Transformer Guided Coevolution: Improved Team Formation in Multiagent Adversarial Games

Extended Abstract

Pranav Rajbhandari
Carnegie Mellon University
Pittsburgh, PA, United States
prajbhan@alumni.cmu.edu

Prithviraj Dasgupta
Naval Research Laboratory
Washington, D.C., United States
prithviraj.dasgupta.civ@us.navy.mil

Donald Sofge
Naval Research Laboratory
Washington, D.C., United States
donald.a.sofge.civ@us.navy.mil

ABSTRACT

With the increasing number of autonomous platforms in everyday life, forming coordinated teams of agents becomes vital. To solve this, we propose BERTeam, an algorithm inspired by Natural Language Processing. BERTeam trains a transformer-based deep neural network to select from a population of agents. It can integrate with coevolutionary deep reinforcement learning, which evolves a diverse set of players to choose from. We evaluate BERTeam in Marine Capture-The-Flag, and find it learns non-trivial team compositions that outperform unknown opponents. In this setting, we find that BERTeam outperforms MCAA, another team selection algorithm.

KEYWORDS

Multiagent games; Team selection; Sequence generation

ACM Reference Format:
Pranav Rajbhandari, Prithviraj Dasgupta, and Donald Sofge. 2025. Transformer Guided Coevolution: Improved Team Formation in Multiagent Adversarial Games: Extended Abstract. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 3 pages.

1 INTRODUCTION

We inspect multiagent adversarial team games, characterized by an environment with multiple teams of agents, each working to achieve a team goal. Each episode results in an outcome, a set of teams that 'win'. Examples of scenarios that can be formulated in this way include pursuit-evasion [5, 13, 28, 33], robotic football [12, 18, 29], and robotic capture-the-flag [23], as well as real-world applications like search-and-rescue and autonomous surveillance.

A crucial problem in this setting is selecting coordinated teams from a set of potential members to outperform unknown opponents. This is challenging since optimal team selection must consider both intra-team and inter-team interactions. Researchers have addressed this with various approaches, such as finding Nash Equilibria with double oracle methods [21] and predicting game outcomes [33].

Additionally, agents often must learn individual policies, increasing the problem's complexity. For this task, evolutionary algorithms have been used for decades due to their adaptability and performance. Self-play [15, 16], used in adversarial settings, is a central concept in these algorithms, creating training data against a variety of opponents by using current and past versions of agents as opponents. Coevolutionary deep reinforcement learning [7, 19], which is used in multiagent settings, blends an evolutionary approach with Reinforcement Learning (RL) to optimize a population of agents.

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

2 TEAM SELECTION IN MULTIAGENT GAMES

Preliminaries: A Markov Decision Process (MDP) is a framework capturing various optimization tasks, described by state space S, action space A, transition function $\mathcal{T}(S \mid S \times A)$, reward function $\mathcal{R}\colon S{\times}A{\times}S \to \mathbb{R}$, and discount factor $\gamma \in [0, 1)$. An MDP agent is described by its policy $\pi(A \mid S)$. The sequence $(s_0, a_0, r_0), (s_1, a_1, r_1), \ldots$ is referred to as a trajectory, where $a_i \sim \pi(\cdot|s_i)$, $s_i \sim \mathcal{T}(\cdot|s_{i-1}, a_{i-1})$, and $r_t = \mathcal{R}(s_t, a_t, s_{t+1})$. An agent's objective is choosing π to maximize the expected *return*: $\mathbb{E}[\sum_i \gamma^i r_i]$ over trajectories. RL methods solve MDPs by sampling trajectories to optimize a policy's expected return. Deep RL methods use deep neural networks to do this.

We are concerned with kvk adversarial games, which can be formalized as an MDP for each agent, with the actions of all other agents considered in $\mathcal{T}$. Agents are partitioned into teams, and an outcome evaluation decides which teams 'won' a specific trajectory. We expect an agent's MDP rewards to correlate with its team's outcome. This ensures agents with high rewards are likely to be on winning teams. Within this framework, our goal is to train and select teams of agents that are strong against a variety of opponents.

