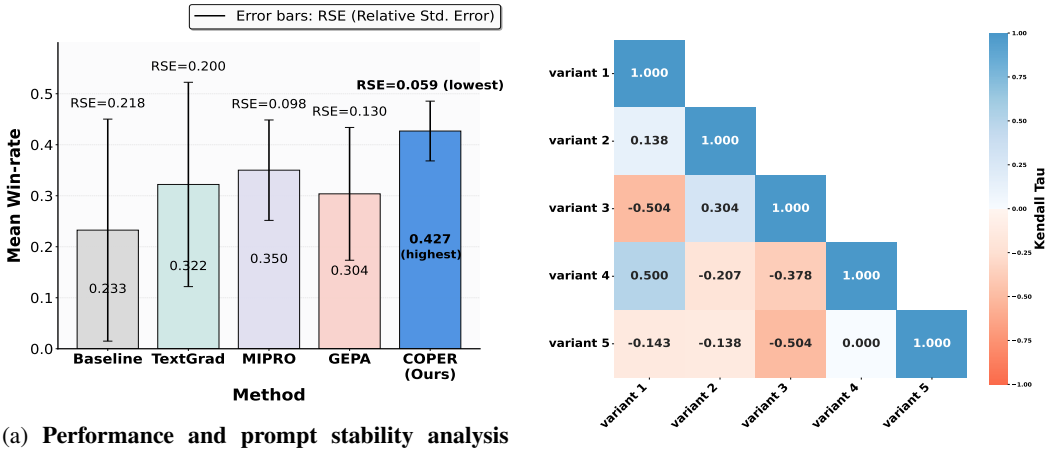
COPER: AGENTIC CONTEXT SIGNIFICANTLY IMPROVES AND STABILIZES LLM IN MULTI-PLAYER GAME

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent multi-player game benchmarks can be sensitive: modest changes to role, system, or judge prompts often flip win-rate rankings under identical decoding; and static, read-once descriptions fail to impart the *game-specific priors* (rules, legality, action→transition effects) needed for consistent play. We document this *context-induced* instability and argue evaluation should be *agentic*: let interaction surface and solidify priors, then evaluate models for both their strength (performance) and reliability (consistency under perturbations). To establish more reliable baselines, we present **COPER**, a backbone-agnostic, tuning-free self-play recipe that (i) *evolves prompts* using a conservative TrueSkill lower-confidence bound, (ii) writes structured reflections into a persistent *experience bank* retrieved across turns to supply rule-aware priors, and (iii) uses *prioritized replay* to revisit rare, informative states for sample-efficient stabilization. Across five text games, COPER raises mean win rate from **24.9%** $\rightarrow$ **49.5%** (GPT-4o-mini) and **21.7%** $\rightarrow$ **44.3%** (Qwen-2.5-7B-Instruct) with a small budget (5$\times$400 self-play games per task), and *stabilizes* agent performance under evaluation. These results show that much of today's LLM game headroom can be unlocked by *context* rather than weight updates, with COPER yielding strong improvements in negotiation games, competitive results in some imperfect-information settings, and RL remaining more effective in perfect-information games.

1 INTRODUCTION



(a) **Performance and prompt stability analysis across different prompt optimization methods.**

(b) **Ranking sensitivity in KUHNPOKER.**

Figure 1: **Left:** We evaluate baseline and other prompt optimization methods by average win-rate and RSE. Our method, COPER, achieved the highest win-rates and the lowest variance, demonstrating enhanced performance and stability. **Right:** With environment and evaluator pools fixed, five *nearly equivalent* prompt variants still flip pairwise outcomes and reshuffle rankings. The heatmap shows Kendall's τ_b for every pair of prompts: blue means very similar rankings ($\tau_b \approx 1$), white means unstable rankings ($\tau_b \approx 0$), and orange means rank reversals ($\tau_b < 0$).

Large language models (LLMs) have rapidly saturated many static benchmarks, leaving limited headroom for further progress on single-turn QA and reasoning datasets such as AIME (AIME, 2024), SWE-Bench (Jimenez et al., 2023), and GPQA (Rein et al., 2024). This saturation has shifted attention toward multi-step evaluations, especially *game-based* benchmarks (Yao et al., 2025; Duan et al., 2024; Topsakal et al., 2024; Fan et al., 2024), which stress long-horizon reasoning and adaptation. Games are a natural testbed: they are easy to simulate, come with well-defined win conditions, and demand capabilities that mirror real-world challenges, e.g., planning under uncertainty, coordination, negotiation, and context adaptation.

Unfortunately, current game-LLM evaluations are found to be sensitive and under-agentic. Firstly, current game evaluations are *prompt sensitive*. Prior work has shown that LLM accuracy can be highly sensitive to prompt phrasing in QA (Mizrahi et al., 2024), and this effect is amplified in interactive games where agents exchange information over many turns. Because prompts couple across agents, judges, and tools, small changes in role or system templates can flip ELO comparisons and reorder models under identical decoding. Our measurements reproduce this phenomenon: near equivalent prompts induce large variations in win rates and produce ranking reversals. As shown in Fig. 1b, in KUHNPOKER, holding the evaluation model fixed, even minor wording changes in the initial game prompt led to ranking reversals (orange cells), as measured by Kendall's τ_b between leaderboards.

Secondly, read-once descriptions lack the *agentic* feedback loop necessary to develop *game-specific priors* such as precise rules, legality constraints, and the effects of actions on game states and payoffs. While benchmarks provide textual rule descriptions, these priors are rarely internalized from a single reading. Without interaction-driven learning, models repeatedly violate rules and exhibit poor long-term strategic play. Unlike Olympiad-style problems that can be solved through careful reasoning alone, games require continuous interaction. This is where players must refine their understanding of the game mechanics and adapt their strategies dynamically based on experience.

Thesis We argue that game-LLM evaluation should *mirror human play*: let interaction surface and solidify priors, then evaluate models for both strength and reliability. We therefore seek a weight tuning-free, agentic evaluation recipe that stabilizes rankings under prompt variation and closes the gap between a model's latent competence and realized in-game performance.

Approach We propose **COPER** (Context Optimized with Prompt, Experience and Replay), an LLM backbone-agnostic, tuning-free framework that pairs (i) **prompt evolution**, (ii) a persistent **experience bank**, and (iii) **prioritized replay**. Prompt evolution treats prompt selection as structured search with a conservative TrueSkill lower-confidence bound $S(p) = \mu - \kappa\sigma$ to favor candidates that are strong and *reliable*. The experience bank consolidates trajectory reflections using CRUD-style updates (Eq. 3) and retrieves relevant insights to refresh the operative description across turns, surfacing rule-aware priors without weight updates. Replay mixes fresh self-play with targeted revisits to rare/informative states via a lightweight gate and priority exponent, accelerating stabilization while preserving coverage (see Eq. 4 and Eq. 5). Together, these components create an *agentic context* at inference time—*prompt* + *experience* + *replay*—that improves adherence to rules and reduces variance across prompts.

Across five text-based games sampled from `SPIN-Bench` (Yao et al., 2025) and `TextArena` (Guertler et al., 2025a), COPER achieves large, budget-efficient gains and more reliable rankings under prompt stratification: for **GPT-4o-mini** (OpenAI, 2024), mean win rate improves from **24.9%** to **49.5%**; for **Qwen-2.5-7B-Instruct** (Yang et al., 2024), from **21.7%** to **44.3%**, using only 5×400 self-play games per task. Moreover, rankings stabilize when the evaluation protocol itself is agentic.

Our contributions can be outlined as follows:

- **Context-induced ranking instability in multi-LLM game benches**. Auditing `SPIN-Bench` and `TextArena` showed that baselines are highly sensitive to how agents, judges and tools are prompted; because prompts couple across agents, small changes in role/system prompts or message templates can flip ELO and model orderings even with identical decoding settings. This effect, amplified by cross-agent interactions and path-dependent dialog, goes beyond single-LM prompt sensitivity and is under-reported despite benches exposing rich prompt hooks. We argue that multi-prompt, prompt-stratified reporting should be mandatory for these benches.

- **Prompt + Experience + Replay (COPER): a simple yet effective recipe.** We introduce a training-free mechanism that (i) writes episodic summaries of trajectories to an experience bank, (ii) retrieves them to edit/evolve per-agent prompts online, and (iii) performs branch-and-replay from flagged states during self-play to guide exploration - thus "fine-tuning without weight updates." COPER unifies verbal-feedback reflection with memory-augmented prompting and experience replay into a single, principled procedure tailored to multi-agent games.
- **Large, sample-efficient gains on multi-LLM games.** As shown in Fig. 1a, COPER achieves substantially higher win rates than prior prompt optimization methods like `MIPRO`, while staying competitive with RL-based baselines like `UnstableBaseline` at a fraction of their computational and rollout cost.

2 Preliminary and Problem Statement

Two-Player Multi-Turn Markov Game. We consider a two-player, turn-based, zero-sum, partially observable environment defined by the tuple (S, A, O, T, R, Ω). Here, S denotes the state space, A the action space, O the observation space, $T : S \times A \to S$ the transition kernel, and $R : S \times A \to \{-1, 0, 1\}$ is a sparse terminal reward. In general, the agent does not observe the full state $s \in S$; instead, it receives a partial observation $o = \Omega(s)$, where $\Omega : S \to O$ is the observation function mapping states to agent observations. Players alternate turns, and we let $p \in \{0, 1\}$ denote the player index. At time step t, the active player $p = (t \bmod 2)$ selects an action $a_t^{(p)} \in A$, while the opponent remains idle. Terminal outcomes are given by $R_0(\tau) = \rho(s_T)$ and $R_1(\tau) = -\rho(s_T)$, where τ denotes the trajectory of play and $\rho : S^{\text{terminal}} \to \{-1, 0, 1\}$ assigns each terminal state s_T to a final outcome. As the interaction length increases, sampling noise, non-stationarity, and error propagation accumulate, resulting in amplified variance in the observed outcomes.

