 ± 24.4	**71.4 ± 21.5**	**65.4 ± 17.2**	36.2 ± 10.3
GRPO + 1st Skill	31.2 ± 20.7	18.4 ± 12.1	22.2 ± 10.9	41.7 ± 24.4	57.1 ± 23.0	50.0 ± 17.9	31.2 ± 10.0
COS-PLAY (Base Qwen3-8B)	33.3 ± 17.7	23.3 ± 12.2	**26.9 ± 10.4**	64.3 ± 22.5	63.2 ± 20.0	63.6 ± 15.6	**39.0 ± 9.4**

Table 4. Per-role win rates for Avalon. We report win rates with 95% confidence intervals for both good-side roles (Merlin, Servant) and evil-side roles (Assassin, Minion). Results show that our method reaches comparable performance to Gemini-3.1-Pro and GPT-OSS-120B.

Model	Diplomacy							Overall
	Austria Mean SC ↑	England Mean SC ↑	France Mean SC ↑	Germany Mean SC ↑	Italy Mean SC ↑	Russia Mean SC ↑	Turkey Mean SC ↑	Mean SC ↑
GPT-5.4	**4.38 ± 1.34**	**4.12 ± 0.82**	**4.50 ± 1.09**	**5.12 ± 0.95**	**4.50 ± 0.77**	**5.12 ± 1.52**	**5.12 ± 0.95**	**4.70 ± 0.35**
Gemini-3.1-Pro	2.50 ± 0.89	2.38 ± 0.63	3.12 ± 0.70	2.88 ± 0.82	3.14 ± 0.35	3.14 ± 0.35	1.86 ± 1.24	2.72 ± 0.26
Claude-4.6	3.20 ± 0.56	2.90 ± 0.41	3.50 ± 0.70	3.78 ± 0.64	3.20 ± 0.30	2.80 ± 0.30	2.80 ± 0.66	3.16 ± 0.19
GPT-OSS-120B	1.75 ± 1.07	2.88 ± 0.29	2.88 ± 0.29	2.62 ± 0.44	3.00 ± 0.00	2.88 ± 0.29	1.25 ± 0.97	2.46 ± 0.25
Qwen3-8B	1.62 ± 0.26	2.75 ± 0.25	2.75 ± 0.16	2.88 ± 0.12	3.00 ± 0.00	2.88 ± 0.23	2.62 ± 0.18	2.64 ± 0.18
SFT w/o skill	2.12 ± 0.23	2.75 ± 0.16	2.88 ± 0.12	2.62 ± 0.26	2.88 ± 0.12	3.00 ± 0.00	3.00 ± 0.00	2.75 ± 0.13
SFT + 1st skill	2.12 ± 0.30	2.88 ± 0.12	**3.12 ± 0.30**	2.50 ± 0.33	3.12 ± 0.12	3.25 ± 0.25	**3.25 ± 0.16**	2.89 ± 0.20
SFT + final skill	2.12 ± 0.44	2.75 ± 0.16	2.25 ± 0.25	2.71 ± 0.29	2.88 ± 0.23	2.88 ± 0.12	3.00 ± 0.57	2.65 ± 0.25
GRPO w/o skill	**2.33 ± 0.33**	2.83 ± 0.17	3.00 ± 0.00	2.80 ± 0.20	3.00 ± 0.00	3.20 ± 0.20	2.20 ± 0.37	2.76 ± 0.19
GRPO + 1st skill	1.38 ± 0.18	**3.12 ± 0.12**	2.25 ± 0.16	2.75 ± 0.37	3.00 ± 0.00	3.25 ± 0.16	2.14 ± 0.26	2.56 ± 0.22
COS-PLAY (Base Qwen3-8B)	2.22 ± 0.55	2.70 ± 0.42	3.10 ± 0.46	**3.12 ± 0.58**	**3.20 ± 0.39**	**3.30 ± 0.30**	3.00 ± 0.72	**2.96 ± 0.20**

Table 5. Per-power mean supply centers (SC) for Diplomacy. We report the mean supply centers achieved by each model for each power, together with the overall mean across all powers, with 95% confidence intervals. GPT-5.4 is evaluated in self-play, while all other models are evaluated against GPT-5.4. **Conclusion:** COS-PLAY outperforms Gemini-3.1-Pro by 8.8% on average.

of these results is provided alongside the main social-games analysis in Section 5.

