# Studying the Impact of Domain Ergodicity on Efficiency of Reinforcement Learning and Training through Self-Play

December 5, 2011

## 1 Introduction

In the paper titled "Why did TD-Gammon Work?" [1] Pollack attributes the success of TD-gammon to "the setup of co-evolutionary self-play biased by the dynamics of backgammon." Pollack used a hill climbing solution to achieve some of the success of TD-gammon. Tesauro wrote a follow-up [3] and argued that the difference between the performance of TD-gammon against the Pubeval program (56% win) and that of Pollack's solution (40-45% win) is significant and compares to the difference between a world class player and an average player. Nonetheless, the success of TD-gammon [2] has not been repeated in other domains.

In his paper, Pollack does not discuss exactly what aspects of the dynamics of backgammon are making self-play and reinforcement learning work. Our hypothesis is that in part it has something to do with the partially ergodic nature of backgammon. It is possible that this hypothesis is not correct, but it would be worth exploring.

If the MDP is observed for an infinitely long period of time, a *recurrent* state is one that gets visited infinitely many times. On the other hand, a *transient* state is one which will be visited a finite number of times in an infinite viewing of the MDP. An *ergodic* MDP is an MDP in which every state is recurrent. In other words, in an ergodic MDP every state can be reached from every other state through some transitions. Therefore, if an agent is making decisions in an ergodic MDP, no current choices can cause it to be limited to a restricted region of the state space in the MDP.

Backgammon is not a completely ergodic game, as there are transient states in the game, for example in the endgame. But, I think we can also use ergodicity in a looser sense and consider a *degree of ergodicity* by comparing the number of recurrent states to the number of transient ones. One can also consider a *degree of recurrency* for a state, based on the expected likelihood of visitations to that state. With these notions, one can say that backgammon exhibits a higher degree of ergodicity and has states that have higher degrees of recurrency, compared to some other games.

As an example, in the game of chess, once the first piece is captured, none of the board configurations which include all the pieces on the board can be visited anymore. In chess, most middle-game states are highly transient. Therefore we can say that the game is less ergodic.

The game of backgammon consists of two phases. In the first phase, the players try to move all their checkers past the opponent's checkers. Since the checkers for the two players move in opposite directions, there is a great chance that the players hit one another's checkers. If a checker is hit, it has to enter the board again. Once a player has moved all their checkers to the last quadrant on the board, the second phase of the game–the race–begins, where the player can take their checkers off the board. The first player to finish wins the game.

Since no checker exits the game in the first phase, all 15 checkers from each player are always going to be in play. This makes most middle-game positions in backgammon recurrent states, as they can be repeatedly visited in the same game.

One interesting fact in support of our hypothesis is that TD-gammon was pretty strong in middle-game positions and pretty weak in the endgame, which does not have recurrent states. Our hypothesis is that the higher degree of ergodicity helps TD-gammon in two ways:

1. It helps with self-play. This is because in any given game, the players are likely to visit many middle-game positions. This, in turn, is caused by the constant hitting and re-entering. Consider a solution that uses an evolutionary setting to evolve a backgammon agent using self-play. When a challenger is matched up against a champion, the challenger will need to have a broad strength on many middle-game positions in order to beat the champion. If the challenger makes slightly better moves in some positions and really bad moves in some others, it is going to have a harder time beating the champion in backgammon. This is because the ergodic nature of the game makes it more likely that those weak positions are visited in any single game.

   Moreover, no matter what opening moves are taken, most middle-game positions can still be encountered. So, this rules out a common problem with self-play and co-evolutionary setups, which is developing strategies that cover only a restricted part of the state space. Usually, such suboptimal strategies can be developed due to some bias created early on for choosing a specific series of opening moves.

2. It helps with reinforcement learning, because the high degree of recurrency in the states makes them roughly equally likely to be encountered in the games in the long run. So, the amount of experience that the agent gathers for updating the state values would be balanced more evenly among the game states.

## 2 Proposal

We propose to study the effect of ergodicity by designing a problem domain that can show varying degrees of ergodicity. Using this domain, we try to evaluate the impact of ergodicity on the learning performance of a reinforcement learning solution and an evolutionary solution using self-play.