Game Context: Prompt and Experience. We use *context* to denote all information that conditions the model before and during play. Let $C = (p, M)$, where p is the **instruction prompt**: role and core system text fixed at the start of play; M is the **experience memory**, interaction-derived knowledge distilled from self-play and evaluation trajectories and retrieved at inference without weight updates. This contextual prior helps the model interpret transitions and payoffs more effectively, promoting stability over extended interactions.

Full-Context Evaluation. Given a method m with context design space $\mathcal{C}_m$ (e.g., choices of p and M), game suite $\mathcal{G}$, and opponent pool $\mathcal{E}$, each independent run produces a best context $C_r \in \mathcal{C}_m$. We execute n runs. For every game $g \in \mathcal{G}$ and opponent $e \in \mathcal{E}$, we play k rounds; each round consists of two games with swapped first-move order to remove first-move bias. Opponent models use fixed reference contexts specified in Appx. D. Let $\mathrm{WR}_{r,g} \in [0, 1]$ denote the evaluated agent's win rate in game g under context C_r, averaged over all opponents in $\mathcal{E}$ and k rounds. The overall performance for run r is then $x_r = \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \mathrm{WR}_{r,g}$. We report mean performance across runs, $\mathrm{mean}(x_1, \ldots, x_n)$, together with *relative standard error* (RSE), defined as $\mathrm{RSE}(\%) = 100 \times \frac{\mathrm{std}(x_1,\ldots,x_n)}{\mathrm{mean}(x_1,\ldots,x_n)\times\sqrt{n}}$, where lower RSE values indicate greater stability across independent context selections.

Variance Across Prompt and Context. Small wording changes in this template can induce large shifts in both absolute and relative performance, which motivates *multi prompt* evaluation and calibration protocols (Mizrahi et al., 2024; Zhao et al., 2021). We evaluate *state-of-the-art* models (GPT-4o (OpenAI et al., 2024), DeepSeek-R1 Guo et al. (2025), Gemini-2.5-Flash Comanici et al. (2025), Grok-3-Mini (xAI, 2025), GPT-o3-mini (OpenAI, 2025), and Qwen3-235B-A22B-2507 (Qwen et al., 2025)) on KuhnPoker via *round robin* tournaments using five *nearly equivalent* prompts. To quantify ranking sensitivity, we use Kendall's τ_b (Kendall, 1938), which compares the ordering of all model pairs; for two rankings with n_c concordant pairs, n_d discordant pairs, and tie corrections t_x and t_y, the coefficient is $\tau_b = \frac{n_c - n_d}{\sqrt{(n_c + n_d + t_x)\,(n_c + n_d + t_y)}}$. For each prompt pair, we compute Kendall's τ_b between the resulting leaderboards and summarize the values in a heatmap (Fig. 1b). The results show considerable dispersion. Across prompt variants, absolute performance and pairwise rankings frequently reverse, reflecting sensitivity to minor prompt design decisions.

3 THE COPER FRAMEWORK

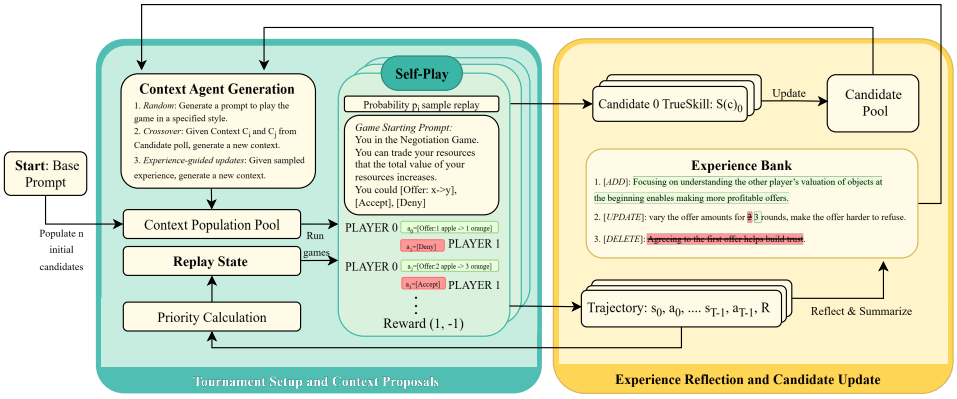


Figure 2: **The COPER Framework**. At each generation, new candidates are proposed by exploring contexts through three strategies: random proposals, crossover, and experience-guided updates. These candidates are then evaluated via self-play, and the best-performing population is used to update the candidate pool. To encourage exploration and mitigate redundant early moves, a prioritized replay module is introduced, enabling efficient search for robust prompts and priors within a single game.

We present **COPER**, an iterative procedure that optimizes prompts and game context to maximize performance and stability in two-player Markov games. In each generation, COPER runs a tournament in a selected game, evolves prompts (Sec. 3.1), derives experience insights from self-play trajectories (Sec. 3.2), and selects state for replay to enable efficient exploration (Sec. 3.3). Fig. 2 provides an overview, and Fig. 3 in Appendix formalizes the procedure.

3.1 COPER CONTEXT OPTIMIZATION LOOP

We begin by describing how COPER evaluates and selects candidate prompts through our context optimization loop. Let $\mathcal{C}_g$ denote the context population (size N) at generation g. Each context $c \in \mathcal{C}_g$ is evaluated by self-play in game G. We maintain Bayesian skill estimates (μ_c, σ_c) via TRUESKILL (Herbrich et al., 2006), where μ_c is the posterior mean skill and σ_c is the posterior standard deviation from observed match outcomes. Selection uses a conservative objective (default $\kappa = 1$):

$$S(c) = \mu_c - \kappa\,\sigma_c. \tag{1}$$

In each tournament at generation g, every context in $\mathcal{C}_g$ plays t rounds of matches against a fixed baseline agent: the same base model instantiated with the *default prompt* only, shown in Appx. Sec. D . For asymmetric games, each round consists of two games with roles swapped to remove first-move bias. The resulting outcomes update TRUESKILL and yield $S(c)$ for selection.

We also maintain a persistent candidate pool $\mathcal{CP}$ with capacity S (initialized as $\mathcal{CP}_0 = \{c_1, \ldots, c_n\}$ from n base-context variants). After generation g, $\mathcal{CP}$ is refreshed by keeping top-scoring elements observed to date, and $\mathcal{C}_{g+1}$ is formed from the best N candidates. Sec. 4.2 covers our configuration.

At each step, COPER forms the next population $\mathcal{C}_{g+1}$ from $\mathcal{C}_g$ and $\mathcal{CP}$ via three proposal operators:

1. **Random proposals**: introduce novel variations to encourage exploration by sampling a playstyle from a fixed catalog and apply small, length-bounded edits to the base context to instantiate that style while preserving legality and interface constraints (Appx. C.1).
2. **Crossover**: recombine high-scoring parents (by $S(c)$) at section- or sentence-level to propagate useful structure (Appx. C.2).
3. **Experience-guided updates**: incorporate insights distilled from trajectory reflections (Sec. 3.2) into targeted prompt edits.

This combination balances exploration and exploitation, progressively accumulating more effective contextual instructions across generations. Following evaluation at generation g, the candidate pool is refreshed to retain the top performers from $\mathcal{CP} \cup \mathcal{C}_g$. After the final generation, COPER returns the best candidate:

$$p^\star \;=\; \arg\max_{c \in \mathcal{C}} S(c). \tag{2}$$

3.2 Experience Reflection

In addition to context optimization, multi-turn games present an asymmetry: during play, agents must reason over uncertain futures, while after the game, analysis can rely on a single realized trajectory, making attribution easier (Andrychowicz et al., 2017). COPER builds on this by prompting the LLM to extract *structured reflections* from complete trajectories.

Experience bank, workflow, and next-generation play. COPER maintains a *permanent memory* $\mathcal{M}$ that persists across all generations. At the end of generation g, completed trajectories τ, together with final outcomes $r(\tau)$ and sampled intermediate states, are collected for reflection. The collector draws a budget of ρ trajectories and elicits up to κ typed insights per trajectory, accumulating them into a working experience set $\mathcal{W}^{(g)}$. Inspired by CRUD (Martin, 1983), the LLM reconciles $\mathcal{W}^{(g)}$ with $\mathcal{M}$ via *create*, *update*, and *delete*:

$$\mathcal{M} \;\leftarrow\; \big(\mathcal{M} \setminus D^{(g)}\big) \;\cup\; U^{(g)} \;\cup\; C^{(g)}, \tag{3}$$

where $D^{(g)}$ discards outdated or conflicting items, $U^{(g)}$ updates matched items with merged versions, and $C^{(g)}$ adds new items from unmatched insights in $\mathcal{W}^{(g)}$.

In generation $g{+}1$, a designated fraction $\pi \in [0, 1]$ of the new agent pool is initialized as *experience-guided*: each such agent receives a sub-sampled context $m^{(g)} \subseteq \mathcal{M}$ drawn directly from the permanent experience bank. The remaining agents are instantiated without additional context. Matches then follow the standard tournament schedule. Upon completion, new trajectories feed the reflection pipeline, insights are computed under the (ρ, κ) budgets, and $\mathcal{M}$ is updated as in Eq. 3.

