Playstyle Generation for Geister With Genetic Algorithm and Clustering

Keisuke Tomoda and Koji Hasebe
Department of Computer Science, University of Tsukuba, Japan

Keywords: Imperfect Information Game, Genetic Algorithm, Playstyle, Geister.

Abstract: Studies on game-playing agents have made various attempts to develop agents with characteristic playstyle. Most of these studies either generated agents with predetermined playstyles or simultaneously generated different playstyles without defining a specific playstyle for single-player complete information games. However, the generation of agents with different playstyles for multi-player imperfect information games has not been thoroughly investigated. Therefore, in this study, we have proposed an automatic playstyle generation method for a two-player imperfect information game called Geister. The basic idea is to use a genetic algorithm to optimize agents whose genes represent parameters that determine the manner of guessing hidden information. By clustering the genes with high fitness, obtained using this process, agents with different playstyles are generated simultaneously. From the results of the experiments, our proposed method generated five different playstyles with cyclic dominance relationships.

1 INTRODUCTION

Research on game-playing agents is developing rapidly, with the performance of some agents, such as AlphaZero (Silver et al., 2018) and Libratus (Brown and Sandholm, 2018), surpassing professional human players. In addition, there are significant interests in developing these agents to achieve broader objectives, not just their strengths (Lara-Cabrera et al., 2015; Hoover et al., 2019; Fan et al., 2019). Among these, there exist studies on the generation of agents with characteristic playstyles (Tychsen and Canossa, 2008; Tampuu et al., 2017; Ishii et al., 2018). A playstyle refers to a set of characteristic behaviors of the player. However, most of these studies predefined the playstyles they wanted to generate, which makes it difficult to identify playstyles that cannot be assumed in advance.

To address this issue, Iwasaki et al. (Iwasaki and Hasebe, 2021) proposed a framework called C-NEAT to generate different playstyles simultaneously without predefining any styles. This framework consisted of an evolutionary computation called NeuroEvlution of Augmenting Topologies (NEAT) (Stanley and Miikkulainen, 2002) and clustering, which optimizes gameplay while classifying the characteristic behavior of the agents by clustering. However, that study focused on a roguelike game, which is a complete information game for a single player. Generation of agents with different playstyles for multiplayer imperfect information games has not yet been thoroughly investigated.

In this study, we have proposed an automatic playstyle generation framework for a two-player imperfect information game. Here, we focus on Geister, a chess like game, in which the colors of the pieces (red and blue) are hidden from the opponent. In this game, the optimal action varies depending on the color of the the opponent's pieces; therefore, it is important to advance the game while guessing them. However, it is not possible to perfectly identify the color of the piece using only the revealed information, and the effective guessing method changes depending on the opponent's strategy. Based on the above observations, we focus on the playstyles that determine the guessing manner.

To simultaneously generate the playstyles defined above, we propose a framework using genetic algorithms and clustering based on the concept of C-NEAT. Specifically, many agents are generated that have parameters of a function for guessing the color of the opponent's piece and, while they are played against each other, agents with a high winning rate (i.e., high fitness) are repeatedly retained. After natural selection, the set of genes possessed by the elites from all generations are clustered. Then, for each cluster, the average value of the genes was evaluated, resulting in different playstyles.

Tomoda, K. and Hasebe, K.
Playstyle Generation for Geister With Genetic Algorithm and Clustering.
DOI: 10.5220/0011801400003393
In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART 2023) - Volume 3*, pages 916-922
ISBN: 978-989-758-623-1; ISSN: 2184-433X

To demonstrate the effectiveness of the proposed method, an experiment was conducted using a simplified Geister with 4×5 board. We observed that five different playstyles with dominant cyclic relationships were generated.

The remainder of this study is organized as follows. Section 2 presents the related work. Section 3 provides an overview of the rules of Geister. Section 4 describes the proposed framework for generating playstyles for a Geister. Section 5 presents the experimental results. Finally, Section 6 concludes the paper and presents future work.

2 RELATED WORK

Significant research has been conducted to develop agents capable of playing games with characteristic playstyles. One of the earliest studies on playstyles in games was conducted by Tychsen et al. (Tychsen and Canossa, 2008). The authors defined a player's playstyle to be a set of their characteristic behaviors. Based on this definition, they focused on the game, Hitman: Blood Money, and identified different playstyles by analyzing various playlogs.

