Code-Space Response Oracles: Generating Interpretable Multi-Agent Policies with Large Language Models

Extended Abstract

Daniel Hennes*
Google DeepMind
Zurich, Switzerland
hennes@google.com

Zun Li*
Google DeepMind
New York City, USA
lizun@google.com

John Schultz
Google DeepMind
New York City, USA
jhtschultz@google.com

Marc Lanctot
Google DeepMind
Montreal, Canada
lanctot@google.com

ABSTRACT

Policy-Space Response Oracles (PSRO) have enabled the computation of approximate Nash equilibria in complex games. However, standard implementations rely on Deep Reinforcement Learning oracles, producing "black-box" neural network policies that are opaque, difficult to verify, and sample-inefficient. We introduce Code-Space Response Oracles (CSRO), a framework that tasks a Large Language Model (LLM) to synthesize code policies. CSRO reframes best-response computation as a code generation task, producing policies as executable, human-readable Python code. We demonstrate that CSRO, particularly when augmented with evolutionary refinement (AlphaEvolve), achieves performance competitive with baselines while offering superior interpretability and leveraging the LLM's pretraining knowledge.

KEYWORDS

Multi-Agent Reinforcement Learning; Game Theory; Large Language Models; Code Synthesis

ACM Reference Format:
Daniel Hennes, Zun Li, John Schultz, and Marc Lanctot. 2026. Code-Space Response Oracles: Generating Interpretable Multi-Agent Policies with Large Language Models: Extended Abstract. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 3 pages. https://doi.org/10.65109/IKJF6607

1 INTRODUCTION

Computing exact equilibria in large-scale multi-agent systems is often intractable, motivating the development of iterative approximation methods for complex domains. Policy-Space Response Oracles (PSRO) [2, 8] is a popular approach for approximating Nash equilibria in large-scale zero-sum games. PSRO builds an empirical game by iteratively expanding a population of policies through best-response computations against a meta-strategy. However, standard implementations rely on Deep Reinforcement Learning oracles, such as PPO [11] or IMPALA [5], to approximate these best responses. While powerful, deep RL oracles produce "black-box" neural network policies that are opaque, difficult to debug, and hard to verify. This lack of interpretability presents a significant barrier to deployment in high-stakes applications. Furthermore, training deep RL oracles is often sample-inefficient, requiring millions of simulation steps to converge to a competent counter-strategy.

We introduce Code-Space Response Oracles (CSRO), a framework that reformulates the best-response computation in PSRO as a program synthesis task. By replacing the deep RL oracle with an LLM that generates executable Python source code, CSRO produces code policies that are inherently interpretable and verifiable. We leverage the pre-trained reasoning capabilities of LLMs to synthesize sophisticated algorithmic strategies—such as pattern matching and opponent modeling—without the extensive interaction data required by gradient-based methods. While recent work such as LLM-PSRO [1] has demonstrated the viability of code-generation oracles using best-of-N sampling, we generalize this approach and introduce extensions for scalability and robustness: iterative refinement loops and context abstraction to handle complex opponent strategies within limited context windows. To optimize policies within each iteration, we explore a linear refinement loop for immediate error correction, and *AlphaEvolve* [9, 10], a distributed evolutionary search utilizing the LLM to mutate candidate programs. We validate CSRO by benchmarking against standardized external populations and game-theoretic solvers, showing that code-generation oracles can compete with traditional baselines in established domains.

2 METHODOLOGY

We consider a two-player symmetric zero-sum game where players share a strategy space Π and utility function u. In this setting PSRO maintains a single set of policies P, and at each iteration solves for a symmetric equilibrium mixture σ in an empirical meta-game. A best response oracle computes $\pi^* \in \arg\max_\pi \mathbb{E}_{\pi' \sim \sigma}[u(\pi, \pi')]$.

In CSRO, the policy space Π is restricted to the set of valid, executable programs in a high-level language (Python). The oracle is realized by an LLM, conditioned on a prompt containing the game

*Equal contribution.

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/IKJF6607

Table 1: Performance in Repeated Rock-Paper-Scissors. Results are averaged over 5 seeds. (*Results reported in [7].)