E Necessity of Splitting LoRAs by Function

In our framework, both the decision agent and the skill bank agent use multiple LoRA adapters for different functional stages. Specifically, the decision agent uses two LoRAs for skill retrieval and action generation, while the skill bank agent uses three LoRAs for trajectory segmentation, contract generation, and skill bank maintenance. To better understand the role of each adapter and the effect of splitting LoRAs by function, we conduct a supplementary experiment on Candy Crush in Table 7, comparing COS-PLAY with stage-specific LoRAs against a merged variant that uses only one LoRA for the decision agent and one for the skill bank agent. The corresponding analysis is included in the main-paper ablation study (Section 5.2).

Game	Total Steps	Episodes / Step	Ckpt Int.	GRPO LR	KL Coeff.	Clip	Max Epochs	Adv. Clip
2048	10	8	3	5×10^{-5}	0.05	0.20	4	–
Candy Crush	10	8	3	5×10^{-5}	0.05	0.20	4	–
Tetris	7	8	1	2×10^{-5}	0.08	0.10	2	3.0
Super Mario Bros.	20	8	1	3×10^{-5}	0.04	0.15	3	5.0
Avalon	20	20	1	2×10^{-5}	0.06	0.15	2	3.0
Diplomacy	25	28	1	1×10^{-5}	0.08	0.12	2	3.0

Table 6. Key co-evolution training hyperparameters across the six game environments in the main paper. "Ckpt Int." denotes checkpoint interval. For 2048 and Candy Crush, all GRPO fields use the default values.

Model	Reward ↑
GPT-5.4	532.6 ± 24.8
Qwen3-8B	519.9 ± 37.8
COS-PLAY (Merged LoRAs)	502.5 ± 22.4
COS-PLAY (Stage-Specific LoRAs)	**648.8 ± 38.8**

Table 7. Necessity of splitting LoRAs. On Candy Crush, COS-PLAY with stage-specific LoRAs (one per function) substantially outperforms a merged variant that compresses all functions into a single LoRA per agent, indicating that splitting LoRAs by function is important for handling the diverse objectives arising in co-evolution.

F GRPO Reward Diversity

A practical challenge shared across all five adapters is *reward saturation*: when multiple GRPO samples in a group receive identical rewards, the group-normalized advantages become zero and learning stalls. This occurs frequently when structurally different LLM completions (e.g., contracts with different wording but identical parsed effects) map to the same reward. To address this, each reward function in Section 4.4 incorporates a lightweight content-sensitive tiebreaker—a deterministic hash of the raw completion text—blended at low weight (8–14%) into the final reward. This ensures that textually different completions always produce some reward spread, preserving the GRPO learning signal without distorting the structural reward ranking.

G Training Curve

Figure 4 presents the training reward curves for all six game environments. In the single-player games (2048, Candy Crush, Tetris, and Super Mario), the ego-player reward increases steadily over co-evolution steps, confirming that the joint optimization of the decision agent and skill bank yields progressively stronger play. The multi-player games (Avalon and Diplomacy), trained under self-play, exhibit flatter reward trajectories. This is expected: since all players share the same model, improvements to the ego agent are mirrored by equally stronger opponents, causing the reward to converge near the game-theoretic equilibrium rather than rise monotonically. Notably, the variance bands in Diplomacy remain tight throughout training, suggesting stable learning dynamics despite the complex 7-player negotiation setting. The true policy improvement in these competitive games is revealed not by the self-play reward curve but by evaluation against external opponents, as reported in Table 1.

