Expert Iteration for Risk

Lucas Gnecco Heredia and Tristan Cazenave

LAMSADE, Université Paris-Dauphine, PSL, CNRS, France
`Tristan.Cazenave@dauphine.psl.eu`

Abstract. Risk is a complex strategy game that may be easier to understand for humans than chess but harder to deal with for computers. The main reasons are the stochastic nature of battles and the different decisions that must be coordinated during the players turn. Our goal is to create an artificial intelligence able to play the game without human knowledge using the *Expert Iteration* [1] framework. This algorithm is very similar to the *AlphaZero* algorithm proposed in [23] that achieved, *tabula rasa*, super-human level in Go using only self-play and that even surpassed the previous versions AlphaGo [22] and AlphaGoZero [24]. We use graph neural networks [29], [21], [14], [12] to learn the policies for the different decisions and the value estimation. Experiments on synthetic boards show that the player is able to learn common sense policies for some game phases like country drafting and initial army placing, and plays better than random on small maps.

1 Introduction

The game of Risk might be somehow simpler for humans compared to chess or Go. It has a much more complex game flow, but it requires less experience to be able to play at a decent level. For Risk, once the game rules and objectives are understood, human players can find out common sense strategies that work fairly well. It seems at a first glance that the game of Risk is simpler than chess for humans, but as we will see, it presents a lot of challenges when it comes to computer play.

To begin with, in Risk each turn consist on different phases that involve multiple decisions that should be coordinated. Moreover, the attack phase is stochastic because the result of the attack depends on a dice roll, introducing chance nodes to the game tree [17]. In terms of the number of players, it can be played with two to six players, so by nature it falls out of the usual two-player zero sum game category, opening the door to questions about coalitions and the impact of other players on the outcome of the game [18], [26]. It has imperfect information because of the game cards that players can use to trade armies. Under traditional rules, trading cards will augment the number of armies for the next trade, making it less obvious to decide when to trade.

The present work aimed to create a Risk player able to learn *tabula rasa*, meaning without human knowledge and just using self-play. The player and training methods follow the *Expert iteration* framework [1], closely related to *AlphaZero* [23]. For the policy and value neural networks we used graph neural networks [29], [21], [14], [12], given the fact that the Risk board can be naturally represented as a directed unweighted graph.

Section 2 covers the basic concepts needed to understand the whole approach. Section 3 is a small survey on previous attempts to create a Risk AI. Section 4 presents the design of the neural-MCTS player and implementation details. Section 5 explains the experimental setup including the modifications to the game rules made to simplify the problem. Results on two maps are showed and discussed. The code related to the project is available as a Github repository[1].

2 Preliminaries

2.1 Monte Carlo Tree Search

Monte Carlo Tree Search is a search method that iteratively builds the search tree relying on random sampling to estimate the value of actions. The general algorithm consist on 4 phases: *selection*, *expansion*, *simulation* and *backpropagation*. The idea is that at each iteration the tree is traversed using a *tree policy* for choosing moves. When a leaf is found usually the tree is expanded by adding its children. From there a *default policy* is used to play further moves (usually random) and end the episode. The results are then backpropagated through the traversed nodes [5].

The most known MCTS algorithm is probably Upper Confidence Tree (UCT) created in 2006 by Levente Kocsis and Csaba Szepesvári [13]. It makes use of the *UCB1* [3] bandit formula to choose moves along the tree descent. Rosin [20] later added prior knowledge to the formula to bias the initial choices towards moves suspected of being strong. This idea was called Predictor+UCB (PUCT) and it is the way in which the *AlphaZero* and *Expert Iteration* frameworks combine neural networks and MCTS. The idea is that neural networks can learn strong policies that can work as priors biasing the search towards promising nodes. They can also learn to evaluate game states which also improves the search process.

2.2 Expert Iteration

Expert iteration [1], [2] is an effective algorithm to learn a policy given by a tree search algorithm like Monte Carlo Tree Search (MCTS). The idea is that a deep neural network can approximate the policy provided by the MCTS and generalize it so that it can be used afterwards without performing the search, or it can be used to bias the search and get better results. At each iteration of the training process, the current neural network is used to guide the search of the MCTS and/or to evaluate game states at the leafs of the tree, resulting in a more efficient search and therefore a stronger policy. This new policy is learned again by the deep neural network and the process repeats.

3 Previous work

The first Risk AI players to the best of our knowledge date back to 2005, where Wolf [27] tried to create a linear evaluation function. Apart from the value function, he programmed different plans that represent common sense strategies and allowed his player to play accordingly. A similar approach was developed in 2013 by Lütolf [16].

[1] https://github.com/lucasgneccoh/pyLux

Another approach developed in 2005 was made by Olsson [19], [10] who proposed a multi-agent player that placed one agent in every country plus a central agent in charge of communication and coordination. Each country agent works like a node in the map graph, passing and receiving information from its neighbors to evaluate its own state. Then every agent participates in a global, coordinated decision. It is worth highlighting the importance it gives to the graph structure of the board and the message passing between countries, concepts highly related to graph neural networks.

The first attempt that used MCTS for playing Risk was done in 2010 by Gibson, et al. [9] where they used UCT only on the initial country drafting phase of the game. They concluded that the drafting phase was key to having better chances of winning the game, and that their UCT drafting algorithm was able to make an existing AI player improve considerably.

In 2020, two contributions were made using neural networks. The one developed by Carr [6] uses graph convolutional networks (GCN) [12] to take information from the board and return an evaluation. The second player proposed in 2020 by Blomqvist [4] followed the *AlphaZero* algorithm. This is very similar to what we intend doing, but the network architecture consists only on fully connected linear layers. One valuable result is that even if the learned policies are not remarkably strong, they improve the MCTS when included as priors, allowing to conclude that they indeed bias the search towards interesting moves.