BERTeam Algorithm: We first consider team selection from a *fixed* set of agents. We view this as sequence generation by assigning each agent a token, and equating a size k team to a length k token sequence. Thus, we can use a transformer [32], a sequence-to-sequence deep neural network widely used in Natural Language Processing (NLP) to create a context-dependent embedding of a sequence [3, 4, 17, 20]. Transformers are encoder-decoder models that take an input and target sequence, and return an embedded vector for each element of the target [1, 6, 30]. A final layer converts each embedding into a probability distribution over tokens, allowing sequence generation by repeated prediction. BERT [8] is a technique that trains a transformer with Masked Language Modeling (MLM) [31]. This teaches the model to predict masked tokens with bidirectional context, improving robustness [8, 11].

We design BERTeam by adapting BERT's algorithm for team selection. BERTeam takes in a partially masked sequence of agents (and any observations), and predicts agents to fill masked positions. We train BERTeam to imitate a distribution of 'good' teams, automatically generated from games between teams sampled from BERTeam. We include winning teams in a replay buffer dataset with inverse probability weighting [14] to ensure a team's contribution is proportional to its success rate against BERTeam's distribution.

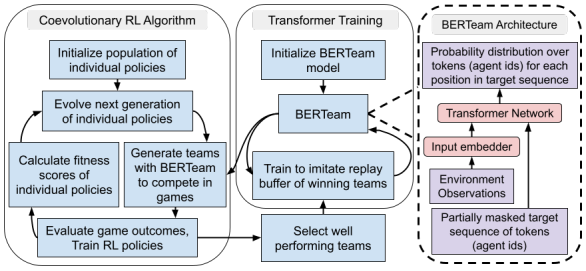


Figure 1: Training of BERTeam alongside coevolutionary RL

To learn individual agent policies alongside BERTeam, we use a coevolutionary RL algorithm, trained with sample games used in BERTeam's training (Figure 1). We use the algorithm in [7], with some caveats: To derive individual fitness from team outcomes, we assign each team a *captain*, and assume the team is selected conditional to the captain's inclusion. We consider each game as between the captains and use Elo [10] for individual fitness. We also stochastically replace only a few agents per generation so that most information BERTeam learns stays relevant between generations.

3 EXPERIMENTS

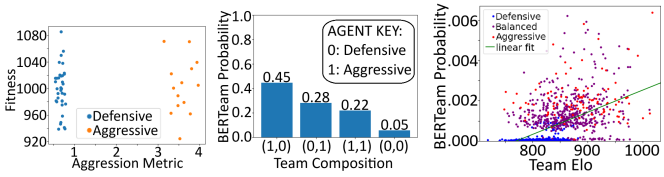
We empirically validate BERTeam with Aquaticus, a *kvk* Marine Capture-The-Flag (MCTF) game [23]. In particular, we test on 2v2 games in Pyquaticus, a simulation of Aquaticus [2]. The team size allows easier analysis of the total distributions learned by BERTeam.[1]

Fixed Policy Agents: We test BERTeam with seven fixed policy agents included in Pyquaticus: Weak/medium/strong offensive agents (agents 0, 1, 2) and defensive agents (agents 3, 4, 5), and one random agent (agent 6). We sample games to estimate *team* Elos[2] of all teams. We compare ranks with the predicted rank from occurrence in BERTeam's total distribution (Table 1). After 15,000 games, BERTeam correctly learns that the balanced team $\{2, 5\}$ performs the best, followed by the offensive $\{2, 2\}$. It correctly predicts the top 7 teams, though with mixed ranks. Overall, this indicates BERTeam can learn non-trivial, diverse, and balanced team compositions.

Team	True Rank/Elo		BERTeam Rank/Occurrence	
$\{2, 5\}$	*1*	1388	*1*	0.14
$\{2, 2\}$	*2*	1337	*2*	0.13
$\{2, 3\}$	*3*	1135	*7*	0.06
$\{1, 2\}$	*4*	1112	*4*	0.10
$\{0, 2\}$	*5*	1097	*3*	0.10
$\{2, 4\}$	*6*	1087	*5*	0.10
$\{2, 6\}$	*7*	1035	*6*	0.07
$\{0, 5\}$	*8*	975	*13*	0.03

