ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.

Integrating Potential-Based Reward Shaping into AlphaZero

Koen Boeckx and Xavier Neyt

Royal Military Academy - CISS Department
Rue Hobbema 8, Brussels - Belgium

Abstract. AlphaZero achieves superhuman performance through pure self-play without human expertise, but its dependence on sparse terminal rewards limits learning efficiency. This paper investigates integrating potential-based reward shaping into AlphaZero to accelerate learning while preserving optimality. We address whether reward shaping improves sample efficiency without compromising final performance, and which integration methods prove most effective. We present two implementation approaches: search-time shaping and auxiliary network heads, each targeting different components of the learning process. Experimental evaluation on Othello provides initial evidence of benefits, with ongoing work on comprehensive performance characterization across diverse environments.

1 Introduction

The AlphaZero algorithm [1] has demonstrated remarkable capability in mastering complex strategic games through reinforcement learning combined with Monte Carlo Tree Search (MCTS) [2]. It operates through an iterative process alternating between two distinct phases.

During the self-play phase, the current neural network policy plays games against itself, using MCTS to generate improved action distributions at each state. For every position encountered, the algorithm samples an action from the MCTS-generated policy, executes it according to game dynamics, and continues until reaching a terminal state. The terminal reward, typically encoded as $z \in \{-1, 0, 1\}$ for loss, draw, or win from the final player's perspective, is recorded. Each step produces a training sample consisting of the state, the action taken, the MCTS policy distribution, and the eventual outcome.

AlphaZero's MCTS implementation maintains visit counts $N(s, a)$, total action values $W(s, a)$, and prior probabilities $p(a|s)$ for each state-action pair encountered during search. The action-value estimate is computed as the empirical mean: $Q(s, a) = W(s, a)/N(s, a)$.

MCTS search proceeds through three steps: (1) during selection, the algorithm traverses the already explored game tree, starting at the root node, by repeatedly choosing actions according to the formula:

$$a = \arg\max_{a'} \left[Q(s, a') + c \cdot p(a'|s) \frac{\sqrt{\sum_b N(s, b)}}{1 + N(s, a')} \right] \tag{1}$$

with c a coefficient that determines the ratio between exploitation and exploration.

ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.

Selection continues until reaching an unexpanded leaf node. The expansion step (2) evaluates the leaf state s with the neural network f_θ, obtaining policy and value predictions $(p(s,\cdot), v(s)) = f_\theta(s)$. Child nodes are initialized with $Q(s,a_i) = v(s)$ and $N(s,a_i) = 1$, while the policy vector is stored for use in subsequent selections. The backup step (3) propagates the value estimate backward along the traversed path, updating statistics at each node:

$$W(s,a) \leftarrow W(s,a) + v \quad \text{and} \quad N(s,a) \leftarrow N(s,a) + 1 \tag{2}$$

The network training phase utilizes batches of (π_i, z_i), sampled from a replay buffer containing accumulated self-play experience: π_i is the policy vector of the root node, z_i the game outcome, both obtained by MCTS. As mentioned previously, the neural network f_θ maps each state s_i to a policy vector p_i over actions and a scalar value v_i estimating the current player's winning probability. The training loss combines cross-entropy loss for policy learning with mean-squared error for value prediction:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{policy}} + \mathcal{L}_{\text{value}} = \sum_i \mathcal{L}_i = \sum_i \Big[-\pi_i^T \log(p_i) + (z_i - v_i)^2 \Big] \tag{3}$$

A common way to improve this basic version of AlphaZero is the addition of league training, as introduced in [3]. Rather than training a single agent against itself, the league architecture creates an ecosystem where multiple agents with different objectives compete continuously. This structure prevents the rock-paper-scissors dynamic where an agent might discover a strategy defeating its immediate predecessor while losing capability against earlier versions.

While AlphaZero (including league training) has proven successful, it faces challenges due to sparse reward signals. The algorithm typically receives meaningful feedback only at game termination, when the outcome is determined as a win, loss, or draw. This sparsity creates difficulties for credit assignment, as the learning system must determine which decisions throughout an episode contributed to the final outcome [4]. In games with long horizons this becomes particularly challenging, requiring extensive computational resources and training time to achieve strong performance.

Reward shaping offers a potential solution to these difficulties by providing intermediate learning signals that guide exploration and stabilize value function learning. The challenge lies in incorporating such signals without introducing bias that might constrain the discovery of novel strategies or lead to suboptimal policies. In potential-based shaping [5], a scalar potential function $\Phi(s)$ over states defines a shaping term added to each transition reward r_t:

$$r'_t = r_t + F(s_t, a_t, s_{t+1}) \quad \text{with} \quad F(s_t, a_t, s_{t+1}) = \gamma\Phi(s_{t+1}) - \Phi(s_t) \tag{4}$$

with $\gamma \in [0,1]$ the discount factor, typically set to 1 in AlphaZero.

