2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Monotonic Model Improvement Self-play Algorithm for Adversarial Games

Poorna Syama Sundar, Manjunath Vasam, Ajin George Joseph

Abstract— **The problem of solving strategy games has intrigued the scientific community for centuries. In this paper, we consider two-player adversarial zero-sum symmetric games with zero information loss. Here, both players are continuously attempting to make decisions that will change the current game state to his/her advantage and hence the gains of one player are always equal to the losses of the other player. In this paper, we propose a model improvement self-play algorithm, where the agent iteratively switches roles to subdue the current adversary strategy. This monotonic improvement sequence leads to the ultimate development of a monolithic, competent absolute no-loss policy for the game environment. This tactic is the first of its kind in the setting of two-player adversarial games. Our approach could perform competitively and sometimes expertly in games such as 4x4 tic-tac-toe, 5x5 domineering, cram, and dots & boxes with a minimum number of moves.**

I. INTRODUCTION

In a turn-based adversarial zero-sum game, an opponent player and the agent attempt to alter the current state of the game to his advantage, where the profits of one player are equal to the losses of the other player. Tic-tac-toe is the classical illustration of an adversarial zero-sum game. At every instant, one player's reward is the other player's anti-reward in equal measure. Other games in this category include chess, Backgammon, Go, and Othello, which are very complex with total board configurations in the range of the number of atoms in the universe. The turn-based two-player adversarial games can be conceptualized as a time-homogeneous, discrete-time, sequential decision-making problem under uncertainty with a finite number of states and actions, where the uncertainty results from the opponent's moves being absolutely unpredictable.

Adversarial games are always been foremost among the challenging problems in artificial intelligence. Several approaches are proposed dating back to the work of Shannon and Turing to automate chess [1]. Another early solution includes the application of self-play to efficiently learn to play Checkers by Arthur Samuel. With the advancement of learning algorithms during the late twentieth century, one could observe an active application of learning methods to tackle adversarial games as deterministic search methods fail to find optimal solutions due to the combinatorial explosive nature of the problem. For example, in a 4×4 tic-tac-toe setting, there are 4×10^{13} different games possible. In [2], the model-free reinforcement learning algorithm Q-Learning is applied to train an RL agent to learn to play tic-tac-toe expertly in full-board and partial-board representations of the game. However, the approach couldn't achieve zero loss performance, thus hindering its potential for superhuman capabilities in tic-tac-toe. In another approach [3], learning is induced in an RL agent through neural networks and temporal difference learning in the 3D version of tic-tac-toe. Here, the results obtained depend on the reference benchmark player that is used to train the agent. This is envisaged since the generalization ability of the learning methods is mostly biased toward the distribution of the sample data. To achieve superhuman performance, one might require samples of that nature, which are hard to come by. Another approach [4] involves the application of least-squares policy iteration which is a model-free reinforcement learning algorithm that performed competitively on Othello.

A hybrid approach where Q-learning and min-max are combined to solve a 2D soccer game [5], where the max operator in the Q-learning algorithm is replaced with min-max, which induces risk-averse behavior. The use of min-max in the Q-learning algorithm suggests that the agent may take a more conservative approach, prioritizing minimizing potential losses over maximizing rewards. A decentralized decision-making algorithm [6] was proposed, which was claimed to be a no-loss strategy, but later found that the strategy failed in three different scenarios [7].

In [8], a genetic algorithm-based solution to generate a single no-loss strategy is proposed. Genetic algorithms are being used extensively in combinatorial optimization problems due to their propensity to find the global optimum solutions. However, to obtain the optimum performance, one has to fine-tune too many hyper-parameters. AI enthusiasts are well aware of the min-max algorithm [9], which is mostly deployed for two-player adversarial games because of its inherent nature. Min-max is a technique commonly used in game theory for decision-making, where a player chooses an action that minimizes the maximum possible loss. The main drawback of this algorithm is that for complex games the state space blows up and the task becomes computationally expensive due to coupled minimum and maximum operations performed at each instant. There are modifications to the classical min-max algorithm like the alpha-beta pruning [10], which can at most reduce the complexity of the algorithm by half which is not asymptotically significant. Another approach based on a customized decision tree is elucidated in [11] specifically for 3×3 tic-tac-toe. The decision tree is built a priori using pre-defined logic obtained through

