Proceedings of the 4th International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/2026.39319

Potential-Based Reward Shaping for Compute-Constrained Self-Play Reinforcement Learning in Animal Chess

Zezheng Liu

School of Informatics, University of Edinburgh, Edinburgh, UK
S2576948@ed.ac.uk

Abstract. Self-play reinforcement learning can produce capable game-playing policies without expert demonstrations, but its effectiveness under limited compute depends strongly on the learning signal. A complete self-play system for Animal Chess (Dou Shou Qi) was independently designed and implemented to test whether potential-based reward shaping improves sample efficiency under a fixed interaction budget. A vectorised Proximal Policy Optimisation pipeline used 48 synchronous environments, a residual policy-value network, legal-action masking, deterministic execution, and five pre-declared seeds per condition. Sparse terminal reward, potential-based shaping, and a non-potential dense reward were compared under one million nominal environment steps per run. The primary outcome was the number of environment steps required to sustain a 95% win rate against a fixed uniform-random legal-move opponent. Potential-based shaping reduced the mean threshold time from 95,894 to 61,488 steps, a 35.9% reduction supported by a stratified permutation test ($p = 0.0317$). Final game length did not differ detectably between the sparse and potential-based arms. The dense reward learned faster still but lacked the policy-invariance guarantee and was executed under a cross-machine caveat. These findings show that potential-based shaping offers a reproducible, theoretically grounded improvement in sample efficiency under constrained compute.

Keywords: Reinforcement learning, self-play, reward shaping, sample efficiency, Animal Chess

1. Introduction

Self-play reinforcement learning (RL) offers a general route to game-playing agents that improve without expert demonstrations. Deep Q-learning established neural control from high-dimensional observations [1], while AlphaGo and AlphaZero showed that policy-value learning, search, and self-play can attain exceptional performance in complex board games [2, 3]. Those systems, however, relied on resources far beyond most individual studies. This study therefore examines which reward-design choices measurably improve learning efficiency under a fixed interaction budget.

Sparse-reward environments make this question difficult because feedback arriving only at episode termination gives early trajectories little guidance. Count-based bonuses and curiosity objectives create intrinsic rewards from novelty or prediction error [4, 5]. Reward shaping instead incorporates domain knowledge directly. Potential-based shaping constructs intermediate feedback

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.39319

from discounted differences in a state potential and, under the original Markov decision process, preserves the set of optimal policies [6]. This distinguishes it from arbitrary dense rewards, which may accelerate training but change the objective. Its relationship to value initialisation has also been established [7].

Animal Chess (Dou Shou Qi) provides a useful testbed. It is a two-player, zero-sum, perfect-information game on a 9 × 7 board, with rank-dependent captures, river jumps, traps, and dens. Computational studies established its PSPACE-hardness and solved only restricted endgames with up to four pieces [8, 9]. Learning under a controlled interaction budget remains underexplored. In a 200-game audit, 69.5% of uniform-random games reached the 300-ply limit and ended as draws. An untrained policy therefore receives no decisive terminal reward from most early episodes.

Reliable comparison also requires more than one successful run. Deep-RL results vary with seeds, implementation choices, and reporting conventions [10]. Robust aggregates and interval estimates are therefore preferable in the few-run regime [11]. The study used five pre-declared seeds per condition, complete learning curves, bootstrap intervals, and permutation tests. Proximal Policy Optimisation (PPO) was selected for its clipped objective and comparatively auditable implementation [12].

This paper presents an independently designed and implemented self-play PPO system and compares three reward designs under a nominal one-million-step budget: sparse terminal reward, potential-based shaping, and a non-potential dense reward. The primary outcome was environment steps to a sustained 95% win rate against a fixed uniform-random legal-move opponent. Potential-based shaping reduced the mean threshold time from 95,894 to 61,488 steps, a 35.9% reduction (permutation $p = 0.0317$), without a detectable change in final game length. The dense reward reached the threshold still earlier, but without the policy-invariance guarantee and under a cross-machine execution caveat. The study therefore isolates a practical trade-off between theoretically safe guidance and more aggressive reward engineering under constrained compute.

2. Method

2.1. Environment and state representation

Animal Chess was implemented as a deterministic, two-player, zero-sum environment with a frozen interface contract. The board contains 63 squares arranged as 9 rows by 7 columns. Each player begins with eight ranked pieces, while rivers, traps, and dens modify movement or capture rules. Episodes ended for one of four reasons: entry into the opponent's den, capture of all opposing pieces, absence of a legal move for the player to act, or the 300-ply limit. The first three conditions produced a decisive outcome; only the move limit was scored as a draw. Rule behaviour, legal-action generation, terminal outcomes, and reward semantics were covered by 22 pytest cases, 10 rule checks, and four reward-contract checks, for 36 independently executed checks in total.