Table 1: Comparison of true ranks and predicted ranks

Coevolving Agents: We test BERTeam trained alongside a Coevolutionary RL algorithm (Figure 1). Our population is 50 PPO agents [27], and we sample 25 games per generation. To classify agents as defensive/aggressive, we define an aggression metric based on an agent's behavior in test games against fixed-policy teams. After 8000 generations, we find a defensive and aggressive cluster in the population (Figure 2(a)). With this labeling, BERTeam's learned total distribution favors a balanced team, choosing {defensive, aggressive} 73% of the time (Figure 2(b)). We calculate Elos of all possible teams by sampling against fixed policy teams, and find BERTeam's output probability correlates with the Elo of each team (Figure 2(c)). Finally, the best performing team has Elo $\approx$ 1017, is the maximizer of BERTeam's total distribution, and is stronger than all fixed-policy teams not containing agent 2.



(a) Agent clustering based on aggression, with 36 defensive and 14 aggressive

(b) BERTeam team composition (ordered)

(c) Correlation of BERTeam total distribution with Elo. Linear regression yields $R^2 \approx .25$.

Figure 2: BERTeam learned distribution on trained agents

Comparison with MCAA: We compare our method with MCAA [9], another team selection algorithm. MCAA partitions agents into islands, uses the evolutionary MAP-Elites algorithm on each island to train agents [22], then learns the proportion each island should contribute to a team. Since MCAA decouples MAP-Elites policy optimization from learning team selection, we hybridize this with BERTeam and coevolution to compare four approaches. We train each for 4000 generations, sample teams using each respective team selection method, and use the sampled game outcomes to estimate performance of a team generated from each algorithm (the *algorithm* Elo: $\mathbb{A}$[Elo]). We find that regardless of the policy optimization method, BERTeam outperforms MCAA in team selection. However, BERTeam is much more computationally costly than MCAA, as each MCAA update takes insignificant time. This cost can be justified by the increased performance, and by the fact that BERTeam can train independent to the policy optimization.

Policy Optimizer	Team Selection	$\mathbb{A}$[Elo]	Avg. update time of	
			Agents	Team Dist.
Coevolution	BERTeam	919	13 s/epoch	46 s/update
Coevolution	MCAA	817	13 s/epoch	$\approx$ 0 s/update
MAP-Elites	BERTeam	883	36 s/epoch	45 s/update
MAP-Elites	MCAA	809	35 s/epoch	$\approx$ 0 s/update

Table 2: Relative performance of hybrid algorithms

4 CONCLUSION

In this paper, we introduce BERTeam, an algorithm for team selection in multiagent adversarial team games. We evaluate BERTeam in Pyquaticus, a simulated MCTF game, and find that it effectively learns strong non-trivial team compositions both in choosing from fixed policy agents as well as trained alongside individual agent policies. We find that in this setting, BERTeam outperforms MCAA, another algorithm designed for team selection. Overall, BERTeam is a strong team selection algorithm with roots in NLP methods.

[1] Our implementation, along with experiments, is available at [24–26].
[2] Distinguished from agent fitnesses, which are individual Elos.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. Neural Machine Translation by Jointly Learning to Align and Translate. arXiv:1409.0473 [cs.CL]

[2] Jordan Beason et al. 2024. Evaluating Collaborative Autonomy in Opposed Environments using Maritime Capture-the-Flag Competitions. arXiv:2404.17038 [cs.RO]

[3] Tom Brown et al. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS '20)*. Curran Associates Inc., Article 159, 25 pages.

[4] Wei-Cheng Chang et al. 2020. Taming Pretrained Transformers for Extreme Multi-label Text Classification. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*. Association for Computing Machinery, 3163–3171.

[5] Mo Chen, Zhengyuan Zhou, and Claire J. Tomlin. 2017. Multiplayer Reach-Avoid Games via Pairwise Outcomes. *IEEE Trans. Automat. Control* 62, 3 (2017), 1451–1457.

[6] Kyunghyun Cho et al. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 1724–1734.

[7] David Cotton, Jason Traish, and Zenon Chaczko. 2020. Coevolutionary Deep Reinforcement Learning. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. Institute of Electrical and Electronics Engineers, 2600–2607.

[8] Jacob Devlin et al. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*, Vol. 1. Association for Computational Linguistics, 2.