Poorna Syama Sundar, Department of Computer Science & Engineering, Indian Institute of Technology Tirupati, cs19b049@iittp.ac.in.
Manjunath Vasam, Department of Computer Science & Engineering, Indian Institute of Technology Tirupati, cs19b041@iittp.ac.in.
Ajin George Joseph, Department of Computer Science & Engineering, Indian Institute of Technology Tirupati, ajin@iittp.ac.in.

analysis of the game. This paper demonstrates a significant shortcoming of the min-max algorithm, particularly in tic-tac-toe games regarding the non-optimal move selection at certain instances, where the min-max algorithm takes additional sub-optimal steps to win a game even though the agent had a chance to win the game early. Despite this drawback, min-max is still a competitive no-loss strategy. The tic-tac-toe game possesses multiple no-loss strategies and this was explored in [7], where the customized genetic algorithms are applied to gather all the no-loss strategies.

In this paper, we propose an algorithm that can obtain an efficient, competent no-loss policy for moderate state-space, symmetric, adversarial board games. The algorithm seeks a no-loss policy, meaning that the policy is competent enough to play the game without losing, regardless of the opponent's moves. The efficiency is in terms of the maximum number of moves required to complete the game. This is important for board games where players aim to win in the shortest number of moves possible. Unlike many of the above algorithms, our algorithm doesn't require any input game episodes for training. Instead, our algorithm adopts a self-play-based model improvement mechanism that converges to the competent policy.

II. Background

A discrete-time, time-homogeneous Markov Decision Process (MDP) $\{(X_t, A_t)\}_{t\in\mathbb{N}}$ [12], [13], [14] is a framework for sequential decision-making paradigm under uncertainty, where an agent learns to interact with a stochastic environment through a set of actions and a reward-penalty response. This paradigm is represented by the 4-tuple (S, A, P, R), where $X_t \in S$ is the state space, $A_t \in A$ is the action space, $P : S \times A \times S \to [0,1]$ is the probability transition function and $R : S \times A \times S$ is the reward function. For $s, s' \in S, a \in A$, we have $P(s,a,s') = \mathbb{P}(X_{t+1} = s' | X_t = s, A_t = a)$, $\forall t \geq 0$, which is the probability of transitioning to the next state s' conditioned on the current state being s and current action is a. The transition matrix which defines the dynamics of the system is referred to as the *model*. The transition at every instant t is associated with a scalar reward $R(X_t, A_t, X_{t+1})$. At every instant t, the agent performs an action according to a deterministic policy $\pi : S \times A$. The goal of the setting is find the optimal policy π^* which is defined as follows:

$$\pi^* = \arg\max_{\pi} V_\pi, \tag{1}$$

$$\text{where } V_\pi(s) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t R(X_t, \pi(X_t), X_{t+1}) \middle| X_0 = s\right], s \in S.$$

Here, V_π is referred to as the value-function associated with policy π, and $\gamma \in [0,1)$ is the discount factor.

III. Proposed Work

The approach presented in this paper views the adversarial game setting as similar to how humans learn to play games. The initial stage involves humans playing the game randomly against an adversary, which may result in poor performance. However, with multiple trial games, humans continuously improve their skills. If the adversary is fixed, the human can only reach the level of the adversary's skill. But when the human and the adversary are in a similar skill range and the adversary is able to improve, then both parties start developing until no further improvement is possible. This stage is referred to as "self-play," where the human plays against a version of itself. In self-play, the human player continuously refines its strategies and adapts to the changing behavior of the adversary, leading to an iterative process of improvement. This approach mimics how humans learn through practice and experience, gradually honing their skills over time. Self-play allows for dynamic adaptation to the changing environment, as the human player learns from their own actions and refines their strategies accordingly.