A single neural network played both colours. To make this possible, every position was transformed into the perspective of the player to move. The observation was a $19 \times 9 \times 7$ tensor: eight planes encoded the acting player's pieces, eight encoded the opponent's pieces, and three represented rivers, traps, and dens. The action space contained 504 indices, obtained by pairing each of the 63 origin squares with eight movement directions. Illegal actions were masked by adding -10^9 to their logits before the softmax, ensuring that the policy normalised only over legal moves. This masking was essential because only 24 of 504 actions are legal in the initial position.

DOI: 10.54254/2755-2721/2026.39319

2.2. Policy network and PPO training

The learner used PPO [12] with a shared residual policy-value network containing 375,375 trainable parameters. A 3 × 3 convolutional stem mapped the 19 input planes to 64 channels, followed by four residual blocks with Group Normalisation and ReLU activations. The policy head produced 504 masked action logits, and the value head produced a scalar in [−1, 1] through a tanh output. The same network therefore estimated both the legal-action distribution and the expected outcome from the current player's perspective.

For each policy update, the probability ratio between the current and rollout policies was constrained by the clipped surrogate objective in Equation (1), where $\hat{A}_t$ denotes the estimated advantage and ε is the clipping range.

$$L^{\mathrm{CLIP}}(\theta) = \mathbb{E}_t[\min(r_t(\theta)\widehat{A}_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}_t)] \tag{1}$$

Here, the probability ratio compares the current policy with the rollout policy for the sampled action. The implementation minimised the combined objective in Equation (2), where the second term is the clipped value loss and the final term is policy entropy.

$$L_{\mathrm{total}}(\theta) = -L^{\mathrm{CLIP}}(\theta) + 0.5L_V(\theta) - 0.01H(\pi_\theta) \tag{2}$$

Training data were generated by 48 synchronous self-play environments. A standard rollout contained 128 vector steps, equivalent to 48 × 128 = 6,144 environment transitions. Returns and advantages were computed with discount γ = 0.99. Because the observation frame changed whenever the acting player changed, bootstrapped values were negated across turns so that the target always represented the current player's expected outcome. CUDA automatic mixed precision and gradient clipping were used during optimisation, while deterministic kernel settings and fixed seeds reduced avoidable execution variance. Each formal run used a nominal budget of 1,000,000 transitions and completed 999,984 transitions over 163 updates; the final update contained 97 rather than 128 vector steps. No run used early stopping.

2.3. Reward conditions

Three reward conditions were evaluated while keeping the environment, network, optimiser, training budget, and evaluation procedure fixed. E1 was the sparse baseline: +1 for a win, −1 for a loss, and 0 for a draw, with no non-terminal reward. E2 added potential-based shaping to the same terminal reward. Its potential combined normalised material balance and positional progress, weighted by α = 0.5 and β = 0.1, with material normalised by 80. The potential was bounded to ±0.35 and set to zero in the initial position. The shaped reward was computed using Equation (3). Since s_{t+1} was stored in the next player's frame, its potential was negated before the temporal difference was formed; this sign correction preserved a consistent player perspective.

$$r'_t = r_t + \gamma\Phi(s_{t+1}) - \Phi(s_t) \tag{3}$$

E8 was a deliberately non-potential contrast condition. It combined capture reward, advancement reward, and a small step cost with weights 0.30, 0.010, and 0.002, respectively. Unlike E2, this reward cannot inherit the policy-invariance result in [6], so it was interpreted as an engineering upper contrast rather than a theoretically equivalent objective. A pure Monte Carlo tree search

DOI: 10.54254/2755-2721/2026.39319

implementation using 256 simulations per move and a handcrafted material-progress-mobility evaluator was also run as a functional sanity check. MCTS followed upper-confidence tree selection [13] but was not included in the reward-design hypothesis test.

The three reward conditions and their interpretive roles are summarised in Table 1.

Table 1. Reward conditions in the formal comparison

Arm	Reward signal	Interpretive role
E1	Terminal only: +1 win, −1 loss, 0 draw	Sparse baseline
E2	Terminal reward + $\gamma\Phi(s') - \Phi(s)$	Policy-invariant shaping under the stated MDP assumptions
E8	Capture + advancement − step cost	Non-potential dense contrast

2.4. Evaluation, statistics, and reproducibility

Each reward condition used five pre-declared training seeds. Evaluation was triggered every 5,000 configured steps and consisted of 200 greedy games against the same uniform-random legal-move opponent, with seats alternated and draws scored as 0.5. Evaluation randomness followed Equation (4), producing non-overlapping random streams across runs.

$$s_{\text{eval}} = 20260821 + 1000 s_{\text{train}} + i_{\text{eval}} \tag{4}$$

Because evaluation could occur only after a vectorised rollout, the effective horizontal resolution was 6,144 steps. The 200 stored evaluation records in each run corresponded to 163 distinct environment-step marks; 37 duplicated marks were grouped by environment steps before curves or threshold statistics were computed.