Method / Agent	PopReturn ↑	PopExpl ↓	AggScore ↑
CSRO			
- AlphaEvolve	50.5 ± 1.9	25.2 ± 20.3	25.4 ± 21.6
- LinearRef. (code)	159.8 ± 7.7	37.7 ± 10.6	122.1 ± 9.8
- LinearRef. (desc.)	99.3 ± 29.9	31.6 ± 12.4	67.7 ± 21.4
- ZeroShot (desc.)	130.2 ± 15.4	66.7 ± 25.9	63.5 ± 11.4
LLM Agent (27B)	193.2	67.2	126.0
PSRO-IMPALA	−108.9 ± 17.6	423.2 ± 28.0	−532.1 ± 41.5
QL (R=10)*	−0.5	8.6	−9.1
ContRM*	164.8	16.3	148.5
LLM Agent (70B)*	201.0	45.8	155.2

Table 2: Performance in Repeated Leduc Hold'em. Results are averaged over 3 seeds.

Method	PopReturn ↑	PopExpl ↓	AggScore ↑
CSRO			
- AlphaEvolve	49.3 ± 3.7	4.4 ± 0.6	44.9 ± 4.1
- LinearRefinement	43.8 ± 2.5	9.8 ± 3.0	34.0 ± 4.3
- ZeroShot	40.4 ± 1.6	19.6 ± 2.1	20.7 ± 3.0
PSRO-IMPALA	13.3 ± 6.9	58.4 ± 3.3	−45.0 ± 10.1
CFR+	39.8 ± 0.3	0.0 ± 0.0	39.8 ± 0.3

rules, API specifications, and optionally a description of the opponents present in the current meta-strategy σ. To address the context window limitations of LLMs, we implement context abstractions: Rather than injecting the raw source code of all opponent policies, we summarize the opponent behaviors using natural language descriptions or filter the context to include only the top-k policies with the highest support in σ. The generated policy π' is a stateful program, capable of maintaining internal memory (e.g., tracking opponent frequency) across stages of the repeated game.

Beyond *ZeroShot* prompting, we use an inner feedback loop designed to ensure that any new policy π' is a robust best response to the current meta-strategy. In *LinearRefinement*, an initial candidate policy is evaluated against σ, and if the expected utility $u < 0$, the LLM is prompted to refine the policy. This process continues until a positive utility is achieved or a predefined budget is exhausted. In *AlphaEvolve*, a population-based evolutionary algorithm, the LLM functions as a mutation operator within a distributed system. The population of code policies evolves through selection based on the score function $F(\pi) = \mathbb{E}_{\pi' \sim \sigma}[u(\pi, \pi')]$. *AlphaEvolve* maintains diverse subpopulations and continuously samples past high-performing programs to prompt the LLM for modifications, effectively searching the combinatorial space of algorithms to discover novel strategic mechanics.

3 EXPERIMENTS

We evaluate CSRO on two analytical benchmarks: Repeated Rock-Paper-Scissors (RRPS) and Repeated Leduc Hold'em [12]. These environments test the ability to model non-stationary opponents and manage imperfect information. We compare CSRO against three primary baselines: (1) PSRO with an IMPALA oracle [5]; (2) Counterfactual Regret Minimization + (CFR+) [13], a regret minimization solver; and (3) LLM agents: Gemma 3 27B [14] and Chinchilla 70B [6] that generate actions via prompting [7].

Performance is measured against a fixed evaluation set of heuristic bots: in RRPS we evaluate against 43 strategies from the international RRPS competitions [3, 4]; in Repeated Leduc Hold'em we evaluate against two heuristic strategies (*AlwaysCall*, *AlwaysFold*) and the single-hand approximate Nash equilibrium calculated by CFR+. We report performance in terms of the three metrics introduced in Lanctot et al. [7]: PopReturn measures the generalization capability against unseen opponents at test-time, PopExpl serves as an approximate exploitability metric, and AggScore balances generalization and robustness.

In RRPS, CSRO with *LinearRefinement* achieved an AggScore of 122.1 ± 9.8, significantly outperforming PSRO-IMPALA (−532.1 ± 41.5) and matching the Gemma 3 27B agent (126.0). Qualitative analysis of the generated code reveals that the oracle synthesized complex "Expert Ensembles" utilizing high-order Markov chains and heuristic predictors to exploit the evaluation population. The standard RL baseline failed to generalize, whereas the code-based policies implemented robust logic that generalized better to unseen heuristics.