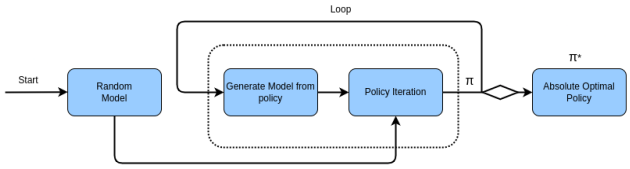

Fig. 1. Proposed algorithm

In our approach, the adversary is modeled using the probability transition function $P_k : S \times A \times S \to [0,1]$ which is constantly evolving with respect to iteration k. At each iteration k, the agent attempts to subdue the adversary by finding the optimal policy π_k^* with respect to the current MDP model with probability transition function P_k. At this stage, the roles are flipped, where the agent now plays the game from the adversary end and a new adversary defined by P_{k+1} which is based on the optimal policy π_{k+1}^* obtained in the previous iteration plays from the other end. Since we are considering only symmetric games, the flip operation is possible. Symmetry refers to games where the same actions and rules apply to all players, and the outcome of the game is unchanged when players swap positions or when the game is reflected or rotated. Many popular board games such as chess, checkers, and Go exhibit symmetry. We refer to this step as the model improvement step. This improvement step allows the agent to continuously adapt to the changing behavior of the adversary and refine its strategies accordingly, resulting in an iterative process of improving the model of the adversary. To elucidate this further, we define the operator $I(s,a)$, $s \in S$, $a \in A$ which returns the intermediate state after action a is applied to state s by the agent. This intermediate state is not among the state space of the agent. This state will not be seen by the agent, instead, this state is visible only to the opponent on which he performs his action. In the case of tic-tac-toe, this intermediate state is the immediate board configuration just after the agent places his symbol and just before the opponent places his symbol. The probability $\mathbb{P}(\text{next state} = s' | \text{current state} = s, \textit{current action} = a)$ can be viewed as $\mathbb{P}(\text{next state} = s' | \text{intermediate state} = I(s,a))$ which is the probability of transitioning from the intermediate state to the

Algorithm 1 Proposed Algorithm

Input: $S, A, R, I, \gamma, \alpha \in (0.5, 1], \varepsilon > 0$
Output: π^*

1. Initialization:
for $(s \in S, a \in A)$ **do**
$P_0(s,a,.) \sim U(0,1)$ *Initial transition probabilities are uniform*
end for

$k \leftarrow 0$
2. Model Improvement:
while (π_k^* improves) **do**
Initialize $V_0, V_{-1} \in \mathbb{R}^{|S|}, \pi_0, \pi_{-1}$
$t \leftarrow 0$
while $\pi_t \neq \pi_{t-1}$ **do**
2.1 Policy Evaluation:
while $(\|V_j - V_{j-1}\|_\infty \geq \varepsilon)$ **do**
$V_{j+1}(s) \leftarrow \sum_{s'} P_k(s, \pi_t(s), s')[R(s, \pi_t(s), s') + \gamma V_j(s')]$, $\forall s \in S$
$j \leftarrow j + 1$
end while
$V_{t+1} \leftarrow V_j$
2.2 Policy Improvement:
$\pi_{t+1}(s) \leftarrow \arg\max_{a \in A(s)} \sum_{s'} P_k(s,a,s')[R(s,a,s') + \gamma V_{t+1}(s')]$, $\forall s \in S$
$t \leftarrow t + 1$
end while
$\pi_k^* \leftarrow \pi_t$
for $(s \in S, a \in A(s), s' \in S(s))$ **do**
/* $S(s)$ *and* $A(s)$ *are the valid states and actions possible from* s */
$\bar{s} \leftarrow I(s,a)$
if $(I(\bar{s}, \pi_k^*(\bar{s})) = s')$ **then**
$P_{k+1}(s,a,s') = \alpha$ /* *Model updation using the previous optimal policy* */
else
$P_{k+1}(s,a,s') = \frac{1-\alpha}{|S(s)|-1}$ /* *Non-optimal transitions are chosen uniformly at random* */
end if
end for
$k \leftarrow k + 1$
end while
return π_k^*

next state s'. Essentially this is the distribution followed by the opponent when playing the game. The next probability transition function P_{k+1} is defined as follows: For every possible tuple (s,a,s'):

- If the state s' is the obtained by applying an action $\pi_k^*(I(s,a))$ on the intermediate state $I(s,a)$, then $P_{k+1}(s,a,s') = \alpha$, where $\alpha \in (0.5, 1]$ is fixed a priori.
- Else the weight is uniformly distributed over the remaining transitions resulting from the intermediate state $I(s,a)$.