Ishii et al. (Ishii et al., 2018) developed agents with different characteristic playstyles (personas) for the fighting game, FightingICE. Their fundamental idea was to determine the agent's actions via Puppet-Master Monte Carlo Tree Search (MCTS), which is an extension of general MCTS, using evaluation functions pertaining to one of two playstyles: short-range attack and long-range attack.

Tampuu et al. (Tampuu et al., 2017) generated agents with two playstyles, cooperative and adversarial, for the video game called Pong. They demonstrated that both cooperative and adversarial gameplay can be achieved by agents based on Deep-Q-Network (Mnih et al., 2013) using a unified reward scheme.

In all of the aforementioned studies, playstyles were predefined based on the authors' expectations. Therefore, generation of playstyles not defined in advance is difficult using these methods. In contrast we attempt to generate multiple playstyles automatically without defining them in advance.

Studies on similar topics have been conducted by Iwasaki et al. (Iwasaki and Hasebe, 2021; Iwasaki and Hasebe, 2022). They generated multiple playstyles for the roguelike game, MiniDungeons, using evolutionary computation and clustering. The game was a single-player complete-information game so the environment of the game visible to the player was fixed. In contrast, we focus on applying the framework proposed by Iwasaki et al. to a two-player imperfect-information game. This challenge raises several new issues, such as changes in fitness depending on the opponent even with the same pointing, and appropriate handling of imperfect information.

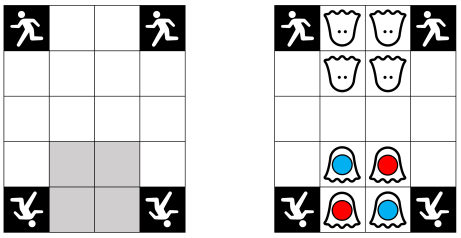

Figure 1: Board of Geister (left) and an example of the initial board setting from the viewpoint of the player on the bottom (right).

3 OVERVIEW OF GEISTER

Geister is a two-player chess-like game in which players take turns moving their pieces on a 6×6 board. However, in this study, in order to reduce computational time, the size of the board is reduced to 4×5, as depicted in Figure 1, without loss of generality.

On the board, the left- and right-most cells (indicated by icons in the left picture in Figure 1) of the farthest row of the board, as seen by each player, are special cells called "exits." Each player initially controls four pieces, two of which are blue and the other two are red. The color of each piece is only visible to its owner. Before starting the game, each player places two of their four pieces in the 2×2 region at the center of the two rows closest to them (marked in gray in the left picture in Figure 1 for the player on the bottom). An example of an initial board configuration from the perspective of the player on the bottom is depicted in the right picture in Figure 1.

During the game, the players take turns moving their pieces. During each player's turn, they can move any one piece to a vertically or horizontally adjacent square, unless it takes the piece outside the board or onto a square already occupied by another of their pieces. If a piece is moved onto a square occupied by an opponent's piece, the latter is removed from the game and may not be reused (henceforth referred to as being captured). Further, each player can make a special move called "escape" on the following turn if one of their pieces is on one of their exits during the current turn.

The first player to achieve any of the following conditions wins.

1. Have one of their blue pieces escape.

2. Capture all of their opponent's blue pieces.
3. Have their opponent capture all of their red pieces.

4 FRAMEWORK FOR PLAYSTYLE GENERATION

4.1 Overview

In Geister, the optimal move on a given board state depends on the colors of the opponent's pieces, thus guessing them is integral to one's strategy. However, since it is not possible to deduce the colors of the opponent's pieces based only on the revealed information, there is no single correct method of guessing. Thus, the usefulness of guessing methods is likely to depend on the opponent's guessing strategy. Therefore, optimization of play is not expected induce convergence to a single guessing method, but, instead, yield multiple methods of guessing. Based on the above considerations, in this study, we define playstyle in terms of the guessing method used to estimate the colors of the opponent's pieces.

A more detailed definition of the playstyle is presented below. The agent determines its move on each turn by evaluating the likelihoods of the opponent's pieces being of certain colors and searching for the best move on the probabilistic board. The former is determined using a special function (called a guessing function), which involves some parameters, that takes into account the specific movements of enemy pieces on the board. The latter is performed using the Expectimax algorithm (Michie, 1966), which is a variant of the Min-Max search algorithm with incorporated probabilistic state transitions. The evaluation function for Expectimax is taken to be one whose usefulness has been confirmed in preliminary experiments. Therefore, the moves of the agent are determined based solely on the parameters of the guessing function and the search algorithm.