3.3 Replay

Finally, while experience reflection equips agents with distilled knowledge from past trajectories, it does not ensure rare states will be revisited. To complement this, we introduce a replay mechanism that selectively revisits stored sequences during self-play.

The replay buffer, with capacity B, records cumulative sequences of all player actions, the corresponding game states, and the game's random seed at each timestep. Because storage occurs at each turn within an episode, replayed trajectories need not cover a full game. Invalid moves are retained to preserve the *unaltered course of play*, ensuring that replays faithfully reflect the original gameplay dynamics. To avoid dominance by common action patterns, the buffer *biases sampling toward infrequently encountered trajectories*, encouraging a more diverse and balanced pool of prompt-level insights.

Formally, the priority of a trajectory τ is defined as the inverse of its occurrence count:

$$\text{priority}(\tau) = \frac{1}{N(\tau)}, \tag{4}$$

where $N(\tau)$ is the number of times trajectory τ has appeared in the buffer $\mathcal{B}$.

During sampling, the probability p_i of selecting trajectory τ_i is obtained by raising its priority to a power $\alpha > 0$ (the priority exponent that controls sharpness) and normalizing over the buffer:

$$p_i = \frac{\big(\text{priority}(\tau_i)\big)^\alpha}{\sum_{j=1}^{|\mathcal{B}|} \big(\text{priority}(\tau_j)\big)^\alpha}, \tag{5}$$

where $|\mathcal{B}|$ is the current number of stored trajectories in the buffer.

The buffer is first populated during step 0 and becomes available from step 1. A gating parameter β determines how often games are initialized from the replay buffer rather than played afresh. When

replay is chosen, the stored trajectory prefix (i.e., the sequence of past player actions, corresponding game states and the associated game's random seed) are injected into the environment, ensuring faithful reproductions of past episodes while balancing new exploration.

In our implementation, we set the buffer size to $B = 100{,}000$ and use $\alpha = 0.6$ with a replay gate of $\beta = 0.4$, unless otherwise stated. Since the buffer operates as a sliding window of capacity B, it continuously refreshes with new data while retaining a diverse set of past plays.

4 Experiment Setup

4.1 Game Environments

We performed experiments across three categories of games: **Negotiation**, which tests cooperation and trade-offs (Kramár et al., 2022; Abdelnabi et al., 2024b); **Imperfect Information**, which probes reasoning under uncertainty from partial observations (Brown et al., 2020; Guo et al., 2024); and **Perfect Information**, which emphasizes planning and long-horizon reasoning with full game visibility (Silver et al., 2017a). Details of each game are provided in Appx. H.

4.2 Optimizer Settings

Baseline: Our baseline uses the default `TextArena` (Guertler et al., 2025a) prompts without optimization (examples in Appx. D).

COPER: Using the COPER Framework detailed in Sec. 3, our optimization runs use a population size N of 8 over 5 generations. Each self-play tournament corresponds to one generation with 50 games per optimized agent. Reflection signals are incorporated into the optimization, and token costs of each method are reported in Tab. 5.

For comparison, we benchmark against other prompt optimization methods—`Textgrad` (Yuksekgonul et al., 2024), `MIPRO` (Opsahl-Ong et al., 2024), and `GEPA` (Agrawal et al., 2025)—as well as reinforcement learning baselines including `UnstableBaseline` (Guertler et al., 2025b) and `SPIRAL` (Liu et al., 2025). Detailed setups of their optimization are provided in Appx. E.

4.3 Evaluation Settings

All experiments use **GPT-4o-mini** (OpenAI, 2024) and **Qwen-2.5-7B-Instruct** (Yang et al., 2024) as base models. For prompt-based methods, we perform **three** independent runs. In each run, the optimized prompt and context are evaluated against held-out opponents: Grok-4-Fast-Non-Reasoning (xAI / Grok Team, 2025), Gemini-2.5-Flash-Lite (Comanici et al., 2025), and Qwen3-235B-A22B-Instruct-2507 (Yang et al., 2024). Unless otherwise noted, each run consists of 50 games. We report mean win rates across runs together with standard error (SE). A fixed sampling temperature of $\tau = 1.0$ is used throughout.

For RL-based methods, we train a single policy, select the best checkpoint, and evaluate it over **three** sets of 50 games each against the same opponents. The mean win rate across these sets is reported.

5 Results and Analysis

Observation 1: COPER outperforms other methods in both win rate and robustness. As shown in Tab. 10, COPER outperforms other optimization methods in most of the five game environments. Compared to prompt optimization methods, COPER significantly outperforms `MIPRO` by an average of 12.8% across all tasks on GPT-4o-mini. Our agentic contextual learning method remains competitive with computationally intensive RL-based approaches. For instance, COPER achieves an average win rate of 44.3%, comparable to `UnstableBaseline`'s 46.1%, while being substantially more efficient both in computational cost and game rounds. Specifically, COPER required only **2,000** games per task which is 19 times fewer than `UnstableBaseline`'s **38,000** games.

Beyond improvements in win rate, COPER enhances robustness by reducing the Relative Standard Error (RSE) defined in Sec. 2. Due to the inherent instability in multi-turn gameplay, significant

Table 1: Benchmark results for different approaches using GPT-4o-mini and Qwen2.5-7B-Instruct across multiple tasks. Each win rate is the mean across three evaluation models (sec. 4.3).

Optimizer	Negotiation		Imperfect Info		Perfect Info	Mean Win Rate	Mean RSE
	SimpleNegotiation	TwoDollar	KuhnPoker	Briscola	Simpletak		
GPT-4o-mini							
baseline	31.3%	32.2%	39.1%	0.3%	21.4%	24.9%	25.9%
Textgrad	42.0%	44.6%	55.6%	7.1%	23.6%	34.6%	18.4%
MIPRO	38.4%	50.9%	55.1%	19.7%	19.1%	36.7%	12.4%
GEPA	36.8%	40.4%	52.2%	3.3%	26.9%	32.0%	11.3%
COPER (Ours)	**54.9%**	**52.4%**	**55.6%**	**42.7%**	**41.8%**	**49.5%**	**6.4%**
Qwen2.5-7B-Instruct							
baseline	24.0%	17.1%	49.3%	2.8%	15.1%	21.7%	17.6%
Textgrad	37.1%	29.3%	52.8%	7.1%	22.4%	29.9%	21.7%
MIPRO	42.4%	47.5%	53.8%	2.2%	20.9%	33.4%	7.3%
GEPA	34.4%	31.7%	55.8%	3.3%	19.3%	28.8%	14.8%
UnstableBaseline	41.1%	30.4%	58.4%	**53.3%**	**47.3%**	**46.1%**	24.8%
SPIRAL	45.7%	–	56.7%	–	32.7%	–	–
COPER (Ours)	**48.0%**	**48.4%**	**60.0%**	31.1%	34.0%	44.3%	**6.1%**

variance is observed in the baseline without optimization. Compared to other prompt optimization methods, COPER achieves a lower mean RSE across different games. For example, using GPT-4o-mini, COPER achieves 6.4% RSE compared to `MIPRO`'s 12.4%. Notably, `UnstableBaseline` exhibits increased RSE, indicating that current outcome-based reinforcement learning with sparse rewards remains unstable when optimizing performance in multi-turn, multi-agent scenarios.

Table 2: GPT-4o-mini ablations with progressive module additions.

Setting	TwoDollar	KuhnPoker	Briscola	Mean Win Rate
Baseline	32.2%	39.1%	0.3%	23.8%
+ Prompt Optimization	24.7%	54.7%	2.0%	27.1%
+ Experience	48.7%	57.2%	38.4%	48.1%
+ Replay	52.4%	55.6%	42.7%	50.2%

Observation 2: Experience unlocks LLMs' game-playing capabilities. Tab. 2 presents our ablation study examining the effectiveness of each component. Without the experience module, prompt optimization alone fails to effectively teach the model game dynamics, yielding only marginal improvements (2.0% in BRISCOLA) or even performance drops (-7.5% in TWODOLLAR). However, adding experience augmentation yields large gains by 38.1% and 16.5% respectively. Prompt-only optimization tends to plateau after initial phrasing improvements. However, experience-based updates distill rule clarifications, violation patterns, and counter-strategies into a stable game-specific prior that sustains useful information. Finally, adding replay showed a further 2.1% average boost by revisiting high-priority states to reinforce successful strategies and correct recurring mistakes.

Table 3: Generalization across task

Training Game	Negotiation		Imperfect Info		Perfect Info	Mean Win Rate
	SimpleNegotiation	TwoDollar	KuhnPoker	Briscola	Simpletak	
GPT-4o-mini						
SimpleNegotiation	46.9% (+15.6%)	37.8% (+5.6%)	48.9% (+9.8%)	0.0% (-0.3%)	37.7% (+16.3%)	34.3% (+9.4%)
TwoDollar	31.1% (-0.2%)	48.7% (+16.5%)	53.3% (+14.2%)	1.1% (+0.8%)	47.8% (+26.4%)	36.4% (+11.5%)
KuhnPoker	31.1% (-0.2%)	34.4% (+2.2%)	57.2% (+18.1%)	22.2% (+21.9%)	30.0% (+8.6%)	35.0% (+10.1%)
Briscola	38.9% (+7.6%)	27.8% (-4.4%)	57.8% (+18.7%)	38.4% (+38.1%)	14.3% (-7.1%)	35.4% (+10.6%)
Simpletak	37.8% (+6.5%)	35.6% (+3.4%)	65.0% (+25.9%)	0.0% (-0.3%)	30.7% (+9.3%)	33.8% (+9.0%)

Observation 3: Cross-game generalization of learned context. Tab. 3 presents our cross-game evaluation results. Columns indicate the source game where our method learned its context through self-play. Rows show target games where we evaluate the learned context *zero-shot*, without any fine-tuning. Each cell reports win rates from 50 independent matches against evaluator models.