H Qualitative Analysis

To complement the aggregate results, we present step-level case studies in Candy Crush and Diplomacy that show how the evolved skill bank changes trajectory-level decisions across stages of play (Figures 5–8). We include strategy annotations and brief justifications explaining why COS-PLAY outperforms GPT-5.4 at key steps. We also provide failure analysis for both GPT-5.4 and COS-PLAY in Figure 9, highlighting qualitative differences and key insights. Finally, we examine skill retrieval and its causal role in Diplomacy in Figure 10, showing how retrieved skills shape phase transitions and subsequent actions.

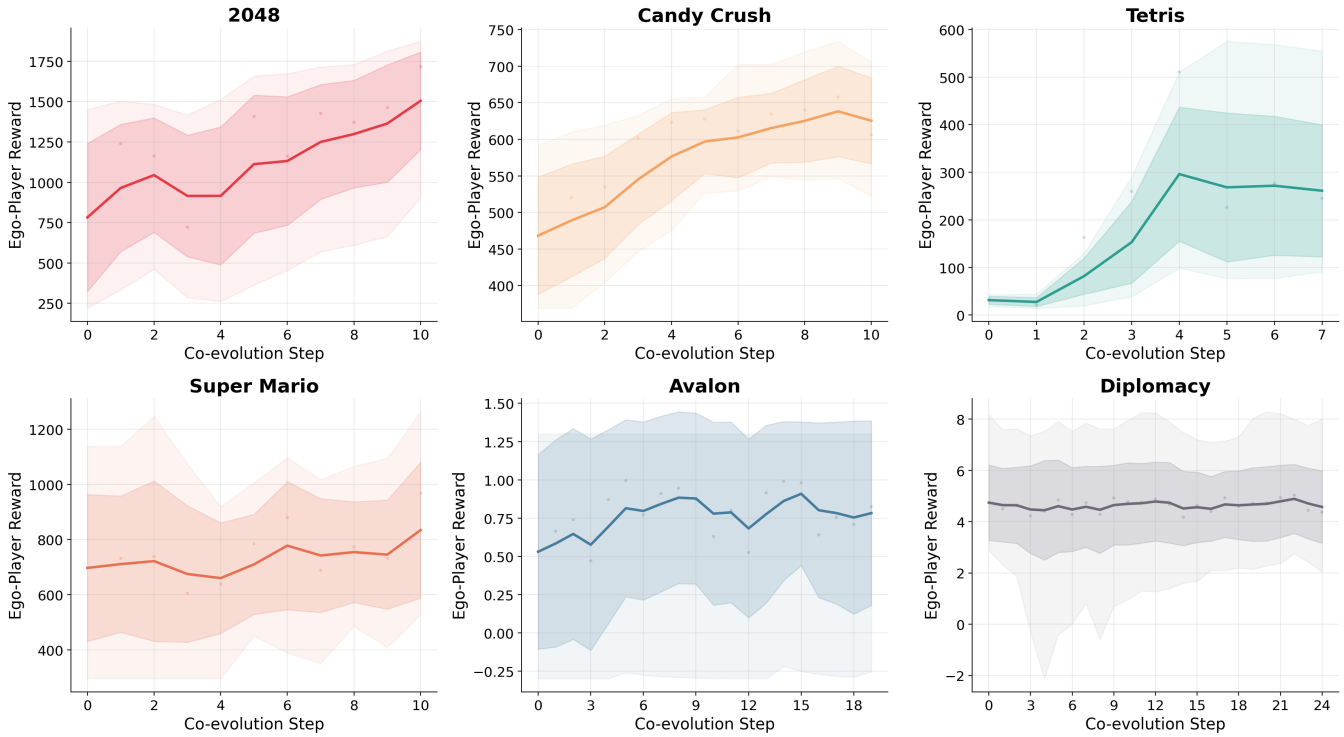

Figure 4. Co-evolution reward curves for all games. Single-player games show steady gains, indicating improved strategies from joint decision-agent and skill-bank training. Multiplayer self-play remains flat because all players improve symmetrically, pushing rewards toward equilibrium.