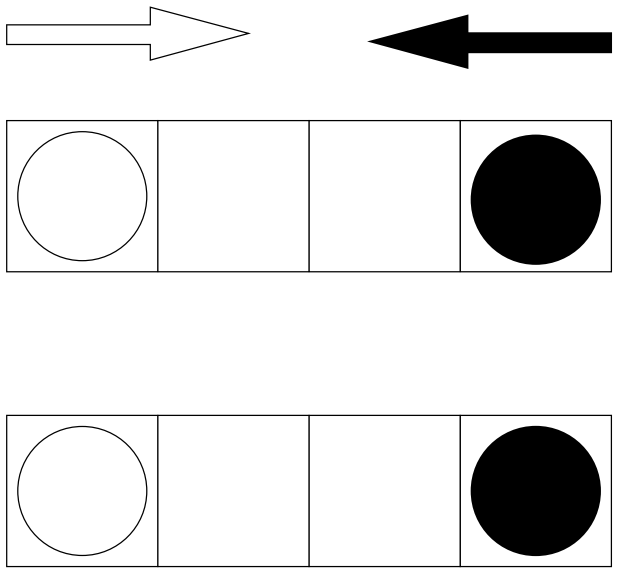

Figure 1: Minigammon Board

The test domain that we have in mind is a simplified version of the backgammon game, whose board is shown in Figure 1.

The board consists of two rows, each with four cells. Each player has two checkers starting at the shown positions. The players take turns and move the checkers in the shown directions. During each turn, the player tosses a coin whose two sides read 1 and 2. The player then picks one of the two checkers and moves it ahead 1 or 2 cells, depending on the outcome of the coin toss. Each cell can be occupied by one checker only. Once a checker on the top row reaches the end, it continues on to the bottom row, entering from the direction of the player.

Players can also capture the opponent's checkers, which causes the checker to be taken off the board. If a checker is captured, the opponent has to play that checker on the next turn. Depending on the outcome of the coin toss, the captured checker will enter on the first or second cell on the top row. If the player has another checker on the same cell,they can't enter the captured checker. If a player has no legal moves, the turn will change and the other player continues.

Once a player has both checkers on the bottom row, they can start taking checkers off the board by moving them past the last cell on the opponent's side. The player who takes both checkers off first wins the game.

The board and the rules have been designed to limit the number of possible states so that a tabular TD method can be used for reinforcement learning.

Requiring that captured checkers enter on the top row increases the chance for conflicts on the board and the likelihood that configurations are repeatedly visited during a single game. This corresponds to a higher degree of ergodicity in the domain. On the other hand, if we require that all captured checkers enter on the bottom row, then some configurations would never be repeated. In particular, since no checker ever re-enters on the top row, configurations with some checker on top first row would be encountered only during the beginning of each game. This increases the number of transient states and reduces the degree of ergodicity.

One can imagine a spectrum of possibilities between these two cases by making the selection of the re-entry row a probabilistic process. If $p$ represents the probability by which a captured checker enters on the top row, then higher values for $p$ would correspond to higher degrees of ergodicity, as it increases the degree of recurrency of those states that include some checker on the top row.

## 3 Experiment Setup

For reinforcement learning: Use self-play for training. At the end of each episode, evaluate the policy against a random opponent for a number of games and record the number of games won.

For evolutionary: Evolve a neural network (topology similar to the ones used by Tesauro and Pollack). During each generation, mutate challengers by adding Gaussian noise to the weights. If the challenger beats the champion, move the champion's weights in the direction of the challenger. Then, evaluate the generation champion against the random opponent.

We would run these learning experiments with varying values of $p$ and evaluate its impact on the learning rates of both methods.

Even if the test domain that we developed does not produce convincing results, this project would give us the ingredients to try different test domains later on.

## References

[1] J. Pollack and A. Blair. Why did td-gammon work? In *Advances in Neural Information Processing Systems*, pages 10–16. Citeseer, 1997.

[2] G. Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[3] G. Tesauro. Comments on co-evolution in the successful learning of backgammon strategy. *Mach. Learn.*, 32:241–243, September 1998.