We found that the learned prompts and context often transfer to unseen environments, improving win rates in most new games. This reveals two key patterns:

Protocol-level skills transfer across game families. Core decision-making components—such as turn management, action formatting, and short-horizon planning—generalize effectively even when payoff structures differ significantly. For example, *SimpleTak* → *KuhnPoker* achieves $+25.9\%$ improvement, and *TwoDollar* → *SimpleTak* yields $+26.4\%$. These gains suggest that the learned prompts create a general "decision scaffold" that extends beyond game-specific heuristics.

Transfer exhibits directional asymmetry. The transfer effectiveness depends on the direction of knowledge transfer. Negotiation strategies from TWODOLLAR improve performance on SIMPLENEGOTIATION ($+5.6\%$), but the reverse is negligible (-0.2%). Similarly, *Briscola* → *SimpleTak* shows negative transfer (-7.1%) despite strong within-family performance. This asymmetry suggests that transfer success depends also on the alignment between source and target game mechanics; for instance, card-tracking strategies may not translate to perfect-information board games.

Table 4: Generalization across models.

Model	Briscola	KuhnPoker	TwoDollar	Mean
Self-play on gpt-4o-mini to find the best context				
Gemini-2.5-flash-lite	22.7%	48.7%	20.0%	30.5%
Gemini-2.5-flash-lite (with best context)	41.3% (+18.6%)	60.7% (+12.0%)	50.0% (+30.0%)	50.7% (+20.2%)
Grok-4-fast-non-reasoning	49.3%	58.7%	24.7%	44.2%
Grok-4-fast-non-reasoning (with best context)	41.3% (-8.0%)	52.7% (-6.0%)	48.0% (+23.3%)	47.3% (+3.1%)

Observation 4: Learned context does not always transfer across models. As shown in Tab. 4, we test whether a context learned via self-play on GPT-4o-mini can generalize to other models. Specifically, we apply the prompts and experience produced by COPER to Gemini-2.5-flash-lite and Grok-4-fast-non-reasoning, and evaluate against the same opponent pool described in Sec. 4.3. The results reveal a mixed picture. This highlights that learned context is not universally portable across architectures, further underscoring the need for per-model agentic context optimization.

Table 5: Output token cost for each prompt optimization method (exact counts).

Optimizer	SimpleNegotiation	KuhnPoker	SimpleTak	Avg. tokens
Textgrad	842	986	938	922
MIPRO	145,864	162,084	754,534	354,161
GEPA	110,325	119,365	111,907	113,865
COPER (Ours)	87,364	94,160	89,152	90,575

Observation 5: COPER is computationally and sample efficient. Beyond its performance gains, COPER is highly efficient. As shown in Tab. 5, it uses only 91K output tokens, which is one-quarter of `MIPRO` (354K) and 20% fewer than `GEPA` (113K). `Textgrad` consumes very few tokens ($\sim$1K) since it updates prompts via a single differentiable loss, but its optimization capacity is limited. Overall, experience-guided prompts strike a better balance of efficiency and effectiveness than reflection- or gradient-only baselines. Notably, incorporating replay with a $\beta = 0.6$ reduced our token usage on SIMPLETAK by 22.7% compared to runs without replay, further underscoring its role in efficient exploration.

6 RELATED WORKS

6.1 PROMPT OPTIMIZATION

Automatic prompt optimization has evolved into a principled, black-box search over prompt seeds, feedback signals, candidate generation, and selection strategies (Ramnath et al., 2025). Programmatic frameworks such as `DSPy` compile LM pipelines and optimize prompts directly toward a user metric (Khattab et al., 2023); gradient-via-text methods propagate natural-language feedback through computation graphs to update intermediate decisions (Yuksekgonul et al., 2024). Recent systems jointly search over agentic patterns and prompt contents (Spiess et al., 2025), offer zero-configuration prompt pipelines with meta-optimizers and `DSPy` backends (Murthy et al., 2025), or meta-learn general system prompts while adapting user prompts (Choi et al., 2025). COPER complements this line by targeting interactive games: it evolves context via conservative selection, writes structured reflections to a persistent experience bank, and reuses them across turns. It provides rule-aware priors without weight updates while remaining backbone-agnostic. For a detailed comparison of our approach and existing prompt optimization methods, please refer to Appx. F.

6.2 LLM FOR GAMES

Early multi-agent evaluations used role prompts and multi-turn dialogue to probe cooperation and theory-of-mind (Abdelnabi et al., 2024a). Community arenas expanded coverage: `TextArena` provides competitive text games with online TrueSkill ranking (Guertler et al., 2025a); `SPIN-Bench` combines planning, cooperative/competitive play, and negotiation, highlighting limits in deep reasoning and coordination (Yao et al., 2025); and `GT-Bench` evaluates strategic play in board and card games (Duan et al., 2024). Prompt design strongly affects move quality (Topsakal et al., 2024), and moving toward off-the-shelf games required harnesses to reduce perception and prompt brittleness (Hu et al., 2025). COPER addresses this brittleness in text-based game settings directly: it treats evaluation as agentic context construction, stabilizing rankings under prompt variation while improving adherence to game capabilities underexplored by fixed-prompt protocols.

6.3 SELF-PLAY AND EVOLUTIONARY LLM

Classical self-play (`AlphaGo/AlphaZero`) established competitive self-improvement through repeated matches and selection (Silver et al., 2017b; 2016). LLM variants close the loop without large curated corpora: `Absolute Zero` leverages data-free RLVR to attain strong math/coding results (Zhao et al., 2025); `SPIRAL` frames multi-turn reasoning as zero-sum self-play (Liu et al., 2025); and language self-play improves instruction following via self-generated interactions (Kuba et al., 2025). Evolutionary approaches perform reflective prompt/program search (e.g., `GEPA` outperforming RL baselines; evolutionary coding agents) (Agrawal et al., 2025; Novikov et al., 2025). COPER combines these ideas in a tuning-free way: it performs evolutionary context search guided by a reliability-aware objective (TrueSkill LCB), augments it with persistent experience to supply game-specific priors, and uses prioritized replay to revisit rare informative states, yielding stronger and more reliable in-game performance without parameter updates.

7 LIMITATION AND FUTURE WORK

While COPER shows strong gains and stable outcomes via prompts, experience, and replay, the specific contributions of *weighted tuning* versus *experience-centric context optimization* remain undercharacterized. Our comparisons emphasize parameter-efficient RL baselines—REINFORCE (Sutton et al., 1999) with LoRA adapters (Hu et al., 2022)—to control compute, and we have only partial comparisons to `SPIRAL`; thus the strength of full-parameter policy-gradient baselines (e.g., PPO (Schulman et al., 2017)) may be understated. A fairer assessment would include full-parameter optimizers under larger training budgets. Finally, our benchmark currently spans five games and relatively lightweight base models due to budget limits, leaving broader task and model coverage for future work.

COPER currently targets game environments, the pipeline is readily applicable to genuinely *multi-step* settings. For example, instruction following in ALFWorld and realistic browser-based tasks in WebArena (Shridhar et al., 2020; Zhou et al., 2023). A complementary direction is to convert short-lived *episodic* gains into durable, *parametric* capabilities by consolidating context memory into model weights via targeted knowledge editing and adapter-to-base consolidation/mixture schemes, aiming for the persistence of full fine-tuning while preserving experience-based sample efficiency.

8 CONCLUSION

We identified a critical challenge in agent systems: amplified context sensitivity in multi-turn multi-agent interactions, particularly in game-based benchmarks. To address this, we introduced COPER, a training-free, backbone-agnostic framework with three synergistic components: evolutionary prompt optimization, persistent experience bank, and prioritized replay mechanism. COPER achieves significant improvements across multiple games while maintaining computational efficiency. The learned contextual knowledge generalizes to different models and unseen games. Ablation studies confirm each component's essential contribution. This work highlights amplified context sensitivity in multi-agent scenarios and opens new directions for developing robust agent systems for complex interactive environments.

Reproducibility Statement

We will release the full source code, configuration files, and instructions for COPER, covering all algorithms and settings. Every effort has been made to ensure that the results presented in this paper are reproducible. Our implementation has been validated through multiple independent runs and diverse evaluation settings to confirm robustness and reproducibility.

Ethics Statement

Evaluating LLMs in multi-agent games advances reasoning, negotiation, and strategy, with potential benefits for education and decision support. At the same time, such environments may incentivize deceptive or manipulative behavior, especially in negotiation tasks. Our study is limited to controlled settings without human data, but future applications in high-stakes domains require safeguards, oversight, and alignment research. While we did not study fairness or bias directly, we see multi-agent games as a useful testbed for identifying such issues early.

References

Sahar Abdelnabi, Amr Gomaa, Sarath Sivaprasad, Lea Schönherr, and Mario Fritz. Cooperation, competition, and maliciousness: Llm-stakeholders interactive negotiation. *Advances in Neural Information Processing Systems*, 37:83548–83599, 2024a.

Sahar Abdelnabi, Amr Gomaa, Sarath Sivaprasad, Lea Schönherr, and Mario Fritz. LLM-deliberation: Evaluating LLMs with interactive multi-agent negotiation game, 2024b. URL https://openreview.net/forum?id=cfL8zApofK.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.

AIME. Aime problems and solutions. https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions, 2024.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Noam Brown, Anton Bakhtin, Adam Lerer, and Qucheng Gong. Combining deep reinforcement learning and search for imperfect-information games. *CoRR*, abs/2007.13544, 2020. URL https://arxiv.org/abs/2007.13544.