Essentially, this implies that the opponent considers the action suggested by the previous optimal policy with high probability mass (α) while considering the remaining possible actions with low probabilities. This process is repeated until no further improvement in the policy is recognized. The improvement is measured based on the number of games the policy beats, namely wins or in the worst case ties, and the number of games it loses. The use of probability mass α for suggesting actions from the previous optimal policy, along with low probabilities for remaining actions, implies a form of exploration-exploitation trade-off, where the algorithm balances between trying out new actions and exploiting the currently known optimal actions. As a caveat, the initial probability transition function is generated uniformly at random, *i.e.*, for each state s, the transition probabilities follow a uniform distribution. The reward function and discount factor remain unchanged throughout the iterations.

IV. GAMES & RESULTS

We evaluated our proposed algorithm exhaustively in various game settings which include tic-tac-toe($3 \times 3, 4 \times 4$), domineering($4 \times 4, 5 \times 5$), cram and dots & boxes. By exhaustive we mean that the resultant policy is played for all possible games by considering all possible sequences of moves the opponent could make. Tic-tac-toe, domineering, cram, and dots & boxes are all popular games with different complexities and strategies, and testing our algorithm in these different settings can provide valuable insights into its strengths and weaknesses. For all the tested games, the algorithm could produce no-loss strategies. However, except for tic-tac-toe, all other games have a first/second-mover advantage. So, for these games, the algorithm can find the no-loss strategy in cases where the agent has the advantage, and for the rest, the number of losses is minimal.

A. Tic-Tac-Toe

Tic-tac-toe is played on a 3×3 empty grid where players choose either the symbol X or O. Players then take turns filling the board with their initially chosen symbol. The first player to complete a diagonal, row, or column with their symbol is the winner. If no one wins and the grid is full, then the game ends in a tie. Here, the game complexity is as follows: $|S| = 5479$ and $|A| = 10$.

1) Tic-Tac-Toe 3×3*:* The algorithm required 4 iterations of model improvement to reduce the losses to zero. After the first iteration *i.e.*, after training against the initial random adversary, the number of losses is still significant. In each further iteration, the losses are reduced, indicating the improvement of the policy over the flips.

2) Tic-Tac-Toe 3×3 *with Deep Q-Learning:* We consider a variation of our algorithm in which we use deep Q-learning to obtain the optimal policy at each instant on a reduced state space version of tic-tac-toe 3×3. The whole setup is unchanged, except that the deep neural network is now used to get the optimal policy, which necessitates the generation of several game episodes. The model improvement step is