Potential-based reward shaping provides theoretical guarantees of policy invariance, ensuring that the set of optimal policies remains unchanged despite the addition of auxiliary rewards.

ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.

This paper presents a systematic investigation of potential-based reward shaping within the AlphaZero framework. We develop two distinct integration methods that target different components of the learning pipeline: (1) search-time modification of MCTS leaf values, and (2) auxiliary prediction heads that learn domain-specific potential functions.

2 Related Work

Potential-based reward shaping [5] provides theoretical guarantees for policy invariance in reinforcement learning, but its integration into self-play algorithms like AlphaZero remains underexplored. Auxiliary prediction tasks have been successfully applied in deep RL [6] to improve representation learning, though these typically predict generic features rather than domain-specific heuristics designed to preserve optimality. While variants of AlphaZero have incorporated architectural modifications and training improvements [3], systematic investigation of reward shaping methods within the MCTS-neural network framework has received limited attention in the literature.

3 Reward Shaping Integration Methods

We present two distinct approaches for incorporating potential-based reward shaping into AlphaZero, each targeting different components of the learning architecture.

3.1 Search-Time Shaping in MCTS

The first method modifies leaf node values during MCTS search. When the neural network evaluates an expanded leaf, returning value v from the leaf player's perspective, we adjust this estimate by adding a scaled potential difference:

$$v' = v + \lambda_{\text{mcts}} \left[\gamma \Phi(s_{\text{leaf}}) - \Phi(s_{\text{parent}})\right] \tag{5}$$

The modified value v' is then backed up through the tree with standard sign alternation to maintain player perspective consistency. This adjustment biases search toward promising regions of the state space as defined by the heuristic potential function. The parameter λ_{mcts} controls the influence of shaping relative to the learned value.

Computing the potential function Φ has to be efficient since it is evaluated at every leaf expansion during search. The shaping contribution affects the MCTS selection phase, causing actions leading to higher-potential states to receive more visits and stronger policy weight.

Importantly, this modification occurs only during search and does not alter the training targets. The stored samples still use the terminal outcome z as the value target.

ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.

3.2 Auxiliary Prediction Head

A second method extends the network architecture with an additional output head that predicts the potential function, similar to the method used in [6]. In addition to policy and value outputs, the network produces a scalar $\hat{\Phi}(s)$ intended to approximate the hand-crafted potential $\Phi(s)$.
The training objective incorporates an auxiliary loss term:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{policy}} + \mathcal{L}_{\text{value}} + \eta\mathcal{L}_{\text{aux}} \tag{6}$$

where $\mathcal{L}_{\text{aux}}$ is the mean-squared error between predicted and true potential values. The weight η is annealed linearly during training, starting with significant influence to establish useful feature representations and gradually decreasing to avoid interference with primary objectives.
Learning to predict Φ encourages the network's shared trunk to develop internal representations that capture strategically relevant positional features. This inductive bias can improve both value and policy learning efficiency, particularly when the potential function encodes genuine strategic insights. Furthermore, once learned, $\hat{\Phi}$ provides a differentiable alternative to hand-crafted heuristics that could be fine-tuned through gradient descent.
The auxiliary head introduces minimal computational overhead during training while potentially accelerating representation learning in the early phases when experience is limited and value targets exhibit high variance.

4 Experimental Methodology

4.1 Game Environment

We evaluate reward shaping on Othello [7] (also known as Reversi), which offers rich strategic play while remaining tractable and supports defining heuristic-based potential functions. Performance is measured against Edax [8], a powerful Othello engine. The potential function $\Phi(s)$ combines seven strategic heuristics: **(1) Disc difference** measures the numerical imbalance between the player's and opponent's discs. **(2) Mobility** quantifies available legal moves, encouraging flexibility and tempo control. **(3) Potential mobility** counts empty squares adjacent to opponent discs, promoting long-term positional prospects. **(4) Corner occupancy** rewards capturing corners, which are stable and cannot be flipped. **(5) Corner-closeness** penalizes positions near unoccupied corners that often enable opponent corner capture. **(6) Frontier discs** counts pieces adjacent to empty squares, penalizing vulnerable configurations. **(7) Stable edge evaluation** rewards secure edge formations connected to corners, which are difficult to reverse. The final $\Phi(s)$ is a weighted linear combination of these heuristics.

ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.

