AlphaZeroES: Direct Score Maximization Outperforms Planning Loss Minimization

Extended Abstract

Carlos Martin
Carnegie Mellon University
Pittsburgh, United States
cgmartin@cs.cmu.edu

Tuomas Sandholm
Carnegie Mellon University
Strategy Robot, Inc.
Optimized Markets, Inc.
Strategic Machine, Inc.
Pittsburgh, United States
sandholm@cs.cmu.edu

ABSTRACT

Planning at execution time has been shown to dramatically improve performance for AI agents. A well-known family of approaches to planning at execution time in single-agent settings and two-player zero-sum games are AlphaZero and its variants, which use Monte Carlo Tree Search together with a neural network that guides the search by predicting state values and action probabilities. AlphaZero trains these networks by minimizing a planning loss that makes the value prediction match the episode return, and the policy prediction at the root of the search tree match the output of the full tree expansion. AlphaZero has been applied to various single-agent environments that require careful planning, with great success. In this paper, we explore an intriguing question: can we outperform it by directly maximizing the episode score instead of minimizing this planning loss, while leaving the MCTS algorithm and neural architecture unchanged? To directly maximize the episode score, we use evolution strategies, a family of algorithms for zeroth-order blackbox optimization. Our experiments indicate that, across all the tested single-agent environments, directly maximizing the episode score instead of minimizing the planning loss yields a dramatic improvement in performance.

KEYWORDS

Reinforcement learning; planning; Monte Carlo Tree Search (MCTS)

ACM Reference Format:
Carlos Martin and Tuomas Sandholm. 2025. AlphaZeroES: Direct Score Maximization Outperforms Planning Loss Minimization: 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

Lookahead search and reasoning is a central paradigm in artificial intelligence, and has a long history [1, 3–7]. In many domains, planning at execution time significantly improves performance. In domains like Sokoban, Pacman, and 2048, all state-of-the-art approaches use some form of planning by the agent. Many planning approaches use *Monte Carlo Tree Search (MCTS)*, which iteratively grows a search tree from the current state, and does so asymmetrically according to the information seen so far. A prominent subfamily of approaches in this category are AlphaZero [9] and its variants, which leverage function approximation via neural networks to learn good heuristic predictions of the values and action distributions at each state, which can be used to guide the tree search. AlphaZero (and its variants) train this prediction function by minimizing a *planning loss* consisting of the sum of a *value loss* and a *policy loss*. In this paper, we set out to explore whether we can outperform AlphaZero and its variants in such environments by *directly maximizing the episode score* instead, while leaving all other aspects of the agent, MCTS algorithm, and neural architecture unchanged. Since MCTS is not differentiable, to maximize the episode score, we employ evolution strategies, a family of algorithms for zeroth-order black-box optimization. We call our method AlphaZeroES.

2 PROPOSED METHOD

We use Gumbel MuZero [2], a variant of AlphaZero and prior state of the art for this setting. It iteratively constructs a search tree starting from an environment state. The *simulation budget* is the total number of iterations, which is the number of times the search tree is expanded, and therefore the size of the tree. The prediction function of the agent takes an environment state as input and outputs a probability distribution over actions and value estimate. In our experiments, we use *DeepSets [12]*, a neural network architecture that can process sets of inputs in a way that is equivariant or invariant (depending on the desired type of output) with respect to the inputs. Our approach keeps *exactly the same* architecture, hyperparameters, and MCTS algorithm as AlphaZero, but changes the optimization objective. Specifically, instead of minimizing the planning loss, we *directly maximize the episode score*. The parameters that are optimized are exactly those of AlphaZero, namely, the neural network parameters of the prediction function. Only the training objective is different. One way to directly optimize the episode score is to use policy gradient methods, which yield an estimator of the gradient of the expected return with respect to the agent's parameters. These methods assume that the policy is *differentiable*—more precisely, that its output action distribution is differentiable with respect to

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).

the parameters of the policy. However, our planning policy uses MCTS as a subroutine, and standard MCTS is not differentiable. To resolve this, we turn to black-box (*i.e.*, zeroth-order) optimization, which uses only function evaluations to optimize a function with respect to a set of inputs. In our case, the black-box function maps our policy's parameters to a sampled episode score. Natural evolution strategies (NES) [10, 11] represent the population as a distribution over parameters and maximize its average objective value using the score function estimator. We use OpenAI-ES [8], an NES algorithm that is widely used for reinforcement learning.

3 EXPERIMENTS

In this section, we briefly describe our experiments. We use the following hyperparameters. We use 10 trials per experiment, 1000 episodes per batch (for both training and evaluation at the end of each epoch), 1000 training batches per epoch, 4 hours of training time per trial, the Adabelief [13] optimizer, a perturbation scale of 0.1 for OpenAI-ES, an MCTS simulation budget of 8, hidden layers of size 16, and the ReLU activation function. In our plots, we show the episode scores attained by AlphaZero (labeled `es=0` in each legend) vs. AlphaZeroES (labeled `es=1` in each legend). At any point along the X axis, AlphaZero and AlphaZeroES have undergone the same number of episodes of learning. To perform a fair comparison, since AlphaZero and AlphaZeroES optimize different objectives, we test both across a wide range of learning rates (labeled `lr` in each legend). Solid lines show the mean across trials, and bands show the standard error of the mean. Our goal is not to develop the best special-purpose solver for any one of these domains. Rather, we are interested in a *general*-purpose approach that can tackle *all* of these domains and learn good heuristics on its own. A brief description of the environments is as follows. **Navigation**: Reach as many targets as possible within a given time limit. **Sokoban**: Push boxes around to get them to storage locations. **Traveling salesman problem (TSP)**: Find the shortest tour through a set of cities. **Vertex k-center problem (VKCP)**: Find a subset of points that minimizes the maximum distance to the whole set. **Maximum diversity problem (MDP)**: Find a subset of points that maxinimizes the minimum distance between distinct points. These environments are illustrated in Figure 1. Resulting scores over the course of training are shown in Figure 2. Our method learns significantly faster.