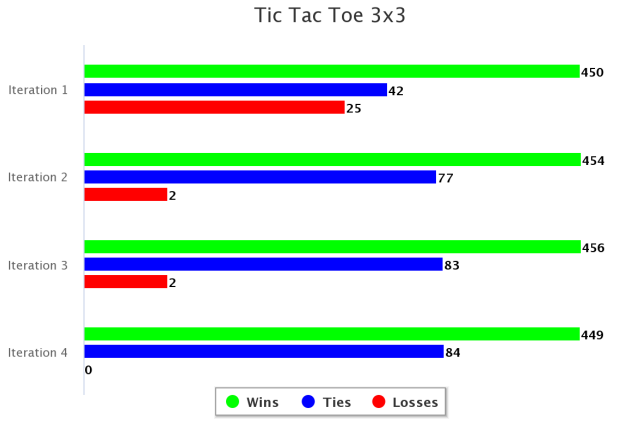

Fig. 2. Tic Tac Toe 3×3

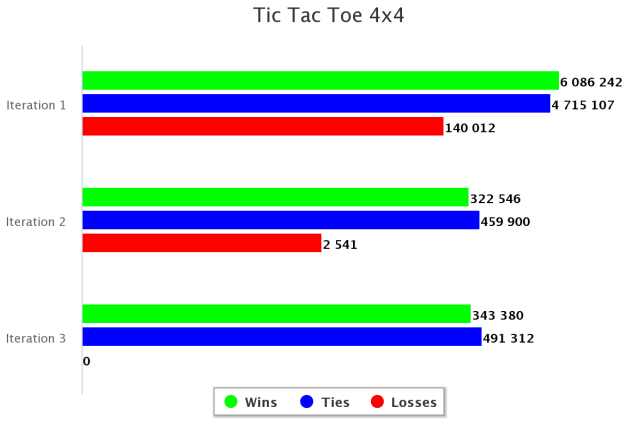

Fig. 4. Tic Tac Toe 4×4

performed using the optimal policy returned by the deep Q-learning method. We consider different deep RL algorithms like A2C, PPO, and DQN. However, only A2C could converge. In each iteration, we train the deep neural network for 10^6 steps and then extract policy for model improvement. Note that deep RL methods are approximation methods and we could not achieve a zero-loss policy due to the inherent approximation bias.

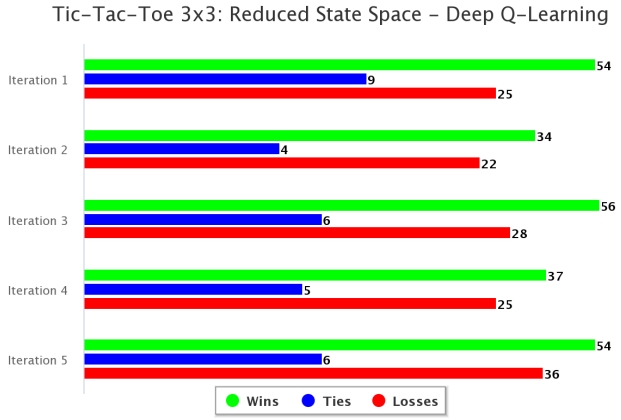

Fig. 3. Tic Tac Toe 3×3 - Deep Q-Learning

3) Tic-Tac-Toe 4×4: The grid dimension is 4×4 and the players have to complete a row, column, or diagonal to win. The game complexity leaps multi-fold compared to its 3×3 counterpart with $|S| = 9722012$ and $|A| = 17$. Just like for tic-tac-toe 3×3, our algorithm performs well for 4×4. The number of possible games is as huge as eleven million. After the first model improvement iteration, the losses were close to 1.4×10^5. But, as more iterations are performed, the losses decrease drastically to 0.

B. First/Second Mover Advantage

Games such as domineering, cram, and dots & boxes possess the First/Second mover advantage, where the player who starts first (similarly second) has an edge for winning in case of first (similarly second) mover advantage. This means that when two expert players play against each other, the outcome of the game depends entirely on who made the first (or second) move. However, the first mover advantage does not make the game outcome predetermined. Even though a game possesses the first-mover advantage, the player still has to make intelligent moves to win, however, there is always a way to win. So, learning is still required to deduce the way to win if the agent starts the game. Because of the first mover advantage, no matter how much the agent is trained, an expert opponent can still win if he starts first. So, to evaluate our algorithm against these kinds of games, we consider two kinds of comparisons for an unbiased performance evaluation. In the first one, the policy is tested for all the possible games in which the agent makes the first move, and in the second one, the policy is tested against all possible games in which the opponent makes the first move.

C. Domineering

Domineering game is a perfect information board game played over a 2D grid of dots. One of the players links only vertically adjacent dots while the other links only the horizontal ones. A dot cannot be linked more than once. The first player with no moves left loses. Since the game will eventually end with one of the players unable to move, the game never ends in a draw.

1) Domineering 4×4: Figures 5 and 6 show the results for two scenarios respectively, one when the agent starts the game, and the other when the opponent starts the game. With just two model improvement iterations, our algorithm can find a no-loss strategy for all those games in which the agent makes the first move. In the latter scenario, the policy was tested against all those games in which the opponent makes the first move. Because of the first mover advantage, there exists no policy for which the number of losses is zero. But, the algorithm attempts to reduce the number of possible games in which the opponent wins. After the first iteration, the number of losses is 9, but it got reduced to 6 in the next iteration and got stagnant at 6 in further iterations. Here, the game complexity is as follows: $|S| = 3121$ and $|A| = 13$.