Yumin Choi, Jinheon Baek, and Sung Ju Hwang. System prompt optimization with meta-learning. *arXiv preprint arXiv:2505.09666*, 2025.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. https://github.com/openai/baselines, 2017.

Jinhao Duan, Renming Zhang, James Diffenderfer, Bhavya Kailkhura, Lichao Sun, Elias Stengel-Eskin, Mohit Bansal, Tianlong Chen, and Kaidi Xu. Gtbench: Uncovering the strategic reasoning capabilities of llms via game-theoretic evaluations. *Advances in Neural Information Processing Systems*, 37:28219–28253, 2024.

Caoyun Fan, Jindou Chen, Yaohui Jin, and Hao He. Can large language models serve as rational players in game theory? a systematic analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17960–17967, 2024.

Leon Guertler, Bobby Cheng, Simon Yu, Bo Liu, Leshem Choshen, and Cheston Tan. Textarena. *arXiv preprint arXiv:2504.11442*, 2025a.

Leon Guertler, Tim Grams, Zichen Liu, and Bobby Cheng. UnstableBaselines, June 2025b. URL https://github.com/LeonGuertler/UnstableBaselines.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Jiaxian Guo, Bo Yang, Paul Yoo, Bill Yuchen Lin, Yusuke Iwasawa, and Yutaka Matsuo. Suspicion-agent: Playing imperfect information games with theory of mind aware GPT-4, 2024. URL https://openreview.net/forum?id=ug8wDSimNK.

Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill™: a bayesian skill rating system. *Advances in neural information processing systems*, 19, 2006.

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

Lanxiang Hu, Mingjia Huo, Yuxuan Zhang, Haoyang Yu, Eric P. Xing, Ion Stoica, Tajana Rosing, Haojian Jin, and Hao Zhang. lmgame-bench: How good are llms at playing games?, 2025. URL https://arxiv.org/abs/2505.15146.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 1938.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, et al. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

János Kramár, Tom Eccles, Ian Gemp, Andrea Tacchetti, Kevin McKee, Mateusz Malinowski, Thore Graepel, and Yoram Bachrach. Negotiation and honesty in artificial intelligence methods for the board game of diplomacy. *Nature Communications*, 13, 12 2022. doi: 10.1038/s41467-022-34473-5.

Jakub Grudzien Kuba, Mengting Gu, Qi Ma, Yuandong Tian, and Vijai Mohan. Language self-play for data-free training. *arXiv preprint arXiv:2509.07414*, 2025.

Harold W. Kuhn. A simplified two-person poker. *Contributions to the Theory of Games*, (2):97–115, 1951. Available as PDF from Rutgers Math: https://sites.math.rutgers.edu/~zeilberg/akherim/PokerPapers/Kuhn1951.pdf, accessed 2025-09-23.

Bo Liu, Leon Guertler, Simon Yu, Zichen Liu, Penghui Qi, Daniel Balcells, Mickel Liu, Cheston Tan, Weiyan Shi, Min Lin, et al. Spiral: Self-play on zero-sum games incentivizes reasoning via multi-agent multi-turn reinforcement learning. *arXiv preprint arXiv:2506.24119*, 2025.

James Martin. *Managing the Data-base Environment*. Prentice-Hall, Englewood Cliffs, NJ, 1983. ISBN 0135505828.

John McLeod. Briscola — card game rules (including other briscola www sites and software). Pagat.com, 2023. URL https://www.pagat.com/aceten/briscola.html#other. Last updated: 30 June 2023. Accessed: 2025-09-23.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. State of what art? a call for multi-prompt llm evaluation. *Transactions of the Association for Computational Linguistics*, 2024. arXiv:2401.00595.

Rithesh Murthy, Ming Zhu, Liangwei Yang, Jielin Qiu, Juntao Tan, Shelby Heinecke, Caiming Xiong, Silvio Savarese, and Huan Wang. Promptomatix: An automatic prompt optimization framework for large language models. *arXiv preprint arXiv:2507.14241*, 2025.

John F. Nash. The bargaining problem. *Classics in Game Theory*, 1950. URL https://api.semanticscholar.org/CorpusID:153422092.

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.

OpenAI. Gpt-4o mini: Advancing cost-efficient intelligence. OpenAI Blog / Model Card, 2024. URL https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/. Text & vision model, 128k token context window, launched July 18, 2024.

OpenAI. Openai o3-mini. https://openai.com/index/openai-o3-mini/, 2025. Accessed: 2025-05-12.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens,

Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. Gpt-4o system card, 2024. URL https://arxiv.org/abs/2410.21276.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. *arXiv preprint arXiv:2406.11695*, 2024.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Kiran Ramnath, Kang Zhou, Sheng Guan, Soumya Smruti Mishra, Xuan Qi, Zhengyuan Shen, Shuai Wang, Sangmin Woo, Sullam Jeoung, Yawei Wang, et al. A systematic survey of automatic prompt optimization techniques. *arXiv preprint arXiv:2502.16923*, 2025.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Patrick Rothfuss. *The Wise Man's Fear*. DAW Books, New York, 2011. ISBN 978-0-7564-0473-4. https://www.isfdb.org/cgi-bin/pl.cgi?222913.

Mary Rowe. Lecture notes: Negotiation and conflict management. https://ocw.mit.edu/courses/15-667-negotiation-and-conflict-management-spring-2001/pages/lecture-notes/, 2001. Accessed: 2025-09-23.

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

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815, 2017a. URL http://arxiv.org/abs/1712.01815.

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

Claudio Spiess, Mandana Vaziri, Louis Mandel, and Martin Hirzel. Autopdl: Automatic prompt optimization for llm agents. *arXiv preprint arXiv:2504.04365*, 2025.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Oguzhan Topsakal, Colby Jacob Edell, and Jackson Bailey Harper. Evaluating large language models with grid-based game competitions: an extensible llm benchmark and leaderboard. *arXiv preprint arXiv:2407.07796*, 2024.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

William Brown. Verifiers: Environments for llm reinforcement learning. https://github.com/willccbb/verifiers, 2025. Commit abcdefg • accessed DD Mon YYYY.

xAI. Grok 3 Beta — the age of reasoning agents, 2025. URL https://x.ai/news/grok-3.

xAI / Grok Team. Grok-4-fast non-reasoning. Model Card / Documentation on xAI site, 2025. URL https://docs.x.ai/docs/models/grok-4-fast-non-reasoning. Non-reasoning variant of Grok-4-Fast, optimized for speed / cost-efficiency.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Jianzhu Yao, Kevin Wang, Ryan Hsieh, Haisu Zhou, Tianqing Zou, Zerui Cheng, Zhangyang Wang, and Pramod Viswanath. Spin-bench: How well do llms plan strategically and reason socially? *arXiv preprint arXiv:2503.12349*, 2025.

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

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. *arXiv preprint arXiv:2102.09690*, 2021. URL https://arxiv.org/abs/2102.09690.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

A LLM USAGE

Large language models were used to assist in the writing and polishing of this manuscript. Specifically, we employed LLMs to refine language, improve readability, ensure clarity in certain sections, check grammar, rephrase sentences, and enhance overall presentation. Importantly, the LLMs were not involved in research modeling or experimental design; all research ideas, analyses, and conclusions were developed and validated by the authors.

B ALGORITHM EXPLANATION

Algorithm 1 COPER: Context Optimization with Reflection

Require: Base context c_{base}, analyzer LLM A, environment G, evaluated model family $\mathcal{F}$, population size N, generation count T, operator ratios $(r_{\text{rand}}, r_{\text{cross}}, r_{\text{exp}})$ with $r_{\text{rand}}+r_{\text{cross}}+r_{\text{exp}} = 1$, experience fraction π, reflection budgets (ρ, L), scoring $S(c) = \mu_c - \kappa\sigma_c$ (default κ=1)

$\mathcal{CP}_0 \leftarrow \{c_{\text{base}}\} \cup \{\textsc{StyleMix}(\mathsf{A}, c_{\text{base}}, i)\}_{i=1}^{N-1}$ ▷ Initialize candidate pool (random proposals)
$\mathcal{C}_0 \leftarrow \textsc{SelectTopN}(\mathcal{CP}_0, N; S)$ ▷ Initialize population
Initialize experience bank $\mathcal{M} \leftarrow \varnothing$
for $g = 0$ **to** $T-1$ **do**
  $\mathsf{Pop} \leftarrow \textsc{InstantiateAgents}(\mathcal{C}_g, \mathcal{F}, \pi, \mathcal{M})$ ▷ π fraction receive subsample of $\mathcal{M}$
  $\mathcal{R}_g \leftarrow \textsc{Tournament}(\mathsf{Pop}, G)$ ▷ Self-play trajectories
  $(\mu, \sigma) \leftarrow \textsc{UpdateTrueSkill}(\mathcal{C}_g, \mathcal{R}_g)$; $S(c) \leftarrow \mu_c - \kappa\sigma_c$
  $\mathcal{W}^{(g)} \leftarrow \textsc{Reflect}(\mathsf{A}, \mathcal{R}_g; \rho, L)$ ▷ Typed insights per trajectory
  $\mathcal{M} \leftarrow \textsc{CRUD\_Update}(\mathcal{M}, \mathcal{W}^{(g)})$ ▷ Create/Update/Delete
  $n_{\text{rand}} \leftarrow \lfloor N r_{\text{rand}} \rfloor$; $n_{\text{cross}} \leftarrow \lfloor N r_{\text{cross}} \rfloor$; $n_{\text{exp}} \leftarrow N - n_{\text{rand}} - n_{\text{cross}}$
  $\mathcal{U}_{\text{rand}} \leftarrow \textsc{RandomProposals}(\mathsf{A}, \mathcal{CP}_g, n_{\text{rand}})$ ▷ Style-guided edits
  $\mathcal{U}_{\text{exp}} \leftarrow \textsc{ExperienceGuided}(\mathsf{A}, \mathcal{CP}_g, \mathcal{M}, n_{\text{exp}})$ ▷ Edits informed by $\mathcal{M}$
  $\mathcal{P} \leftarrow \textsc{SampleParents}(\mathcal{CP}_g, n_{\text{cross}})$ ▷ Inverse-rank sampling within elite set
  $\mathcal{U}_{\text{cross}} \leftarrow \textsc{Crossover}(\mathsf{A}, \mathcal{P})$ ▷ Section-/sentence-level recombination
  $\mathcal{B} \leftarrow \mathcal{CP}_g \cup \mathcal{U}_{\text{rand}} \cup \mathcal{U}_{\text{exp}} \cup \mathcal{U}_{\text{cross}}$
  $\mathcal{C}_{g+1} \leftarrow \textsc{SelectTopN}(\mathcal{B}, N; S)$; $\mathcal{CP}_{g+1} \leftarrow \textsc{RetainTop}(\mathcal{B}; S)$
end for
return $c^\star = \arg\max_{c \in \mathcal{CP}_T} S(c)$