Candy Crush Strategy Comparison: GPT-5.4 vs. COS-PLAY

Setting

Game: *Candy Crush* **Episodes:** *GPT-5.4 median* vs. *our best* **Final Outcome:** *COS-PLAY: 806* vs. *GPT-5.4: 547* (+47%)

Evolved Skill Bank (2 of 6 skills deployed):

OPTIMIZE	Removes inefficiencies, clears blockers, restructures the board.	*Protocol:* Analyze → improve → verify.	Steps 0–4, 11–16, 23–28, 35–40
CLEAR	Harvests the best available match by exploiting a prepared board state.	*Protocol:* Identify → execute → assess.	Steps 5–10, 17–22, 29–34, 41–49

Key Distinction: *OPTIMIZE* prepares the board; *CLEAR* harvests it. GPT-5.4 uses four overlapping labels (*CLEAR* ≈ *ATTACK* ≈ *EXECUTE*) with no clear phase boundaries.

Aggregate Contrast: COS-PLAY averages **6.6 steps per skill run** with **7.6 switches per game**; GPT-5.4 averages 3.1 steps per run with 14.0 switches. Best single-step reward: 62 (prepared cascade) versus 33 (opportunistic match).

Early Game (Steps 0–10)

GPT-5.4

Step 0 – Opening: *[CLEAR]* Swaps ((5,4),(5,5)) for an immediate central G match.
Reward: +15, cumul.=15
Strategy: Takes the first available central match for 15 points; the board is no better off afterward.

Step 3 – Minimal match: *[CLEAR]* Swaps ((2,4),(2,5)) for a vertical red 3-match.
Reward: +3, cumul.=33
Strategy: A bare 3-match—no cascade, no board restructuring.

Step 5 – Label switch: Already switched *CLEAR*→*ATTACK* by step 4.
Reward: +15, cumul.=58
Strategy: Labels keep changing, but no actual preparation happens between harvests.

COS-PLAY

Step 0 – Opening: *[OPTIMIZE]* State: `note=Bottom row has 4 P's – potential 4-match.`
Targets the flagged P-cluster via #19→((7,0),(7,1)) rather than pursuing any immediate match.
Reward: +17, cumul.=17
Why better: Similar immediate reward (17 vs. 15), but COS-PLAY targets the flagged cluster to set up future cascades; GPT-5.4 just takes the nearest match.

Step 3 – Setup payoff: *[OPTIMIZE]* 4th consecutive step. State: `note=Swapped G/R to create 2 new pairs.`
Prior restructuring from steps 0–2 pays off: #18→((5,3),(5,4)) triggers a massive cascade.
Reward: +34, cumul.=71 (vs. +3, cumul.=33)
Why better: 10× GPT-5.4's reward at this step: three OPTIMIZE steps produce a 34-point cascade versus a 3-point match. Cumulative score is 2.2× ahead.

Step 5 – OPTIMIZE→CLEAR: First deliberate skill switch after 5 steps of board preparation. State: `note=Swapped P/C to create 2 new pairs.`
Reward: +13, cumul.=91 (vs. 58)
Why better: CLEAR inherits a board that OPTIMIZE already restructured. GPT-5.4 switched CLEAR→ATTACK by step 4 with no preparation in between. Lead: +57%.

Figure 5. Step-level comparison between GPT-5.4 and our method in Candy Crush (1/2: Setting & Early Game). Continued in Figure 6.

Candy Crush Strategy Comparison (cont'd)

Mid Game (Steps 11–34)

GPT-5.4

Step 23 – Reactive midgame: Still in *[CLEAR]*, following the generic "immediate match and cascade setup" approach.
Reward: +6, cumul.=277
Strategy: Still in reactive mode since step 22; no restructuring to set up step 28's cascade.

Step 28 – Missed jackpot: *[CLEAR]* "Complete the bottom-right candy match" ((6,6),(6,7)).
Reward: +3, cumul.=327
Strategy: Picks a corner match for 3 points; a higher-value cascade is available elsewhere on the board.

COS-PLAY

Step 23 – OPTIMIZE cycle: First step of the mid-game OPTIMIZE phase. State: `note=Swapped G/P to clear threat, +11.`
The previous CLEAR phase earned a steady 10–31 per step; the agent now switches back to preparation, setting up step 28's cascade.
Reward: +25, cumul.=355 (vs. 277)
Why better: Back in preparation while GPT-5.4 stays in CLEAR. Lead: +28%.