[9] Gaurav Dixit, Everardo Gonzalez, and Kagan Tumer. 2022. Diversifying behaviors for learning in asymmetric multiagent systems. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '22)*. Association for Computing Machinery, 350–358.

[10] Arpad Elo. 1978. *The Rating of Chessplayers, Past and Present*. Arco Pub.

[11] William Fedus, Ian J. Goodfellow, and Andrew M. Dai. 2018. MaskGAN: Better Text Generation via Filling in the _____. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

[12] Vanessa Frías-Martínez and Elizabeth Sklar. 2004. A team-based co-evolutionary approach to multi agent learning. In *Proceedings of the 2004 AAMAS Workshop on Learning and Evolution in Agent Based Systems*. Citeseer, Autonomous Agents and Multiagent Systems.

[13] Eloy Garcia et al. 2020. Optimal Strategies for a Class of Multi-Player Reach-Avoid Differential Games in 3D Space. *IEEE Robotics and Automation Letters* 5, 3 (2020), 4257–4264.

[14] Morris H. Hansen and William N. Hurwitz. 1943. On the Theory of Sampling from Finite Populations. *The Annals of Mathematical Statistics* 14, 4 (1943), 333–362.

[15] Johannes Heinrich and David Silver. 2016. Deep Reinforcement Learning from Self-Play in Imperfect-Information Games. arXiv:1603.01121 [cs.LG]

[16] Max Jaderberg et al. 2019. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science* 364, 6443 (2019), 859–865.

[17] Marcin Junczys-Dowmunt. 2019. Microsoft Translator at WMT 2019: Towards Large-Scale Document-Level Neural Machine Translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*. Association for Computational Linguistics, 225–233.

[18] Hiroaki Kitano et al. 1997. The RoboCup synthetic agent challenge 97. In *Proceedings of the 15th International Joint Conference on Artifical Intelligence - Volume 1 (IJCAI'97)*. Morgan Kaufmann Publishers Inc., 24–29.

[19] Daan Klijn and A. E. Eiben. 2021. A coevolutionary approach to deep multi-agent reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '21)*. Association for Computing Machinery, 283–284.

[20] Xiaodong Liu et al. 2020. Very Deep Transformers for Neural Machine Translation. arXiv:2008.07772 [cs.CL]

[21] Stephen McAleer et al. 2023. Team-PSRO for Learning Approximate TMECor in Large Team Games via Cooperative Reinforcement Learning. In *Advances in Neural Information Processing Systems*, Vol. 36. Curran Associates, Inc., 45402–45418.

[22] Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. arXiv:1504.04909 [cs.AI]

[23] Michael Novitzky et al. 2019. Aquaticus: Publicly Available Datasets from a Marine Human-Robot Teaming Testbed. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. Institute of Electrical and Electronics Engineers, 392–400.

[24] Pranav Rajbhandari. 2024. BERTeam. https://github.com/pranavraj575/BERTeam.

[25] Pranav Rajbhandari. 2024. Transformer based Coevolver. https://github.com/pranavraj575/coevolution.

[26] Pranav Rajbhandari. 2024. unstable_baselines3. https://github.com/pranavraj575/unstable_baselines3.

[27] John Schulman et al. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG]

[28] Daigo Shishika et al. 2019. Team Composition for Perimeter Defense with Patrollers and Defenders. In *2019 IEEE 58th Conference on Decision and Control (CDC)*. Institute of Electrical and Electronics Engineers, 7325–7332.

[29] Yan Song et al. 2024. Boosting Studies of Multi-Agent Reinforcement Learning on Google Research Football Environment: The Past, Present, and Future. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multi-agent Systems (AAMAS '24)*. International Foundation for Autonomous Agents and Multiagent Systems, 1772–1781.

[30] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'14)*. MIT Press, 3104–3112.

[31] Wilson Taylor. 2016. "Cloze Procedure": A New Tool For Measuring Readability. In *Journalism Quarterly*. Sage Journals, 415–433.

[32] Ashish Vaswani et al. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., 6000–6010.

[33] Yue Zhao, Lushan Ju, and Josè Hernández-Orallo. 2024. Team formation through an assessor: choosing MARL agents in pursuit-evasion games. *Complex & Intelligent Systems* 10, 3 (2024), 3473–3492.