Algorithm 2 Replay-Augmented Tournament Step (extension of Alg. 1)

Require: Replay buffer $\mathcal{B}$, priority exponent α, replay gate β

for each scheduled tournament game in Alg. 1 **do**
  $u \leftarrow \mathcal{U}(0, 1)$
  if $u < \beta$ **and** $|\mathcal{B}| > 0$ **then**
    $\tau \leftarrow \textsc{SampleReplay}(\mathcal{B}, \alpha)$ ▷ Prioritized sampling: $p_i = \dfrac{\big(\text{priority}(\tau_i)\big)^\alpha}{\sum_j \big(\text{priority}(\tau_j)\big)^\alpha}$
    Initialize game with τ's stored trajectory prefix ▷ Replay episodes excluded from TrueSkill updates
  else
    Play a fresh game as in Alg. 1
    Update TrueSkill scores with the resulting match outcome
  end if
  $\mathcal{B} \leftarrow \textsc{Insert}(\mathcal{B}, \tau)$ ▷ Push trajectory with updated inverse-frequency priority
end for

Figure 3: (Top) Self-play loop integrating candidate and population pools, trajectory reflection, and experience-guided prompt evolution. (Bottom) Replay buffer sampling procedure for prioritizing high-interest states.

C PROMPT OPTIMIZATION OPERATORS

We instantiate two lightweight proposal operators that generate candidates for the next population. Defaults are fixed to concrete values for reproducibility.

C.1 RANDOM PROPOSALS (STYLE-GUIDED AUGMENTATION)

Objective. Inject controlled diversity by editing a base context c to reflect a sampled playstyle while preserving legality and interface constraints.

Style catalog. A fixed library $\mathcal{S}$ spanning core play patterns (aggressive, defensive, analytical, creative, strategic, adaptive, balanced), tactical approaches (opportunistic, conservative, risk-taking, methodical, intuitive, predictive, reactive, proactive, experimental, systematic), game-specific strategies (positional, territorial, sacrificial, blocking-focused, center-control, edge-control, fork-creating, trap-setting, opening-focused, endgame-focused), cognitive styles (minimax-oriented, probabilistic, rule-based, principle-driven, context-aware, meta-gaming, exploitative, counter-play), and behavioral patterns (deceptive, transparent, unpredictable, consistent, alternating, escalating, de-escalating, mirroring, contrarian, harmonizing).

Procedure. Sample $s \sim \mathrm{Unif}(\mathcal{S})$ and ask the base model to produce c' by (i) inserting a brief style preface and (ii) making length-bounded edits to directives to embody s. Allowed edits: token substitution, clause insertion/deletion, and reordering; tool descriptions, legality reminders, and input/output schema must remain intact.

C.2 CROSSOVER (TEMPLATE RECOMBINATION)

Objective. Recombine high-value elements from two parents to propagate useful structure while maintaining a coherent, compact context.

Parent selection. From the candidate pool $\mathcal{CP}$, sample two parents $(c^{(1)}, c^{(2)})$ using inverse-rank sampling with power parameter $r = 1.5$: if $\mathrm{rank}_S(c) \in \{1, 2, \ldots\}$ (1 is best), then $P(c) \propto \mathrm{rank}_S(c)^{-r}$. Restrict to the elite set comprising the top $q = 0.25$ fraction by $S(\cdot)$; break ties by maximizing Hamming distance between section signatures to encourage diversity.

Granularity. Always perform sentence-level crossover on individual directives.

Merge procedure. (i) Extract candidate fragments from $c^{(1)}$ and $c^{(2)}$, prioritizing the higher-S parent; (ii) prompt the base model to synthesize a single coherent template from these fragments; (iii) enforce maximum sentence length and overall token budget. The resulting child is c'.

D BASE PROMPT EXAMPLES

D.1 BASE SYSTEM PROMPT

> You are a competitive game player. Make sure you read the game instructions carefully, and always follow the required format.

D.2 NEGOTIATION GAMES

SimpleNegotiation Game Starting Prompt

You are Player 0 in the Negotiation Game.
You have some resources, and your task is to trade such that the total value of your resources increases.
The resources and associated values you currently have are:
+ [Wheat] Qty: 14 Value: 6
+ [Wood] Qty: 15 Value: 11
+ [Sheep] Qty: 18 Value: 18
+ [Brick] Qty: 12 Value: 27
+ [Ore] Qty: 22 Value: 38

At each turn, you can talk to your opponent and make a trade offer.
Use the following special tokens for actions:
- '[Offer: 3 Sheep, 2 Ore -> 5 Brick, 2 Sheep]': [Offer: Offered Resources -> Requested Resources]
- '[Accept]': To accept an incoming offer.
- '[Deny]': To deny an incoming offer (default).

The game lasts for 10 turns in total.

Figure 4: SimpleNegotiation Game Starting Prompt

TwoDollar Game Starting Prompt

[GAME] TWO DOLLAR NEGOTIATION GAME
You are Player 0 negotiating with another player over how to split $2.00 total. There are 20 maximum rounds.
GAME RULES:
- You must reach an agreement on how to split the $2.00
- If no agreement is reached, both players get $0.00
- You have secret instructions that only you can see (below)

REQUIRED ACTION FORMAT:
Always provide your reasoning/persuasion BEFORE the bracketed action.
Any text after the bracketed action will be ignored.
Available actions:
- Make a proposal: "I think this is fair because... [Propose] $X.XX"
- Accept current proposal: "This works for me because... [Accept]"
- Reject current proposal: "I can't accept this because... [Reject]"

=== YOUR SECRET INSTRUCTIONS ===
This person you are dealing with probably will not want to tell you about it but he or she has had a terribly hard month. This past month was full of disaster: money troubles, family difficulties, and none of it your colleague's fault. Through it all, your colleague has been bearing up. Nevertheless, this is zero-sum bargaining, so get all you can. Make up whatever story you wish, and get as much of the $2.00 as possible. (Just remember that your colleague has been having a very hard time.)
VICTORY CONDITION: Reach any deal while being appropriately sympathetic
FAILURE CONDITION: No deal reached OR be too harsh given their difficulties
[GAME] === ROUND 1 of 20 ===

Figure 5: TwoDollar Game Starting Prompt

D.3 IMPERFECT INFORMATION GAMES

KuhnPoker Game Starting Prompt

You are Player 0 in a 3 round game of Kuhn Poker.
Game Rules:
- Kuhn Poker uses a 3-card deck with J, Q, K (J lowest, K highest)
- Each player antes 1 chip and receives 1 card each round (note that the cards are dealt without replacement, so you cannot have the same card as your opponent).
- Game continues for 3 rounds
- The player with the most chips after all rounds wins
Action Rules:
- '[check]': Pass without betting (only if no bet is on the table)
- '[bet]': Add 1 chip to the pot (only if no bet is on the table)
- '[call]': Match an opponent's bet by adding 1 chip to the pot
- '[fold]': Surrender your hand and let your opponent win the pot
Starting round 1 out of 3 rounds. Your card is: 'Q'
Player 1, submitted move: '[bet]'.
Your available actions are: '[fold]', '[call]'

Figure 6: KuhnPoker Game Starting Prompt

Briscola Game Starting Prompt

You are playing Briscola - Player 0.
Goal: Win tricks and collect the most points (120 total points in the deck).
Card Points: A=11, 3=10, K=4, Q=3, J=2, others=0
Card Power: A > 3 > K > Q > J > 7 > 6 > 5 > 4 > 2
Trump cards beat non-trump cards regardless of power.

Action: '[play X]' where X is the position (1-3) of the card in your hand

Briscola game started! Trump suit: ♣ (Trump card: Q♣)
Your hand:
1. J♠ [2 pts]
2. K♣ [4 pts] (TRUMP)
3. A♦ [11 pts]

No cards played yet this trick.

Scores: Player 0: 0 pts | Player 1: 0 pts
Trump suit: ♣ | Cards left in deck: 34

Play a card using [play X]

Figure 7: Briscola Game Starting Prompt

D.4 PERFECT INFORMATION GAMES

SimpleTak Game Starting Prompt

You are Player 0 in SimpleTak.
On the board, your stones appear as 'O' and your opponent's stones appear as 'X'.