Step 28 – Cascade jackpot: *[OPTIMIZE]* 6th step of this phase. State: `note=Three new pairs created, threat of chain reaction.`
The previous five OPTIMIZE steps yielded modest 7–24 rewards while patiently restructuring the board. Now #9→((3,5),(4,5)) triggers the episode's largest payoff.
Reward: **+62**, cumul.=455 (vs. +3, cumul.=327)
Why better: Five restructuring steps turn the flagged "chain reaction" into a 62-point cascade—20× GPT-5.4's reward at this step. Gap: +39%.

Late Game (Steps 35–49)

Step 40 – Endgame panic: Switches to *[EXECUTE]*: "cut pairs fast with 10 moves left."
Reward: +21, cumul.=455
Strategy: Switches to EXECUTE under time pressure. Score: 434 vs. COS-PLAY 573.

Step 46 – Broken intent: *[EXECUTE]* "maximize endgame points."
Reward: +3, cumul.=509
Strategy: States "maximize endgame points" but scores only +3.

Step 49 – Final move: *[EXECUTE]* "horizontal four-match."
Reward: +10, **final score=547**

Step 40 – Endgame OPTIMIZE: The skill description adapts to "maximize remaining moves" (compared to the mid-game's "maximize cascades"). State: `note=Created 3 new pairs.`
Reward: +24, cumul.=637 (vs. 455)
Why better: Score: 573 vs. 434 (+32%) with the same 10 moves left. OPTIMIZE keeps preparing; GPT-5.4 switches to EXECUTE. Lead: +40%.

Step 46 – CLEAR + URGENCY: The system injects an urgency signal: "very few moves left—maximise every action." Preceding steps scored +26, +21, +26.
Reward: +30, cumul.=760 (vs. +3, cumul.=509)
Why better: With the URGENCY signal, the agent sustains 21–30 points per step; GPT-5.4 scores +3 despite its own "maximize endgame" label.

Step 49 – Final move: *[CLEAR]* + URGENCY. #9→((4,4),(5,4)).
Reward: +22, **final score=806** (vs. +10, 547). **+47% improvement.**

Figure 6. Step-level comparison between GPT-5.4 and our method in Candy Crush (2/2: Mid & Late Game). See Figure 5 for setting and early game.

Diplomacy Strategy Comparison: GPT-5.4 vs. COS-PLAY

Setting

Game: *Diplomacy (Austria)* **Episodes:** *COS-PLAY best* vs. *GPT-5.4 best & typical* **Final Outcome:** *COS-PLAY: 7 SC* vs. *GPT-5.4: 7 SC (best), 4 SC (typical)*

Center Trajectories:

COS-PLAY	3→3→3→3→**4**→4→4→4→4→**5**→5→5→**6**→6→6→**7**→7→7→7→7
GPT-5.4 (best)	3→3→**5**→5→5→5→5→5→5→5→5→5→5→5→**7**→7→7→7→7
GPT-5.4 (typ.)	3→3→3→**4**→4→4→**3**→3→3→3→3→4→4→4→4→**4**→4→4→4→4

Evolved Skill Bank (4 skills deployed):

EXPLORE	Scout borders and probe neighbor intentions.	*Protocol:* Observe → assess → commit.	Steps 0–4
SETUP	Secure supply centers, prepare for expansion.	*Protocol:* Reposition → coordinate → strike.	Steps 5–10, 17–19
DEFEND	Protect holdings after territorial growth.	*Protocol:* Consolidate → support → hold.	Steps 11–16
ATTACK	Aggressive pivot when SETUP yields no growth.	*Protocol:* Target → commit → exploit.	Steps 11–14 (3/28 ep.)

Key Distinction: EXPLORE → SETUP → DEFEND forms a strict temporal pipeline. GPT-5.4 uses five overlapping labels (*Pressure Serbia* ≈ *Secure Balkan* ≈ *Defensive Line*) with no clear phase boundaries.