We obtain the desired playstyles using genetic algorithm and clustering by considering the parameters of the guessing function used by the agent as genes. The detailed procedure is given by the following algorithm. (Figure 2 illustrates the procedure, where the numbers correspond to the step numbers presented below.)

1. Generate multiple agents with random guessing parameters.
2. Match the generated agents with other agents sequentially and measure the corresponding winning rates, which are considered to be measures of fitness.

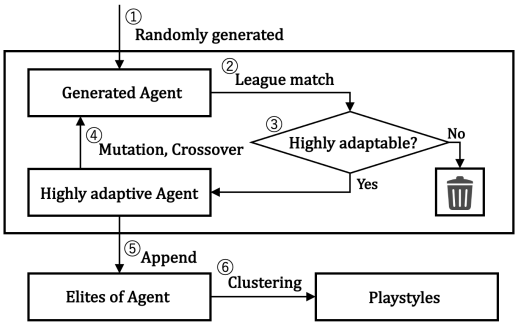

Figure 2: Procedure for obtaining playstyles.

3. Retain agents with high fitness and remove others.
4. Create a new generation of agents based on the retained agents via mutation and crossover.
5. Repeat steps 2-4 and save all agents with high fitness in each generation.
6. Cluster the parameters of all saved agents as feature values.
7. Determine individual playstyles based on the average values of all parameters in each cluster.

4.2 Guessing Colors of Opponent's Pieces

In Geister, the colors of one's opponent's pieces cannot be guesses solely by considering one's own perspective. That is, the opponent's estimation of the colors of one's own pieces must also be accounted for. Therefore, the proposed agent formulates one guess, $e(p_i^{opp})$, for the color of each piece, p_i^{opp} $(1 \leq i \leq 4)$, of the opponent, and another, $e(p_i^{pro})$, for the opponent's guess for the color of each piece, p_i^{pro} $(1 \leq i \leq 4)$, owned by themselves. The guessed values lie in the range, $[0,1]$, with values close to 1 corresponding to guesses of blue with high conviction, and those close to 0 corresponding to guesses of red with high conviction. Each initial value is set to 0.5.

Guessed values are updated after the movement of pieces during each turn. In particular, each agent involves three parameters g_1, g_2, and g_3, with values in the range, $[-0.5, 0.5]$, which correspond to the following movements.

- g_1: Move to a cell adjacent to an enemy piece.
- g_2: Move away from an adjacent enemy piece.
- g_3: Move closer to the exit.

These three actions are related to Geister's victory conditions, and various playstyles can be defined by combining different configurations of the parameters.

During the updating process, the availability of actions corresponding to these parameters are verified for each move. In case of availability, the value of the corresponding parameter is added to the guess value of the piece moved. For example, if the opponent's move on their t-th turn is to move the piece, p^{opp}, adjacent to a piece of the agent, the guess value, $e_t(p^{opp})$, is updated using the following formula:

$$e_t(p^{opp}) := g_1 + e_{t-1}(p^{opp}).$$

Then, the guess values are adjusted to satisfy: $\Sigma_{i=1}^{4} e(p_i^s) = 2$ (for $s \in \{opp, pro\}$) and $\forall i e_t(p^s) \in [0,1]$.

However, in the exceptional case in which a piece adjacent to an exit is not moved to the exit, the corresponding guess value is set to 0 and is not changed thereafter. This is because, if the color of the piece were blue, the only rational move would have been to move it to the exit.

4.3 Decision of Moves

Based on the guess values, the agent determines the optimal move during each turn using an extension of the Expectimax algorithm. Expectimax is an exhaustive, depth-first game tree search algorithm, which is often used to determine moves in imperfect-information games. Both Expectimax and its extension are described below.