On your turn, choose one empty cell (by its numbered index) and place your stone there. For example, '[12]' places your stone in cell 12.

Your objective is to form a continuous path of your stones that connects two opposite edges of the board (top-to-bottom or left-to-right).

Current Board:

```
+----+----+----+----+
| 0  | 1  | 2  | 3  |
+----+----+----+----+
| 4  | 5  | 6  | 7  |
+----+----+----+----+
| 8  | 9  | 10 | 11 |
+----+----+----+----+
| 12 | 13 | 14 | 15 |
+----+----+----+----+
```

Available Moves: [0], [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15]

Figure 8: SimpleTak Game Starting Prompt

E EXPERIMENTAL SETUP AND BASELINE DETAILS

We incorporate three prompt optimization methods to refine prompts using tournament trajectories. Specifically, we leverage offline trajectories collected during the tournament's self-play process to improve the agents' prompts. The experimental settings are as follows: the number of generations is set to 5, the population size to 8, the number of self-play rounds to 25, and the number of evaluation rounds to 25. We discuss `Textgrad` in detail in Section E.1, describe our implementation of `MIPRO` in Section E.2, and provide a comprehensive overview of `GEPA` in Section E.3. Training details for `UnstableBaseline` are presented in Section E.4.

E.1 TEXTGRAD

Textgrad (Yuksekgonul et al., 2024) is a framework that performs "text differentials" to optimize prompts. Within this framework, a text-based loss function analyzes errors, which are then back-propagated to the original prompt through the `Textgrad` engine. In our case, the goal is to optimize the system prompt of the agent using the trajectories generated under the current system prompt. We design a text-based loss that highlights deficiencies in the generated trajectories. The `Textgrad` backpropagation engine then propagates gradients back to the system prompt, updating it accordingly. The loss template we adopt is shown in Figure 9.

For each optimization step, we concatenate multiple trajectories, embed them into the template, and use the completed template as the loss input. To ensure balanced feedback, we select an equal number of win, loss, and draw trajectories. This design allows the `Textgrad` engine to develop a more comprehensive understanding of the current system prompt's overall game-play patterns.

Table 6: Performance of the `Textgrad` method across three independent trials using GPT-4o-mini and Qwen2.5-7B-Instruct. Results are reported as mean win rates with standard deviations.

Textgrad	Negotiation		Imperfect Info		Perfect Info
	SimpleNegotiation	TwoDollar	KuhnPoker	Briscola	Simpletak
GPT-4o-mini					
Trial 1	41.3%	48.3%	58.7%	1.3%	25.3%
Trial 2	44.7%	41.3%	56.0%	2.0%	23.3%
Trial 3	40.0%	44.0%	52.0%	18.0%	22.0%
Avg.	42.0%	44.6%	55.6%	7.1%	23.6%
Std.	2.4	3.5	3.4%	9.4%	1.7%
Qwen2.5-7B-Instruct					
Trial 1	40.0%	38.0%	51.3%	3.3%	18.0%
Trial 2	34.0%	34.0%	54.7%	16.7%	22.7%
Trial 3	37.3%	16.0%	52.7%	1.3%	26.7%
Avg.	37.1%	29.3%	52.8%	7.1%	22.4%
Std.	3.0	11.7%	1.7%	8.3%	4.3%

Text-based loss template for Textgrad

You are an objective evaluator for a two-player zero-sum game agent's SYSTEM PROMPT.

Goal of the SYSTEM PROMPT (what it MUST enforce):
- Maximize the agent's win rate.
- Minimize the opponent's win rate.
- Have strategies that lead to victory.
- Ensure all moves strictly follow game rules and formats.

Here are some game trajectories using the current SYSTEM PROMPT:
{{trajectory examples}}

Identify specific weaknesses or flaws in the SYSTEM PROMPT that may have led to losses or suboptimal plays.
Do NOT suggest improvements or rewrites, only identify weaknesses.
Be very concise and specific.

Figure 9: Text-based loss template for Textgrad

E.2 MIPRO

MIPRO (Opsahl-Ong et al., 2024) optimizes prompts based on downstream task performance. In our work, we adopt the `MIPROv2` implementation provided by the `Dspy` library (Khattab et al., 2023). The optimization procedure consists of three main steps: (1) Sampling examples: For each candidate prompt, `MIPRO` samples a set of examples. (2) Proposing prompts: New system prompts are proposed by a propose model based on the current system prompt, along with additional game-related information such as the program description, data description, random sampling tips, and few-shot examples. (3) Evaluation through trials: Several trials are conducted to evaluate which combination of proposed prompts and few-shot examples yields the best performance. A Bayesian search strategy is then applied to guide the selection of the next candidate combination, improving efficiency and reducing computational cost.

In our experiments, we only have access to offline game data. Therefore, we treat each step in a trajectory as an individual data point. For each step, we record the outcome (win, loss, or draw) of the trajectory it belongs to. `MIPRO`'s evaluation metric is defined based on the model's re-inference of these steps: (1) If the model outputs an invalid action (i.e., one that does not conform to the required format), the score is 0. (2) For steps from winning trajectories, if the model predicts the same action

as the original step, the score is 1; otherwise, it is 0. (3) For steps from losing trajectories, if the model predicts the same action, the score is 0 (to discourage repeating losing moves); otherwise, it is 1. (4) For steps from draw trajectories, if the model predicts the same action, the score is 0.2; otherwise, it is 0.5, encouraging exploration beyond draw-inducing moves.

This scoring scheme encourages the model to replicate winning strategies, avoid losing ones, and explore alternatives to drawn outcomes. The overall `MIPRO` scoring standard is shown in Figure 10. In practice, we set the number of proposed prompts to 6, the number of few-shot examples to 3, and the number of trials to 10. If the optimal configuration includes few-shot examples, these are appended to the final proposed system prompt to form the new system prompt.

Table 7: Performance of the `MIPRO` method across three independent trials using GPT-4o-mini and Qwen2.5-7B-Instruct. Results are reported as mean win rates with corresponding standard deviations.

MIPRO	**Negotiation**		**Imperfect Info**		**Perfect Info**
	SimpleNegotiation	**TwoDollar**	**KuhnPoker**	**Briscola**	**Simpletak**
GPT-4o-mini					
Trial 1	38.7%	53.3%	50.7%	23.3%	16.0%
Trial 2	38.0%	52.7%	60.0%	32.7%	20.0%
Trial 3	38.7%	46.7%	54.7%	3.33%	21.3%
Avg.	38.4%	50.9%	55.1%	19.7%	19.1%
Std.	0.38	3.67	4.68	14.99	2.78
Qwen2.5-7B-Instruct					
Trial 1	43.3%	40.7%	54.0%	2.0%	18.7%
Trial 2	37.3%	52.0%	50.0%	2.0%	19.3%
Trial 3	46.7%	50.0%	57.3%	2.7%	24.7%
Avg.	42.4%	47.5%	53.8%	2.2%	20.9%
Std.	4.73	6.05	3.67	0.38	3.29

MIPRO scoring standard

Invalid Action: score = 0.0

Win Trajectory: Action match: score = 1.0 / Action mismatch: score = 0.0

Lose Trajectory: Action match: score = 0.0 / Action mismatch: score = 1.0

Draw Trajectory: Action match: score = 0.2 / Action mismatch: score = 0.5

Figure 10: MIPRO scoring standard

E.3 GEPA

GEPA (Agrawal et al., 2025) builds upon the high-level idea of `MIPRO`, but extends it by incorporating both evaluation scores and explicit feedback from the evaluation metric to guide prompt optimization. The process can be summarized as follows: (1) Initial evaluation: Run a set of examples through the evaluation metric to obtain an initial score and feedback. (2) Prompt proposal: Generate a new prompt based on the current prompt and the feedback collected. (3) Testing and retention: Evaluate the new prompt on a mini-batch. If its score surpasses the initial score, retain it in the candidate pool. (4) Candidate selection: In the next round, apply a Pareto-based filtering strategy to identify the set of candidate prompts that dominate on the validation set. Select one of these

Pareto-optimal prompts for further iteration. (5) Stopping condition: The optimization continues until the maximum number of evaluation metric calls reaches a predefined limit.

In our experiments, we set the maximum number of evaluation metric calls to 100 for each prompt optimization in `GEPA`. For win and lose trajectories, we adopt the same evaluation metric as `MIPRO`. For draw trajectories, we assign a score of 0 when the predicted action matches the trajectory action, and a score of 1 otherwise. In addition, we incorporate feedback signals in `GEPA` evaluation metric. The structured feedback template shown in Figure 11 is used during `GEPA` evaluation.

Table 8: Performance of the `GEPA` method across three independent trials using GPT-4o-mini and Qwen2.5-7B-Instruct. Results are reported as mean win rates with corresponding standard deviations.

GEPA	**Negotiation**		**Imperfect Info**		**Perfect Info**
	SimpleNegotiation	**TwoDollar**	**KuhnPoker**	**Briscola**	**Simpletak**
GPT-4o-mini					
Trial 1	34.7%	32.7%	54.7%	1.3%	23.3%
Trial 2	38.0%	43.3%	50.7%	3.3%	29.3%
Trial 3	38.0%	45.3%	51.3%	5.3%	28.0%
Avg.	36.8%	40.4%	52.2%	3.3%	26.9%
Std.	1.92	6.81	2.14	2.00	3.15
Qwen2.5-7B-Instruct					
Trial 1	29.3%	22.7%	56.0%	4.0%	20.0%
Trial 2	38.7%	30.0%	54.0%	2.0%	12.0%
Trial 3	35.3%	42.7%	57.3%	2.0%	26.0%
Avg.	34.4%	31.7%	55.8%	3.3%	19.3%
Std.	4.73	10.12	1.68	1.55	7.02

E.4 UNSTABLEBASELINE

Table 9: Performance of the `UnstableBaseline` method across three independent trials using Qwen2.5-7B-Instruct. Results are reported as mean win rates with corresponding standard deviations, where each mean win rate was from the average of 3 rounds of 50 matches with each opponent, with alternating starting positions.