Aggregate Contrast: COS-PLAY averages **3 phase transitions** per episode with **2–3 switches**; GPT-5.4 uses 8+ intention tags with a switching rate of ∼0.43. The stability floor for COS-PLAY is min=3 SC (it never loses starting centers), compared to GPT-5.4's min=1 SC (near-elimination in 27% of episodes).

Early Game (Steps 0–4)

GPT-5.4

Step 0 – S1901M, Opening: *[POSITION]* Textbook Balkan opening: A BUD–SER, A VIE–GAL, F TRI–ALB.
Effect: Immediate Serbia grab + Galicia contest + Albania staging.
Strategy: Commits all three units to the Balkans on turn 1—fast but inflexible, with no reconnaissance.

Steps 2–3 – S1902M, 5 SC: Already at **5 centers and 5 units** by step 2.
Orders: A BUD–RUM, A SER S A GRE–BUL, F GRE–BUL/SC.
Effect: All-in Balkan push—aggressive three-army surge.
Strategy: All-in on one theater; if opponents push elsewhere, there is nothing left to respond with.

Summary: Reaches 3 → 5 by step 2, but the lack of flexibility sets up the 13-phase plateau that follows.

COS-PLAY

Step 0 – S1901M, Opening: *[EXPLORE]* Scout borders. Orders: A BUD–TRI (not the standard BUD→SER grab).
Effect: The army moves to Trieste to test Italian intent instead of rushing Serbia.
Why better: Gives up the immediate Serbia grab for information. If Italy is hostile (moving to TRI), this blocks them; if friendly, Austria pivots south next turn.

Steps 3–4 – F1902M, 3→4 SC: *[EXPLORE]* Fourth consecutive scouting step. A BUD–RUM: a delayed Balkan push that gains the first new center in Fall 1902.
Effect: Reaches 4 centers—one turn behind GPT-5.4's pace, but with superior positional information.
Why better: Three EXPLORE steps identify Russia (not Turkey) as the better expansion target. The 1-SC delay avoids the overcommitment that traps GPT-5.4 for 13 phases.

Phase transition at step 5: EXPLORE→SETUP triggers at exactly step 5 (μ=5.0, σ=0.0 across all 28 episodes). Five turns of observation before acting, regardless of center count (3–5).

Figure 7. Step-level comparison between GPT-5.4 and our method in Diplomacy as Austria (1/2: Setting & Early Game). Continued in Figure 8.

Diplomacy Strategy Comparison (cont'd)

Mid Game (Steps 5–13)

GPT-5.4

Step 9 – F1903M, 5 SC: *[DEFEND]* Orders: A BUD–VIE, A RUM–SEV, F AEG–ION.
Effect: Attacks Sevastopol and the Ionian Sea simultaneously—both fail, and Rumania is lost the following turn.
Strategy: Spreads across three fronts at once. The center count stays at 5 for **13 consecutive phases**.

Step 12 – S1904M, still 5 SC: *[DEFEND]* Orders: A BUD S A GAL–RUM, A GAL–RUM, A BUL H, F AEG H.
Effect: Four hold/support orders with zero territorial gain, all effort directed at retaking Rumania.
Strategy: *Secure Balkan Frontier* is selected for 14 of 15 steps, providing no signal to change approach.

COS-PLAY

Step 9–10 – F1904M→W1904A, 4→5 SC: *[SETUP]* Orders: F APU S F ION—fleet cooperation in the Adriatic and Ionian, establishing a persistent naval coordination chain.
Effect: Centers steadily grow to 5 through sustained mutual support.
Why better: Instead of spreading thin, COS-PLAY repeats one coordination pattern (F APU S F ION, in 12 of 20 steps) that locks down the Adriatic–Ionian corridor.

Step 12–13 – F1905M→W1905A, 5→6 SC: *[DEFEND]* The SETUP→DEFEND transition triggers after expansion. F APU S F ION continues—consistent naval support secures the 6th center.
Effect: Steady growth while GPT-5.4 remains stuck at 5 for 13 phases.
Why better: SETUP→DEFEND acts as a "stop expanding, hold what you have" gate. GPT-5.4 has no such boundary and tries to attack and defend at the same time.