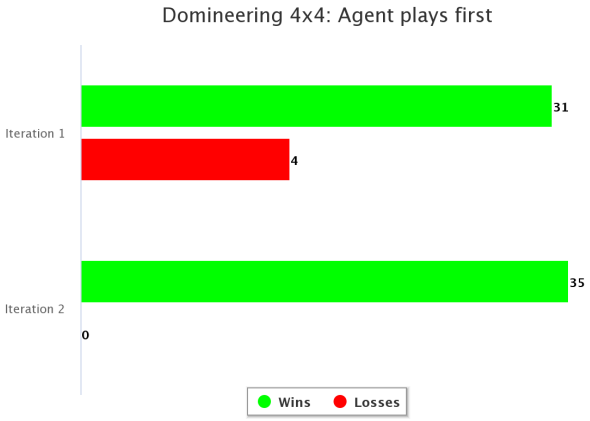

Fig. 5. Domineering 4×4 : Agent plays first.

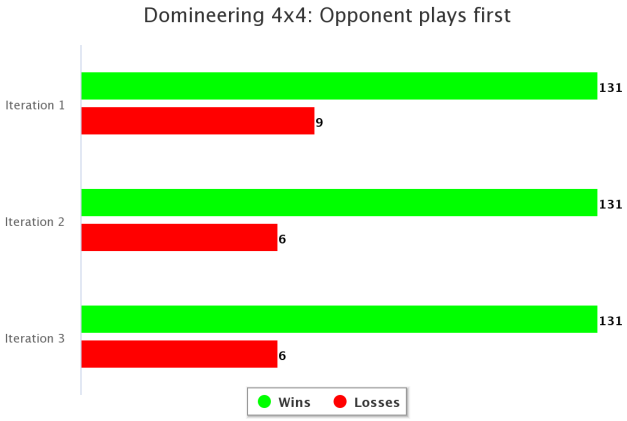

Fig. 6. Domineering 4×4 : Adversary plays first.

2) Domineering 5×5*:* Unlike, domineering 4×4, domineering 5×5 has a second-mover advantage. This means that if a player does not make the first move and plays intelligently, then he/she always wins. Our algorithm could find the optimal policy for all those games, where the opponent starts the game in only two model improvement steps. Also for those games in which the agent makes the first move, our algorithm minimizes the number of losses. Here, the game complexity is as follows: $|S| = 723997$ and $|A| = 21$.

D. Cram

Cram is a two-player abstract strategy board game similar to domineering. The game consists of a 2D grid of dots. The players take turns linking two adjacent dots with either a vertical or a horizontal line. A dot cannot be linked more than once. The player with no moves left loses. Since there is always a player with no moves left, the game never ends in a draw. The only difference between this game and domineering is that in domineering a player will have to place either a vertical or horizontal edge throughout the game, whereas in cram, a player can choose to place any of the edges at each step of the game. Here the game complexity of a 4×4 cram game is as follows: $|S| = 5700$, $|A| = 25$.

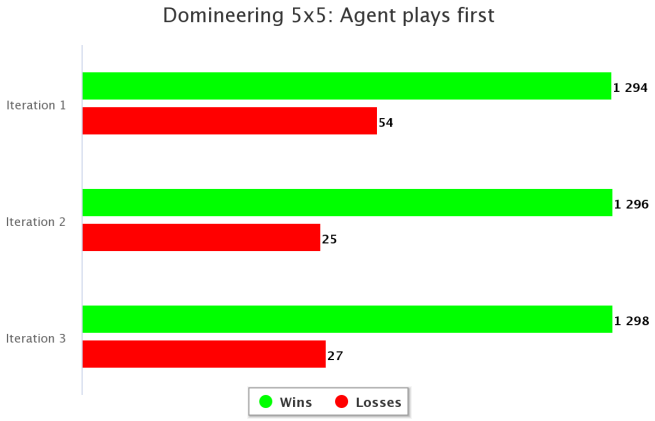

Fig. 7. Domineering 5×5 : Agent plays first.