4 CONCLUSION

In this paper, we set out to study whether AlphaZero and its newest variants can be improved by maximizing the episode score directly instead of minimizing the standard planning loss. Since MCTS is not differentiable, we maximize the episode score by using evolution strategies. We conducted experiments across multiple domains, including standard combinatorial optimization problems and motion planning problems from the literature. In each setting, our approach yielded a dramatic improvement in performance over planning loss minimization. Our method suggests that maximizing "self-consistency", as the standard AlphaZero loss does, is not necessarily aligned as an objective with performing better in the environment in terms of score. One reason might be that optimal or strong performance does not actually require *internal consistency*,

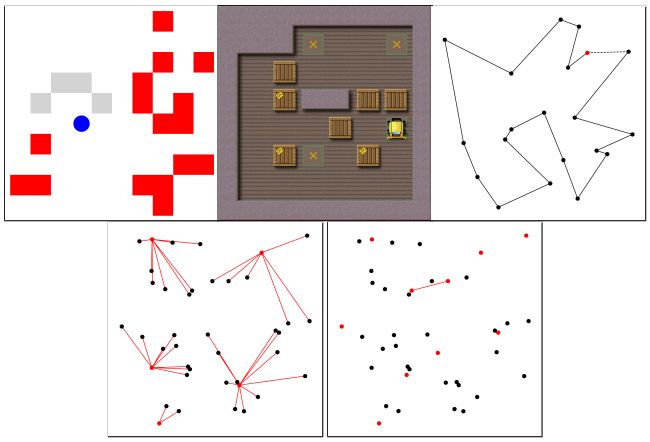

Figure 1: Example states for each environment. Top–bottom, left–right: Navigation, Sokoban, TSP, VKCP, MDP.

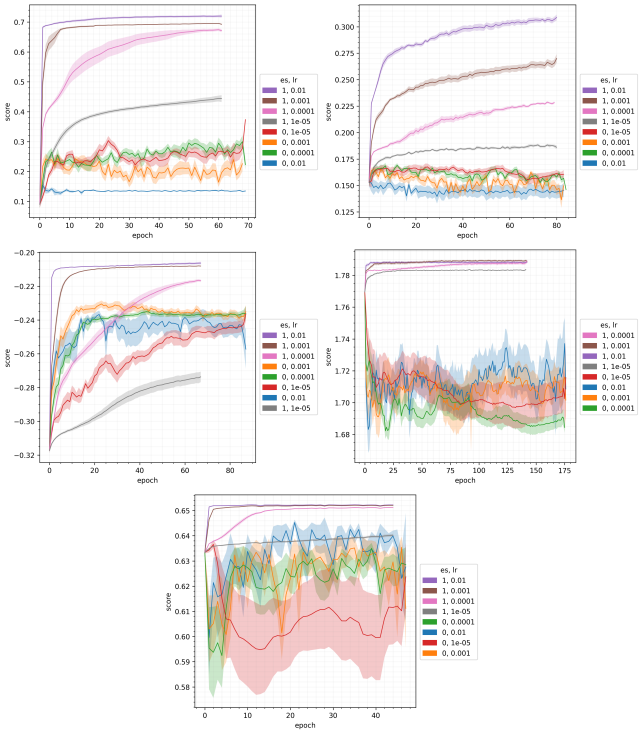

Figure 2: Scores during training. Top–bottom, left–right: Navigation, Sokoban, TSP, VKCP, MDP.

and achieving *good performance* might be easier than achieving internal consistency (of value and action predictions).

ACKNOWLEDGMENTS

This material is based on work supported by the Vannevar Bush Faculty Fellowship ONR N00014-23-1-2876, NSF grants RI-2312342 and RI-1901403, ARO award W911NF2210266, and NIH award A240108S001.

REFERENCES

[1] Noam Brown et al. 2018. Depth-Limited Solving for Imperfect-Information Games. In *NeurIPS*.

[2] Ivo Danihelka et al. 2022. Policy improvement by planning with Gumbel. In *ICLR*.

[3] Peter Hart et al. 1972. Correction to "A formal basis for the heuristic determination of minimum cost paths". *ACM SIGART Bulletin* (1972).

[4] Peter Hart, Nils Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics* (1968).

[5] Marc Lanctot et al. 2017. A unified game-theoretic approach to multiagent reinforcement learning. *NeurIPS* (2017).

[6] Allen Newell and George Ernst. 1965. The search for generality. In *Proc. IFIP Congress*.

[7] Nils Nilsson. 1971. Problem-solving methods in artificial intelligence. *Artificial Intelligence* (1971).

[8] Tim Salimans et al. 2017. Evolution strategies as a scalable alternative to reinforcement learning.

[9] David Silver et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* (2018).

[10] Daan Wierstra et al. 2014. Natural evolution strategies. *JMLR* (2014).

[11] Sun Yi et al. 2009. Stochastic search using the natural gradient. In *ICML*.

[12] Manzil Zaheer et al. 2017. Deep Sets. *NeurIPS* (2017).

[13] Juntang Zhuang et al. 2020. AdaBelief optimizer: Adapting stepsizes by the belief in observed gradients. *NeurIPS* (2020).