The pre-declared primary measure was the first evaluated step at which win rate reached at least 0.95 and remained at or above 0.95 for every later evaluation in that run. This sustained definition avoided treating a temporary upward fluctuation as convergence. Secondary outcomes were normalised area under the win-rate curve during the first 100,000 steps (AUC@100k), final mean episode length, and final captures per game. Results were aggregated across seeds using the mean, median, and interquartile mean. Uncertainty was estimated with stratified bootstrap 95% confidence intervals, and arm differences were tested with a stratified permutation test. Statistical interpretation followed the few-run recommendations in [11], with $p < 0.05$ treated as evidence against the null rather than as a measure of practical importance.

All formal records were written as JSONL and validated before analysis. The released data comprise 15 real-environment logs: E1, E2, and E8 with five seeds each. Every log contains one metadata record, 122 training records, and 200 evaluation records. Validation rejected malformed schemas, non-finite values, and inconsistent accounting. E1 and E2 used the same commit and machine, permitting environment-step and timing comparisons. E8 used a different commit and machine; it was therefore compared with the other conditions only in environment steps, never in wall-clock time or throughput.

3. Results and discussion

The principal outcomes for the three formal reward conditions are summarised in Table 2. E8 was executed from a different code commit and on a different machine; consequently, comparisons

DOI: 10.54254/2755-2721/2026.39319

across all three arms are restricted to environment steps, while wall-clock time and throughput are not compared.

Table 2. Formal reward-design results averaged over five seeds

Arm	Sustained 0.95(steps)	AUC@100k	Final capturesper game	Final length(plies)
E1	95,894	0.6434	1.48	38.22
E2	61,488	0.7985	1.68	38.52
E8	49,200	0.8600	2.22	33.11

3.1. Sample efficiency

Potential-based shaping improved the primary measure consistently across the five formal seeds. E1 reached a sustained 0.95 win rate after 86,064, 129,072, 104,496, 92,208, and 67,632 environment steps. The corresponding E2 values were 43,056, 67,632, 67,632, 67,632, and 61,488 steps. Mean threshold time therefore fell from 95,894 to 61,488 steps, an absolute reduction of 34,406 steps and a relative reduction of 35.9%. At the experiment's 6,144-step resolution, the effect was equivalent to 5.6 evaluation ticks, making it substantially larger than the quantisation floor.

The bootstrap 95% confidence interval for the mean reduction was [15,974, 55,296] steps, and the stratified permutation test gave $p = 0.0317$. The E1 and E2 learning curves were non-overlapping except for a single tied evaluation tick at 67,632 steps. Together, these observations support the conclusion that the E2 result was not created by one unusually favourable seed. They do not, however, imply faster learning against every possible opponent: all competence measurements used the same fixed random benchmark.

E8 reached the same sustained threshold in 49,200 steps on average, faster than both E2 and E1. This result answers the engineering question differently from the theoretical one. Stronger task-specific feedback can produce greater apparent sample efficiency, but E8 provides no guarantee that the original objective is preserved. Moreover, because E8 was trained from a different commit and on a different machine, only its environment-step count is directly compared here. Its wall-clock time and throughput cannot be used to rank the three conditions.

3.2. Early-curve performance

AUC@100k captured performance across the early learning curve rather than the timing of one sustained threshold. The mean values were 0.6434 for E1, 0.7985 for E2, and 0.8600 for E8. The E1-to-E2 difference was directionally consistent with the primary result, but its permutation p-value was 0.0556 and therefore did not meet the pre-specified 0.05 criterion. This distinction matters: the study found statistically significant evidence for earlier sustained competence, but not for every plausible summary of early learning.

The difference between the two measures is explained by the non-monotonic nature of the curves. Evaluation batches are finite, and a policy can briefly rise above or fall below a threshold even while its broader trend improves. A long calibration run had previously exhibited a $0.970 \rightarrow 0.818 \rightarrow 0.983$ sequence, motivating the sustained definition. Reporting both outcomes avoids presenting a threshold-only result as universal evidence of curve dominance. It also demonstrates that the conclusion is strongest when stated narrowly: potential shaping reduced the time required to attain stable benchmark competence under this protocol.

DOI: 10.54254/2755-2721/2026.39319

3.3. Behavioural outcomes

Final behaviour did not indicate a capture-driven shortcut. Mean captures per game were 1.48 for E1, 1.68 for E2, and 2.22 for E8, all below the 4.54 captures observed in the uniform-random audit. E2 therefore learned faster than E1 without producing a large increase in capture frequency. This pattern is compatible with the potential function guiding the agent toward strategically useful material and positional states rather than simply rewarding every capture.