4.2 Experimental Design

Our experimental evaluation compares four configurations. The baseline implements standard AlphaZero without any reward shaping modifications. The MCTS-only configuration applies search-time shaping with $\lambda_{\text{mcts}} = 0.15$, affecting only the values used during tree search while maintaining pure terminal rewards as training targets. Auxiliary head prediction starts with $\eta = 1.0$ until $\eta = 0.1$ after 20 iterations. We also include a method that combines MCTS search with an auxiliary head.
All configurations employ identical network architectures, MCTS parameters, and training hyperparameters to ensure fair comparison. During testing, the number of MCTS simulations was set to 200 and all shaping heuristics were turned off - shaping only affects training. We measure performance by comparing win rate against different levels of gameplay of the Edax Othello engine. We assume that a difficulty level is reached if our model achieves a 75% win rate against the corresponding Edax-level. The maximum reachable level is set at Edax-level 30, which corresponds to superhuman play.

5 Results and Discussion

Figure 1 summarizes for each method how many training iterations it took to reach Edax level 30, as a function of the number of MCTS simulations during training (lower is better). From this figure, we can conclude:

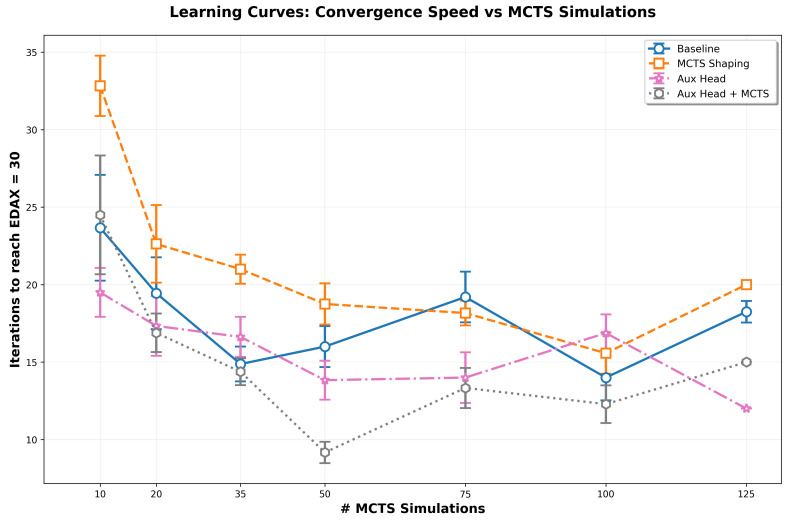

Fig. 1: Number of training iterations required to reach Edax level 30

1. Guiding the search directly with heuristic potentials (MCTS shaping) performs slightly worse than baseline model.

ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.

2. The use of an auxiliary head improves performance slightly, but not always

3. The only method that consistently outperforms the baseline is a combination of the MCTS shaping with an auxiliary head, although the advantage decreases as the number of simulations increases. This is likely due to the fact that with deeper exploration, AlphaZero is capable of finding successful strategies itself, without the need for any guidance.

6 Conclusion and Future Work

This paper provides an initial assessment on the use of shaping methods with AlphaZero. Our experimental results reveal that guiding the search directly with heuristic potentials (MCTS shaping) performs slightly worse than the baseline model, while using an auxiliary head improves performance slightly but inconsistently. However, the combination of MCTS shaping with an auxiliary head consistently outperforms the baseline, though this advantage diminishes as the number of MCTS simulations increases.
While these results are specific to Othello, we hypothesize that similar benefits may emerge in other fully-observable games with comparable characteristics, though this conjecture requires empirical validation. Future work should validate these findings on games with larger action spaces, where focused search may yield greater sample efficiency. Other approaches, such as shaping of value z before network training, should also be explored.

References

[1] David Silver et al. *Mastering chess and shogi by self-play with a general reinforcement learning algorithm*. arXiv preprint arXiv:1712.01815, 2017.

[2] Browne, Cameron B., et al. "A survey of monte carlo tree search methods." IEEE Transactions on Computational Intelligence and AI in games 4.1 (2012): 1-43.

[3] Oriol Vinyals, et al. *Grandmaster level in StarCraft II using multi-agent reinforcement learning*. Nature, 575(7782):350–354, 2019.

[4] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. Vol. 1. No. 1. Cambridge: MIT press, 1998.

[5] Andrew Y. Ng, Daishi Harada, and Stuart Russell. *Policy invariance under reward transformations: Theory and application to reward shaping*. In Proceedings of the 16th International Conference on Machine Learning (ICML), pages 278–287, 1999.

[6] Jaderberg, Max et al. *Reinforcement learning with unsupervised auxiliary tasks*. arXiv preprint arXiv:1611.05397, 2016

[7] Rose, Brian. *Othello: A Minute to Learn, A Lifetime to Master*. Landsdown Press, 1982.

[8] Delorme, Richard - edax-reversi. Available at: *https://github.com/abulmo/edax-reversi*, 2021.