The nodes in the search tree of the Min-Max algorithm are classified into two types: Max nodes and Min nodes. Nodes of the former type represent the player's turn and transition to a child node with the maximum evaluation value, while those of the latter type represent the opponent's turn and transition to a child node with the minimum evaluation value. In addition, in Expectimax, a chance node is inserted between Min and Max nodes when the transition destination node changes depending on the probability. The evaluation value of a chance node is given by a weighted sum of evaluation values, with the transition probabilities of its child nodes as coefficients. In Expectimax, the evaluation value, $V(n)$, of a node, n, is calculated using the following formula.

$$V(n) = \sum_{c_i \in C(n)} P(n, c_i) \times V(c_i),$$

where $C(n)$ represents the set of child nodes of node n, $P(n,c)$ represents the probability of reaching node c from node n, and $V(c)$ represents the evaluation value for node c.

We now consider applying Expectimax to Geister. If a move reveals the color of an opponent's piece (e.g., when an opponent's piece is captured), the state after the move transitions stochastically depending on the likelihood of the color of the opponent's piece. Thus, a chance node is inserted after such a move.

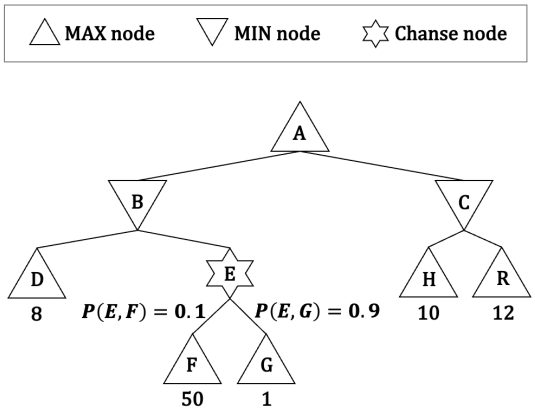

Figure 3: Expectimax game tree.

As explained previously, Expectimax can be used to determine moves in this way. However, better moves can be found by considering the information regarding the colors of one's own pieces. For example, let us consider the search tree depicted in Figure 3. If the color of the piece, say p, is red and the opponent chooses a move that induces a transition from node B to node E, then the transition node is known to be F. At this time, from the opponent's perspective, $V(E) = 5.9$, as they do not know the color of p. However, from the player's perspective, $V(E) = V(F) = 50$, as the color of p is known to be red. Therefore, if the opponent chooses a move to transition to node E from node B, it is possible to transition to a state where it is better to choose the move to transition to node B from node A.

In order to utilize the aforementioned information asymmetry between the proponent and the opponent during optimal move search, we extend Expectimax by incorporating the real value, which is the evaluation value from one's own perspective, and the pretense value, which is the evaluation value from the opponent's perspective.

In the extended Expectimax, nodes are selected to increase the real value at the MAX node and reduce the pretense value at the MIN node. If the transition destination at a chance node is known, its real value becomes the real value of the transition destination node. The pretense value of the chance node is always a weighted sum of the pretense values, with the transition probabilities of the child nodes of the chance node as coefficients.

Blue

10	5	5	10
5	3	3	5
3	2	2	3
2	1	1	2
1	0	0	1

Red

3	5	5	3
5	10	10	5
3	5	5	3
2	3	3	2
1	2	2	1

Figure 4: Evaluation tables for blue (left) and red (right) pieces.

4.4 Genetic Manipulation and Clustering

In our framework, genetic manipulation is performed based on mutation and crossovers, as in several existing genetic algorithms. As depicted in Figure 2, in Step 5 of the genetic algorithm, individuals with high fitness are saved. Once a sufficient number of agents with high fitness have been generated, the parameters of guessing functions of all stored agents are clustered using the k-means method. Further, the average value of the parameters of all agents within each cluster is representative of the playstyle corresponding to that cluster.

5 EXPERIMENTS

5.1 Parameter Configuration

In this research, the following experiments are performed to verify that the proposed framework actually generates multiple playstyles in Geister.

5.1.1 Optimal Move Search

In our experiments, the depth of optimal move search is set to 3. The evaluation function used in the search is defined as the sum of values assigned to each piece by its position. In particular, values are assigned to blue and red pieces as depicted in Figure 4. As explained previously, in Geister, the first player to move a blue piece to the exit wins. Therefore, we assume that high proximity between the blue pieces and the exits is desirable. Further, we assume that the value of a red piece is high when it is positioned to protect a blue piece near the exit without blocking its course to it. The table depicted in Figure 4 lists the highest winning rates in matches conducted during preliminary experiments using various similar tables.