Late Game (Steps 14–19)

GPT-5.4

Step 14–15 – F1904M→W1904A, 5→7 SC: *[ATTACK]* Orders: F AEG S A BUL–CON, A BUL–CON.
Effect: Finally breaks through to Constantinople after 12 turns of stagnation.
Strategy: The breakthrough comes from Turkey collapsing, not from a change in GPT-5.4's play.

Step 18–19 – F1905M, 7 SC: Orders: A BUL–RUM, A CON–SMY, F AEG S A CON–SMY.
Effect: Continues pressing Turkey but achieves no further gains.
Final: **7 SC**. All growth came in one late burst (5 → 7 at step 15).

COS-PLAY

Step 15–16 – F1906M→W1906A, 6→7 SC: *[DEFEND]* Orders: F APU S A TYR–VEN—a decisive Venice strike supported by the persistent Adriatic fleet chain, with A BUD B built for consolidation.
Effect: The 7th center is secured through a coordinated multi-unit operation.
Why better: Venice falls because Tyrolia (positioned since EXPLORE) and the Adriatic fleet chain converge—a multi-phase setup that GPT-5.4's single-skill system cannot reproduce.

Step 17–19 – S–F 1907M, 7 SC: *[SETUP]* Late-game repositioning. F APU S F ION continues, and no center is ever lost.
Final: **7 SC**. Steady growth (3 → 4 → 5 → 6 → 7), one gain per game stage. **Zero centers lost at any point.**

Key Findings

COS-PLAY scouts for four turns before committing; GPT-5.4 rushes to 5 SC by step 2 and then stalls for 13 phases. Three skill phases (EXPLORE→SETUP→DEFEND→SETUP) replace 8+ intention tags (switching rate ∼0.43). Stability floor: min=3 SC versus GPT-5.4's min=1 SC (16 of 60 episodes end at ≤ 3). One repeated naval pattern (F APU S F ION, 12 of 20 steps) outperforms GPT-5.4's scattered fleet orders.

Figure 8. Step-level comparison between GPT-5.4 and our method in Diplomacy as Austria (2/2: Mid & Late Game). See Figure 7 for setting and early game.

Diplomacy Failure Analysis: Stagnation vs. Collapse

COS-PLAY Failure Mode: "Stuck in Support Loop" (5/28 episodes end at 3 SC)

Episode	Power	Most-Repeated Action	Repeat Rate
diplomacy_c40f3dc3	Italy	F NAP S A ROM (14/20 steps)	42%
diplomacy_2b255046	Italy	F NAP S A ROM (15/20 steps)	47%
diplomacy_95bd25f4	Italy	A APU H (9), F NAP S A ROM (7)	53%
diplomacy_e66ffd8e	Austria	F ALB S A VEN–TRI (13/20 steps)	37%
diplomacy_c5f6577c	Germany	A BER–MUN (5), F BAL S A RUH–KIE (5)	21%

Root Cause: The model has a strong **action-1 bias**: action #1 is selected in 85% of all steps (90% in failure episodes vs. 82% in successes). Action 1 is usually a SUPPORT order, so failing episodes get stuck supporting the same unit over and over. Skill transitions still fire (EXPLORE→SETUP→DEFEND), but the action adapter does not vary its orders enough to break out.

GPT-5.4 Failure Mode: "Collapse" (16/60 episodes, 27%, end with ≤ 3 SC)

Episode	Center Trajectory	Failure Pattern
episode_010	3→5→5→5→4→4→2→2→1→1	Peaked step 2, continuous decline for 17 steps
episode_028	3→5→5→5→5→3→3→3→2→2	Peaked step 2, lost 3 centers mid-game
episode_043	3→4→4→4→4→4→3→3→2→2	Slow bleed from mid-game onward

Root Cause: *Secure Defensive Line* stays active for 12–15 consecutive steps, with the intention cycling SURVIVE→DEFEND→SURVIVE. Centers keep dropping, but the skill bank has **no recovery skill** for losing positions. Once the decline starts, there is no way to pivot—the defensive skill orders retreats and disbands that accelerate the loss.