UnstableBaseline	**Negotiation**		**Imperfect Info**		**Perfect Info**
	SimpleNegotiation	**TwoDollar**	**KuhnPoker**	**Briscola**	**Simpletak**
Qwen2.5-7B-Instruct					
Gemini-2.5-Flash-Lite	54.7%	43.3%	60.0%	88.6%	90.0%
Grok-4-Fast-Non-Reasoning	44.7%	22.0%	58.6%	33.3%	20.0%
Qwen3-235B-A22B-Instruct-2507	24.0%	26.0%	56.7%	38.0%	32.0%
Avg.	41.1%	30.4%	58.4%	53.3%	47.3%
Std.	15.6	11.3	1.67	30.7	37.4

UnstableBaseline (Guertler et al., 2025b) is an asynchronous online multi-agent reinforcement learning library that uses Low-Rank Adapters (LoRA) for model training. Unlike its peers such as `Verifiers` (William Brown, 2025) and `SPIRAL` (Liu et al., 2025), `UnstableBaseline` is designed to be lightweight and closely integrated with the `TextArena` (Guertler et al., 2025a) environment, in the same spirit that the baseline (Dhariwal et al., 2017) library complements OpenAI Gym (Brockman et al., 2016).

For our experiments, we trained Qwen2.5-7B-Instruct with LoRA adapters applied to the attention and feedforward projections. We used a rank of $r = 16$ and $\alpha = 32$ and dropout $= 0.0$. Using the REINFORCE algorithm (Williams, 1992), we found that the best-performing checkpoints were obtained between 100 and 150 steps, where each step consisted of 384 game trajectories.

GEPA scoring standard

# invalid action
score = 0.0
feedback = “Your predicted action is invalid. Please ensure that your action is a valid move in the game. Here is the reasoning process {{model_raw_output}}. Think about how you could have reasoned to choose a valid action that leads to a WIN."

# Win Trajectory
# Action match
score = 1.0
feedback = “You correctly predicted the action {{pred_action}} that led to a WIN. This action was indeed the one taken in the winning trajectory. Great job!"

# Action mismatch
score = 0.0
feedback = “You predicted the action {{pred_action}}, but the action taken in the winning trajectory was {{traj_action}}. This mismatch means you did not predict the winning action correctly. Here is the reasoning process {{pred_raw_action}}. Think about how you could have reasoned to get the correct action."

# Lose Trajectory
# Action match
score = 0.0
feedback = “You correctly predicted the action {{pred_action}} that led to a LOSE. However, this action was part of a losing trajectory. While your prediction matches the trajectory, it did not lead to a win. Here is the reasoning process {{pred_raw_action}}. Think about how you could have reasoned to choose an action that leads to a WIN."

# Action mismatch
score = 1.0
feedback = “You predicted the action {{pred_action}}, but the action taken in the losing trajectory was {{traj_action}}. This mismatch means you did not predict the losing action correctly. Here is the reasoning process {{pred_raw_action}}. Think about how you could have reasoned to choose an action that leads to a WIN."

# Draw Trajectory
# Action match
score = 0.0
feedback = “You predicted the action {{pred_action}}, which matches the action taken in the TIE trajectory. However, since the trajectory resulted in a TIE, this does not help in achieving a WIN. Here is the reasoning process {{pred_raw_action}}. Think about how you could have reasoned to choose an action that leads to a WIN."

# Action mismatch
score = 1.0
feedback = “You predicted the action {{pred_action}}, but the action taken in the TIE trajectory was {{traj_action}}. This mismatch means you did not predict the TIE action correctly. Here is the reasoning process {{pred_raw_action}}. Think about how you could have reasoned to choose an action that leads to a WIN."

Figure 11: GEPA scoring standard

From the best performing checkpoints, we held 3 rounds of 50 games against each of our evaluation models that is similarly used in our training settings for the other prompt evolution experiments. Their results can be found in table 9.

E.5 SPIRAL

Liu et al. (2025) is a framework that enables language models to autonomously develop reasoning capabilities through self-play in multi-turn, zero-sum games. For our experiments, we train Qwen2.5-7B-Instruct using Dr. GRPO, following the default rollout size in the provided example—each rollout comprising 128 games over 400 total steps. We then select the best-performing checkpoint and evaluate it over three rounds of 50 games each.

F Comparison with Existing Prompt Optimization Methods

In Section E, we introduced three baseline prompt optimization methods. Here, we further highlight how our approach differs from these methods.

As shown in Figure 2, our method evolves a population of prompts using elitism, local edits/expansions, random exploration, and crossover. Random exploration enables broader search over prompt variants, while crossover leverages strategies from high-performing prompts to refine new prompt candidates.

Versus Textgrad. `Textgrad` relies on hand-crafted text losses and gradient-style backpropagation over natural language. In contrast, our method is entirely *gradient-free*: it requires no differentiable loss functions or template engineering. This avoids sensitivity to wording in loss templates and reduces dependence on diagnostic outputs, where weak language models often fail to generate meaningful diagnostic responses.

Versus MIPRO. `MIPRO` frames optimization as Bayesian search over (prompt, few-shot) pairs, requiring many trials and frequent evaluation metric calls. Its effectiveness hinges on having a well-defined evaluation metric, which is difficult to obtain in text-based games where no concise supervision signal exists. As a result, `MIPRO` consumes many tokens without achieving strong performance. Our method, by contrast, does not rely on explicit evaluation metrics. It can leverage diverse signals from self-play trajectories, achieving better performance with fewer model calls and without heavy trial scheduling.

Versus GEPA. `GEPA` extends `MIPRO`'s evaluation process by augmenting it with verbose textual feedback and repeatedly querying an evaluation oracle until its call budget is exhausted, making it heavily dependent on the quality of the evaluation metric. Its key mechanism is a Pareto-based selection strategy, which identifies promising prompts from the candidate pool based on the Pareto frontier. However, the construction of this frontier relies strongly on the evaluation scores, and when the metric is not well-defined, the selected prompts may not be optimal. In contrast, our method replaces such reliance on external feedback with *experience-guided edits* distilled directly from self-play outcomes, while maintaining diversity through crossover and randomization. This design reduces token usage, improves robustness under noisy feedback, and removes dependence on external evaluation metrics.

G Full Results

Table 10: Performance of the COPER method across three independent trials using GPT-4o-mini and Qwen2.5-7B-Instruct. Results are reported as mean win rates with corresponding standard deviations.

COPER	Negotiation		Imperfect Info		Perfect Info
	SimpleNegotiation	TwoDollar	KuhnPoker	Briscola	Simpletak
GPT-4o-mini					
Trial 1	57.3%	46.0%	54.0%	54.0%	45.3%
Trial 2	55.3%	62.7%	57.3%	38.0%	40.7%
Trial 3	52.0%	48.7%	55.3%	36.0%	39.3%
Avg.	54.9%	52.4%	55.6%	42.7%	41.8%
Std.	2.69	8.95	1.68	9.87	3.15
Qwen2.5-7B-Instruct					
Trial 1	48.0%	53.3%	60.7%	39.3%	37.3%
Trial 2	47.3%	54.0%	59.3%	26.7%	32.0%
Trial 3	48.7%	38.0%	60.0%	27.3%	32.7%
Avg.	48.0%	48.4%	60.0%	31.1%	34.0%
Std.	0.67	9.05	0.67	7.13	2.90

H GAME ENVIRONMENTS

These are the more detailed descriptions of the games we selected the following set of text-based games from `TextArena` (Guertler et al., 2025a) and `SPIN-Bench` (Yao et al., 2025).

Simple Negotiation (Nash, 1950) requires players to reason about trade-offs through the exchange of resources such as wood, wheat, sheep, brick, and ore. Each player aims to maximize the value of their inventory by making offers and counteroffers with their opponent. Success depends on the each player's ability to infer the opponent's valuation of resources and strategically increase their own portfolio without making disadvantageous trades.

Two Dollar Game (Rowe, 2001) is a classroom negotiation game where two players have to agree on how to divide a fixed sum of $2.00. Typically, players each receive private role instructions that impose certain constraints or encourage specific negotiation styles. This asymmetric information requires players to balance their objectives with compromises while inferring the opponent's position.

Kuhn Poker (Kuhn, 1951) is a simplified form of poker played with three cards (Jack, Queen, and King). Two players each receive one card, while the third remains unseen. A single round of betting follows, where players can check, bet, call, or fold. If neither folds, the winner is determined by the higher card.

Briscola (McLeod, 2023) is a traditional Italian trick-taking card game played with a 40-card deck. At the start, a single card is revealed to determine the trump suit, and each player is dealt a hand of cards. Players take turns playing one card per trick, with the highest card of the leading suit or the highest trump winning the round. The objective is to accumulate points by capturing valuable cards, requiring players to balance tactical play with long-term strategy and inference of the opponent's hand.

Simple Tak (Rothfuss, 2011) is a two-player connection game inspired by the traditional game Tak. Players place tiles on a square grid with the objective of forming a continuous path that connects opposite sides of the board. Unlike full Tak, stacking pieces is not allowed, though players may block their opponent's path by occupying critical spaces. The game emphasizes spatial reasoning, foresight, and the balance between advancing one's own path and disrupting the opponent's progress.