On the other hand, we assume low proximity between the opponent's pieces and the exits to be desirable. Therefore, during the evaluation of the opponent's pieces, we used the inverted version of Figure 4, with negative values weighted by the guess value.

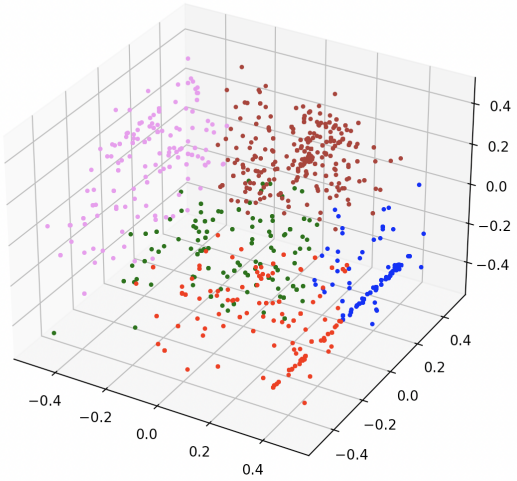

Figure 5: Clustering result.

5.1.2 Agent Generation

During the execution of genetic algorithm, the number of agents in each generation is taken to be 100, and 30 agents with high fitness are retained in each subsequent generation. During the generation of new agents based on agents with high fitness, mutation and crossover are performed with probabilities of 0.6 and 0.4, respectively. Moreover, generation of excessively weak agents is avoided by allowing each generated agents to play against an agent taking random actions 100 times in advance, and deleting those with win rates less than 90%.

5.1.3 Clustering

The parameters of 1,500 experimentally obtained agents over 50 generations are subsequently classified via clustering. A non-hierarchical k-means method is adopted as the clustering algorithm. Preliminary experiments varying the number of clusters confirmed that only 3-5 playstyles with a high winning rate were generated even when the number of clusters was increased, so the number of clusters was set to 5 in this study. This process yields five agents with different playstyles. We also determine the compatibilities between these playstyles via round-robin matches.

5.2 Results

Figure 5 depicts the three-dimensional plot obtained by color-coding the 30 individuals with highest fitness in each generation via clustering. Figure 6 illus-

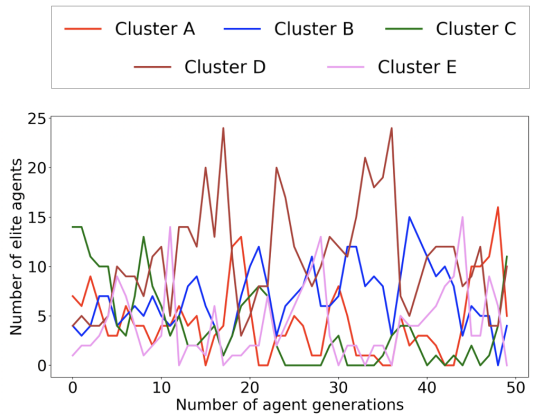

Figure 6: Change in the ratio of the number of agents for each playstyle.

Table 1: Parameters of cluster representative.

	Estimated Parameters		
	Adjacent	Distancing	GoalDirected
Cluster A	0.2685	-0.2646	-0.2592
Cluster B	0.3847	0.2832	-0.3515
Cluster C	-0.0794	0.1240	-0.3194
Cluster D	0.1012	0.2411	0.1975
Cluster E	-0.3819	0.0200	0.1309

trates the variation in the ratios of different playstyles in each generation.

The representative parameters of the generated clusters are presented in Table 1. In this table, a positive (negative, resp.) value is adopted while determining that a piece that has made a move corresponding to each parameter is a blue (red, resp.) piece.

Further, Table 2 presents the results of matches performed using the representative parameters of each generated cluster. The table lists the winning rates of all possible initial placements after 72 matches in aggregate, with the first and second player roles switched once each. Figure 7 depicts a graphical representation of the dominance relationships between the identified playstyles. In this figure, the nodes represent the generated playstyles and an edge from X to Y indicates that X wins (dominates) against Y in 60% or more instances.

Based on Table 1, the characteristics of the playstyles can be interpreted as follows.

Cluster A: A piece that is adjacent to another of the same player is considered to be blue, and one that has moved away from another of the same player is considered to be red.