In Repeated Leduc Hold'em, the *AlphaEvolve* variant of CSRO demonstrated superior performance. It achieved a PopReturn of 49.3 ± 3.7 and a PopExpl of 4.4 ± 0.6, effectively matching the exploitability of the CFR+ solver (0.0) while achieving a higher return against the heuristic evaluation pool. Notably, against a specific exploitability probe (*AlwaysCall*), CSRO-AlphaEvolve achieved a return of 110.3, nearly double that of PSRO-IMPALA (57.7). Inspection of the generated code shows the synthesis of explicit equity calculation functions and dynamic "bravery" parameters that adapt betting aggression based on observed opponent passivity. This confirms that the LLM oracle can discover and implement advanced concepts like value betting and semi-bluffing in executable code, without the massive sample complexity required to train a neural network to approximate these functions latently.

4 CONCLUSION

We introduced CSRO, a framework for finding approximate equilibria in multi-agent games. By replacing deep RL oracles with LLM-driven program synthesis, we enable the generation of human-readable policies that compete with state-of-the-art baselines. Results show that CSRO achieves competitive performance in terms of convergence to low-exploitability equilibria, particularly in complex games like multi-hand Leduc Hold'em. Although trailing the 70B baseline, *LinearRefinement* synthesizes reusable policies, ensuring LLM costs scale only with training iterations rather than accumulating at every environment step. While pre-training data likely includes generic RPS strategies, synthesizing best responses to dynamic, in-context opponent mixtures demonstrates sophisticated strategic reasoning beyond mere retrieval. CSRO performance depends on the base LLM capability and prompt quality; the scalability of CSRO to games with high-dimensional observation spaces remains an open question.

REFERENCES

[1] Yoram Bachrach, Edan Toledo, Karen Hambardzumyan, Despoina Magka, Martin Josifoski, Minqi Jiang, Jakob Foerster, Roberta Raileanu, Tatiana Shavrina, Nicola Cancedda, Avraham Ruderman, Katie Millican, Andrei Lupu, and Rishi Hazra. 2025. Combining Code Generating Large Language Models and Self-Play to Iteratively Refine Strategies in Games. In *Thirty-Fourth International Joint Conference on Artificial Intelligence - Demo Track*. 10999–11003.

[2] Ariyan Bighashdel, Yongzhao Wang, Stephen McAleer, Rahul Savani, and Frans A. Oliehoek. 2024. Policy Space Response Oracles: A Survey. In *Thirty-Third International Joint Conference on Artificial Intelligence - Survey Track*. 7951–7961.

[3] Darse Billings. 2000. The First International RoShamBo Programming Competition. *ICGA Journal* 23, 1 (2000), 42–50.

[4] Darse Billings. 2000. The Second International RoShamBo Programming Competition. https://icga.org/icga/games/roshambo/.

[5] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. 2018. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. In *Thirty-Fifth International Conference on Machine Learning*, Vol. 80. 1407–1416.

[6] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. 2022. Training Compute-Optimal Large Language Models. In *Thirty-Sixth International Conference on Neural Information Processing Systems*. 15.

[7] Marc Lanctot, John Schultz, Neil Burch, Max Olan Smith, Daniel Hennes, Thomas Anthony, and Julien Perolat. 2023. Population-based Evaluation in Repeated Rock-Paper-Scissors as a Benchmark for Multiagent Reinforcement Learning. *Transactions on Machine Learning Research* (2023), 25.

[8] Marc Lanctot, Vinicius Zambaldi, Audrūnas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. 2017. A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning. In *Thirty-First International Conference on Neural Information Processing Systems*. 4193–4206.

[9] Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. 2025. AlphaEvolve: A coding agent for scientific and algorithmic discovery. arXiv:2506.13131 [cs.AI] https://arxiv.org/abs/2506.13131

[10] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. 2024. Mathematical discoveries from program search with large language models. *Nature* 625, 7995 (2024), 468–475.

[11] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG]

[12] Finnegan Southey, Michael Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and Chris Rayner. 2005. Bayes' Bluff: Opponent Modelling in Poker. In *Twenty-First Conference on Uncertainty in Artificial Intelligence*. 550–557.

[13] Oskari Tammelin. 2014. Solving Large Imperfect Information Games Using CFR+. arXiv:1407.5042 [cs.GT] https://arxiv.org/abs/1407.5042

[14] Gemma Team. 2025. Gemma 3 Technical Report. arXiv:2503.19786 [cs.CL]