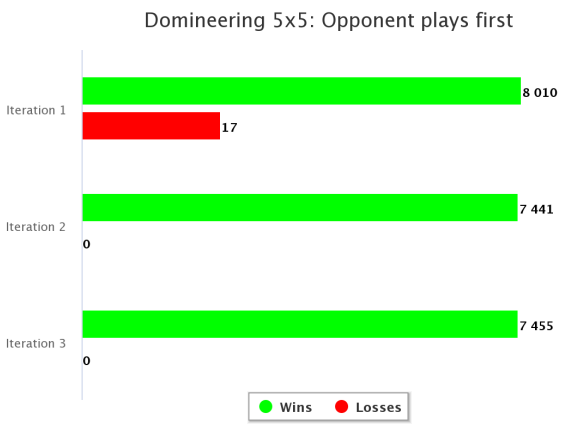

Fig. 8. Domineering 5×5 : Adversary plays first.

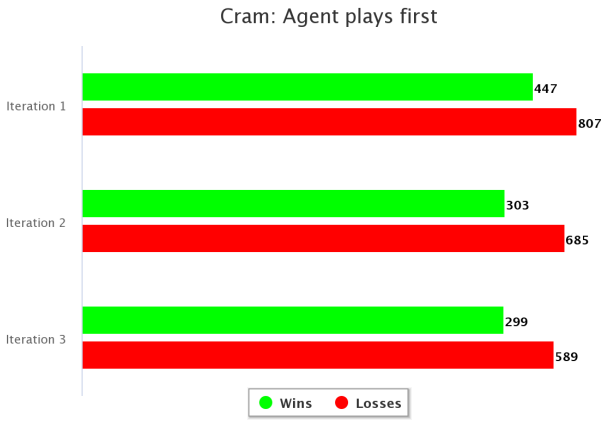

Fig. 9. Cram 4×4: Agent plays first.

Cram has a second-mover advantage. But unlike domineering, the number of losses is much higher than the number of wins when the agent plays first. This suggests that cram has a very explicit second-mover advantage. Here also, our algorithm minimizes the number of losses. Further, when the opponent makes the first move, our algorithm can find the optimal policy in only three model improvement iterations.

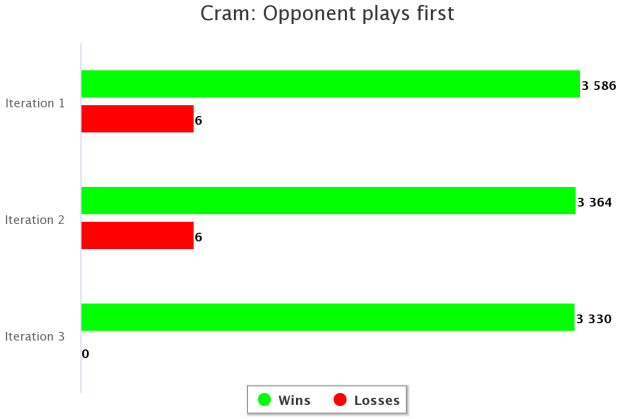

Fig. 10. Cram 4×4: Adversary plays first.

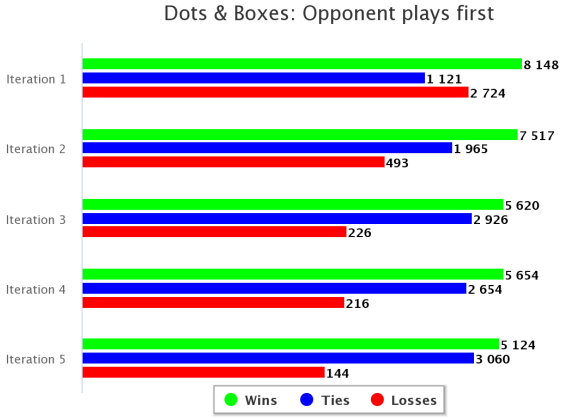

Fig. 12. 3×3 Dots and Boxes: Opponent plays first

E. Dots and Boxes

The game of dots and boxes is also played on a 2D grid of dots. Players join two vertically or horizontally adjacent dots, creating squares. If a player completes the fourth side of a square, it gets assigned to them and also they get the next turn. In the end, the player with the maximum number of squares assigned wins. Even though the rules of the game are simple, the game dynamics are complex and require some skillful play on the part of the players. The game complexity of 3×3 version is as follows: $|S| = 5286$, $|A| = 13$.