Key Insight & Qualitative Outplay

Stagnation vs. Collapse: COS-PLAY fails by **stagnation**—stuck at 3 SC, never growing. GPT-5.4 fails by **collapse**—grows to 4–5 SC, then drops to 1–2. Stagnation is the safer failure mode: the agent never loses its starting centers. The phased skill structure provides a *floor*—even with poor action selection, the agent does not abandon centers (min=3 vs. GPT-5.4 min=1).

Outplay Example (France, Step 8, F1904M): France holds only 3 SC but has units in aggressive positions: A TYR (deep in Austrian territory), A BEL, F MAO. COS-PLAY plays A BEL S F NTH–HOL—a joint strike with England—taking **3 new centers in one turn** (3 → 6). The preceding 8-turn "holding pattern" (BRE↔MAO cycling) made GPT-5-mini opponents commit elsewhere; the simultaneous strike went unanswered. At the same stage, GPT-5.4's best Austria episode sits at 5 SC with four hold/support orders and *Secure Balkan Frontier* selected for 14 of 15 steps.

Figure 9. Failure analysis for both methods in Diplomacy. COS-PLAY fails by stagnation (5/28 episodes plateau at 3 SC); GPT-5.4 fails by collapse (16/60 decline to 1–2 SC). Stagnation is the safer failure mode.

Diplomacy: Skill Retrieval & Causal Mechanism

Skill Retrieval Stress Test

COS-PLAY: Clean Phase Boundaries

Transition	Trigger	Step (μ)	N
EXPLORE→SETUP	Temporal boundary	5.0 ± 0.0	28/28
SETUP→DEFEND	Reward ≥1.83	11.6 ± 1.2	14/28
SETUP→ATTACK	Low centers (3–4)	13.0 ± 1.0	3/28
DEFEND→SETUP	End-game reposition	17.5 ± 0.8	13/28

Pattern: EXPLORE fires only at 3 SC (opening). ATTACK fires only at 3–4 SC (desperation pivot). DEFEND spans the full 3–7 SC range. The model uses **step number**, not center count, as its primary switch signal—transitions are time-driven.

GPT-5.4: Noisy, State-Independent

Transition	Steps	Centers	N
Secure Balkan→Def. Line	2–19	2–8	184
Def. Line→Secure Balkan	3–19	2–8	145
Pressure Serbia→Secure Balkan	1–19	3–6	61
Def. Line→Fortify Fronts	3–19	3–8	30

Pattern: The top two transitions are inverses, creating a "ping-pong" loop (Balkan→Def.→Balkan→ $\cdots$) across *all* step and center ranges with no consistent trigger. Confidence scores are uniformly low (0.23–0.24)—the retriever cannot tell when each skill applies. POSITION, DEFEND, and ATTACK all appear across the full center range (2–8), with no specialization.

Causal Mechanism: Skills → Temporal Structure → Action Diversity → Better Outcomes

Factor	Success ($c \geq 5$)	Failure ($c \leq 3$)	Mechanism
Action repeat rate	22%	40%	Fewer repeats = more action-space exploration
Action #1 selection	82%	90%	Failures nearly degenerate—always pick first option
Skill transitions	2–3	2–3	Same temporal schedule fires for both
First center gain	Step 3–9	Never	Early growth compounds; stagnation is self-reinforcing

Why skills help: **(i)** They impose *temporal structure*: the EXPLORE→SETUP boundary at step 5 is a hard "stop observing, start acting" cutoff that breaks support loops. **(ii)** SETUP *shifts the action distribution*: the subgoal prompt changes from "scout intentions" to "secure supply centers," which is enough to move past purely defensive orders. **(iii)** Skills provide a *floor, not a ceiling*—they prevent collapse (min=3 vs. GPT-5.4 min=1) even with poor action selection, because the phased structure stops the agent from abandoning centers.

Figure 10. Skill retrieval patterns and causal mechanism in Diplomacy. Skills function as a curriculum schedule for action exploration, as they impose temporal structure that broadens the action distribution and establishes a safety floor.