Final mean game length was 38.22 plies for E1 and 38.52 for E2. The arm comparison gave $p = 1.0000$, a clear negative result: no detectable change in game length was found. The near-identical endpoint is consistent with, but does not prove, the policy-invariance motivation for potential-based shaping. The theoretical guarantee concerns optimal policies under the stated MDP assumptions, whereas the present agents were finite-budget approximations evaluated against one weak opponent. E8 ended games in 33.11 plies on average and produced more captures, but these descriptive differences should not be attributed causally without a same-commit replication and an explicit behavioural hypothesis test.

Endpoint win rate alone was uninformative because the benchmark saturated. Across the five final E1 evaluation batches, the agent recorded 1,000 wins, no draws, and no losses. This establishes reliable end-of-run performance against the random opponent, not general mastery of Animal Chess. The MCTS sanity baseline likewise won all 150 games played against random, confirming that the search implementation functioned but providing no direct ranking between MCTS and the trained policies.

3.4. Discussion

The results show that reward design can materially alter learning speed even when the network, algorithm, and interaction budget are held fixed. Potential-based shaping recovered roughly one third of the samples required by the sparse baseline while retaining a principled connection to the original terminal objective. The dense condition was faster still, illustrating the cost of that theoretical restraint: policy invariance is a safety property, not a guarantee of maximal acceleration.

The evidence remains deliberately narrow. Five seeds reduce but do not eliminate uncertainty; a 6,144-step evaluation grid limits temporal precision; and success against a uniform-random opponent does not establish strength against trained policies, search agents, or human play. The next decisive experiment would reproduce all three rewards from one commit and machine, then evaluate them against a stronger, pre-registered opponent set while retaining per-game behavioural records. Within the present scope, the supported conclusion is that potential-based reward shaping provided a statistically detectable and reproducible sample-efficiency gain under a one-million-step, compute-constrained Animal Chess protocol.

4. Conclusion

This study examined whether potential-based reward shaping can improve sample efficiency in compute-constrained self-play reinforcement learning for Animal Chess. Under a fixed nominal budget of one million environment steps per run, potential-based shaping reduced the mean number of steps required to sustain a 95% win rate against the fixed random benchmark from 95,894 to 61,488. The reduction was statistically detectable and remained large relative to the 6,144-step evaluation resolution. At the same time, final game length was virtually unchanged, and capture frequency showed no evidence of a capture-driven shortcut. These results are consistent with the

DOI: 10.54254/2755-2721/2026.39319

intended role of potential-based shaping: accelerating learning while retaining a principled connection to the original sparse terminal objective. The non-potential dense reward reached the threshold sooner, demonstrating that stronger task-specific feedback may deliver greater speed when policy invariance is not required. However, that arm was executed from a different code commit and machine, so only environment-step comparisons are defensible. Overall, the evidence supports potential-based shaping as a safe and reproducible accelerator within this experimental setting, while stronger opponents, same-commit replication, and broader behavioural evaluation remain necessary before claiming general Animal Chess competence.

References

[1] Mnih, V., et al. (2015). Human-level control through deep reinforcement learning. *Nature, 518*, 529–533.
[2] Silver, D., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature, 529*, 484–489.
[3] Silver, D., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science, 362*(6419), 1140–1144.
[4] Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., & Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. *Advances in Neural Information Processing Systems, 29*.
[5] Pathak, D., Agrawal, P., Efros, A. A., & Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. *Proceedings of the 34th International Conference on Machine Learning*, 2778–2787.
[6] Ng, A. Y., Harada, D., & Russell, S. J. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. *Proceedings of the Sixteenth International Conference on Machine Learning*, 278-287.
[7] Wiewiora, E. (2003). Potential-based shaping and Q-value initialization are equivalent. *Journal of Artificial Intelligence Research, 19*, 205-208.
[8] van Rijn, J. N., & Vis, J. K. (2013). Complexity and retrograde analysis of the game Dou Shou Qi. *Proceedings of the 25th Benelux Conference on Artificial Intelligence*, 239-246.
[9] van Rijn, J. N., & Vis, J. K. (2014). Endgame analysis of Dou Shou Qi. *ICGA Journal, 37*(2), 120-124.
[10] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018). Deep reinforcement learning that matters. *Proceedings of the AAAI Conference on Artificial Intelligence, 32*(1), 3207-3214.
[11] Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., & Bellemare, M. G. (2021). Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems, 34*, 29304-29320.
[12] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv Preprint, arXiv:1707.06347.
[13] Kocsis, L., & Szepesvári, C. (2006). Bandit based Monte-Carlo planning. *Machine Learning: ECML 2006, Lecture Notes in Computer Science, 4212*, 282-293. Springer.