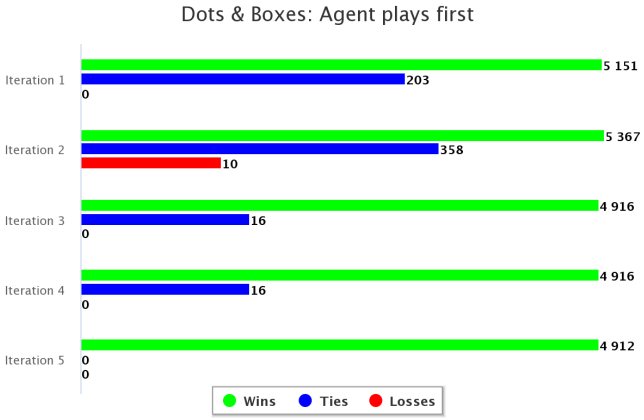

Fig. 11. 3×3 Dots and Boxes: Agent plays first

Dots and Boxes have a first-mover advantage. It is interesting to note that, it took five model improvement iterations to achieve the optimal policy. By the end of three model improvement iterations, the number of losses is reduced to zero, but there are still some ties. In the next two iterations, the algorithm could reduce the number of ties to zero as well. So, if the agent makes the first move, it is guaranteed that the agent will certainly win, there is not even a chance for the opponent to tie the game.

V. CONCLUSION

In this paper, we propose a self-play-based approach for generating efficient and competent no-loss policies for moderate state-space, symmetric, and adversarial board games. Our algorithm could deduce no-loss policies that require a minimum number of moves for different board games of varying complexities. By eliminating the need for input game episodes and relying on self-play, our algorithm offers scalability and adaptability for various board games, making it a promising approach for developing game-playing agents.

REFERENCES

[1] C. E. Shannon, "Xxii. programming a computer for playing chess," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 41, no. 314, pp. 256–275, 1950.

[2] D. H. Widyantoro and Y. G. Vembrina, "Learning to play tic-tac-toe," in *2009 International Conference on Electrical Engineering and Informatics*, vol. 01, 2009, pp. 276–280.

[3] M. van de Steeg, M. M. Drugan, and M. Wiering, "Temporal difference learning for the game tic-tac-toe 3d: Applying structure to neural networks," in *2015 IEEE Symposium Series on Computational Intelligence*, 2015, pp. 564–570.

[4] I. E. Skoulakis and M. G. Lagoudakis, "Efficient reinforcement learning in adversarial games," in *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*, vol. 1, 2012, pp. 704–711.

[5] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine learning proceedings 1994*. Elsevier, 1994, pp. 157–163.

[6] E. Soedarmadji, "Decentralized decision making in the game of tic-tac-toe," in *2006 IEEE Symposium on Computational Intelligence and Games*. IEEE, 2006, pp. 34–38.

[7] A. Bhatt, P. Varshney, and K. Deb, "In search of no-loss strategies for the game of tic-tac-toe using a customized genetic algorithm," in *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, 2008, pp. 889–896.

[8] G. Hochmuth, "On the genetic evolution of a perfect tic-tac-toe strategy," *Genetic Algorithms and Genetic Programming at Stanford*, pp. 75–82, 2003.

[9] J. V. Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*. Princeton, NJ, USA: Princeton University Press, 1944.

[10] D. E. Knuth and R. W. Moore, "An analysis of alpha-beta pruning," *Artificial Intelligence*, vol. 6, no. 4, pp. 293–326, 1975.

[11] S. Sriram, R. Vijayarangan, S. Raghuraman, and X. Yuan, "Implementing a no-loss state in the game of tic-tac-toe using a customized decision tree algorithm," in *2009 International Conference on Information and Automation*. IEEE, 2009, pp. 1211–1216.

[12] D. Bertsekas, *Reinforcement learning and optimal control*. Athena Scientific, 2019.

[13] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[14] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.