Cluster B: A piece that is adjacent or has moved away from another of the same player is considered to be blue, and one that is close to an exit is considered to be red.

Table 2: Match results between playstyles.

	Opponent's playstyle				
	A	B	C	D	E
A		0.375	0.312	0.625	0.813
B	0.625		0.437	0.406	0.563
C	0.688	0.563		0.250	0.688
D	0.375	0.594	0.750		0.625
E	0.187	0.437	0.312	0.375	

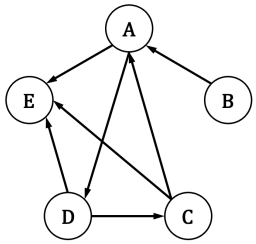

Figure 7: Dominant relations between playstyles.

Cluster C: A piece approaching an exit is considered to be red.

Cluster D: A piece that has moved away from another of the same player and a piece that is close to an exit are considered to be blue.

Cluster E: A piece that is adjacent to another of the same player is considered to be blue.

Based on the above results, the following conclusions can be drawn regarding the pairwise compatibilities of these playstyles. In general, agents are more likely to capture pieces that they consider to be blue, and less likely to capture pieces that they consider to be red. Therefore, Playstyle A prefers to play slowly, only capturing adjacent pieces instead of chasing pieces moving away. Playstyles B and C prefer to aim for the goal early, thinking that their pieces will not be captured even if they approach the goal. Finally, Playstyles D and E prefer to play slowly, without aiming for the goal.

As depicted in Figure 7, a cyclic dominance relationship is observed. More specifically, a performs favorably against C, C performs favorably against D, and D performs favorably against A. Further, as illustrated in Figure 6, for no playstyle does the number of individuals monotonically increase with each generation. The number of individuals following a playstyle with a high winning rate is observed to trend upwards. Based on these results, we can conclude that no playstyle is always optimal when playstyles are defined based on the guessing strategy utilized. Also,

the effective playstyles may change dynamically depending on the playstyle of the majority of the players in each generation.

6 CONCLUSIONS AND FUTURE WORK

In this study, we proposed a framework using genetic algorithms and clustering to generate multiple playstyles for Geister, a two-player imperfect information game. Specifically, many agents with genes are generated as the parameters of a function to guess the color of the opponent's piece. While these are played against each other, agents with high fitness are obtained. Furthermore, by clustering the genes possessed by the elites of all generations, we can obtain our target multiple playstyles. As a result of the experiment, we observed that five playstyles with circular dominance relationships were generated.

In this experiment, we have considered playstyles that focused only on the guessing manner; however, in the future, we would like to generate more diverse playstyles determined by the border evaluation functions. In addition, based on methods such as reinforcement learning, we investigated an improved framework that allowed agents to play better with characteristic playstyles.

REFERENCES

Brown, N. and Sandholm, T. (2018). Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424.

Fan, T., Shi, Y., Li, W., and Ikeda, K. (2019). Position control and production of various strategies for deep learning go programs. In *2019 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pages 1–6. IEEE.

Hoover, A. K., Togelius, J., Lee, S., and de Mesentier Silva, F. (2019). The many ai challenges of hearthstone. *KI - Künstliche Intelligenz*, 34:33–43.

Ishii, R., Ito, S., Ishihara, M., Harada, T., and Thawonmas, R. (2018). Monte-carlo tree search implementation of fighting game ais having personas. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE.

Iwasaki, Y. and Hasebe, K. (2021). Identifying playstyles in games with neat and clustering. In *2021 IEEE Conference on Games (CoG)*, pages 1–4. IEEE.

Iwasaki, Y. and Hasebe, K. (2022). A framework for generating playstyles of game ai with clustering of play logs. In *ICAART (3)*, pages 605–612.

Lara-Cabrera, R., Nogueira-Collazo, M., Cotta, C., and Fernández-Leiva, A. (2015). Game artificial intelligence: Challenges for the scientific community. *CEUR Workshop Proceedings*, 1394:1–12.

Michie, D. (1966). Game-playing and game-learning automata. In *Advances in programming and non-numerical computation*, pages 183–200. Elsevier.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144.

Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127.

Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J., and Vicente, R. (2017). Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395.

Tychsen, A. and Canossa, A. (2008). Defining personas in games using metrics. In *Proceedings of the 2008 conference on future play: Research, play, share*, pages 73–80.