4 Player design

4.1 General player design and architecture

The first thing to notice about the game of Risk is that the board can be naturally represented as a graph. This immediately suggests the use of a Graph Neural Network (GNN) [21] instead of more traditional networks. Moreover, having seen the importance of convolutional layers on the development of the Go players [22], we decided to use Graph Convolutional Networks [12].

Following the same line of thought presented by Olsson [19], we considered countries as the fundamental blocks for any reading of the game board as most of the information needed is stored at a country level. We considered only basic information as input, such as the percentage of armies in the country from the boards total, the owner of the country, the continent it makes part of, the bonus this continent has, etc. Just as in Go [22], more complex features could be created such as if the country is on the border of the continent or not, if it has an ally/enemy neighbor. Once this features are computed, each country yields a tensor that is fed to deep residual convolutional layers [14], [15] to create a hidden board representation. We were inspired by the increase in performance due to residual blocks and deeper models in the game of Go [7,24]. We used four deep GCN layers for the initial block and another four for each head.

One key idea we wanted to keep present in the network design was that for every action except for the card trade, a policy could be represented as a distribution over either the nodes or the edges of the graph. In our model, each head ends up with something similar to the input: one tensor for each node of the graph that can then be easily transformed into the desired policy. This makes the design flexible enough to adapt to any map and is similar to how the Go players were designed, reading as input a game board and giving as output a distribution that has the same shape [28,8]. Figure 1 shows a general diagram of the network design.

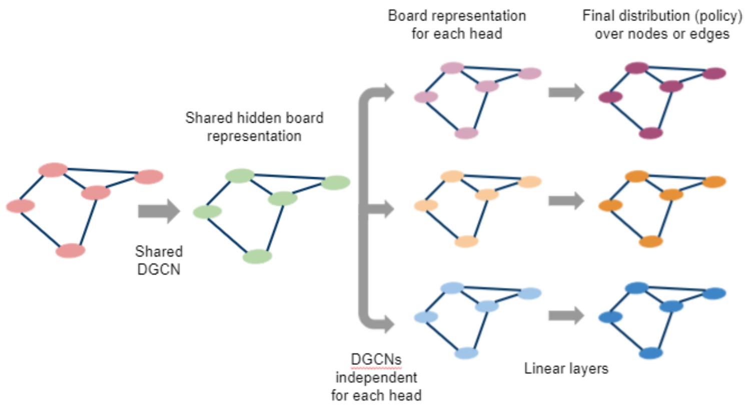
Board representation for each head
Final distribution (policy) over nodes or edges
Shared hidden board representation
Shared DGCN
DGCNs independent for each head
Linear layers

Fig. 1: General flow of the network

Note that the presented design is highly customizable either at the graph convolutional layers that build the hidden board representation, the final layers that take the output of the convolutions and turn it into distributions over the nodes or edges, or even at the creation of the node tensors where node embeddings can be considered. The final model has 5 heads: one policy head for each decision (pick country, place armies, attack, fortify) and one value head to estimate the value of each player.

4.2 Loss function and further parameters

The loss function used was the one detailed in [1]. It is composed of two terms related to the policy and value estimations. The target policy is the distribution induced by the visit counts of the MCTS at the root node. The target value is ideally a vector with 1 for the winning player and 0 for the rest, but for unfinished games we created an evaluation function that takes into account the number of armies a player would receive at the start of his turn. If P represents the neural network policy, z the target value vector and $\hat{z}$ the value estimation given by the network, the loss functions is given by

$$\mathcal{L} = -\sum_{a} \frac{N(s,a)}{N(s)} \cdot \ln P(a|s) - \sum_{i=1}^{6} z_i \ln\left[\hat{z}_i\right] + (1 - z_i) \ln\left[1 - \hat{z}_i\right]$$

where $N(s,a)$ is the visit count for action a from state s and $N(s)$ is the total visit count for state s.

Regarding the evaluation function used for unfinished games we normalized the values and made sure that the value of an unfinished board was always less than 0.9, keeping it far from 1 so that it was easier to differentiate between a good unfinished game and a winning one. For the tree policy, we used the PUCT bandit formula from *AlphaGo*:

$$U(s,a) = Q(s,a) + c_b \cdot P(a|s)\frac{\sqrt{N(s)}}{N(s,a)+1}$$

where $Q(s,a)$ is the mean action value estimated so far and c_b is a constant controlling the exploration.

For the optimizer, we chose the Adam optimizer [11] with a learning rate of 0.001 that decayed exponentially by a factor of 0.1 every 50 epochs. Regarding the *Expert Iteration* algorithm, we used 4000 simulations per expert labeling step due to the small size of the board. To speed up self play we chose moves by sampling the action proportional to the network policy without performing a search.

5 Results

In terms of gameplay, we made the following simplifications to the game:

1. Cards are only traded automatically for all players.
2. Attacks are always *till dead*, meaning that if an attack is chosen, it will only finish when either the attacker has only one army or when the defender is eliminated and the attacker conquers the country. This was also made for fortifications.
3. The choice of how many armies to move to a conquered country was also eliminated. By default, all possible armies are moved.
4. When placing armies, only one country is chosen and all armies are placed on it.

This simplifications were made to reduce the game complexity and test our first prototype on more basic game strategy.

5.1 Hexagonal map

The hexagonal map has 6 countries grouped into three continents: *Red*, *Yellow* and *Blue*. The bonuses for each continent were designed so that their value is clearly different: *Red* has bonus 1, *Yellow* has bonus 4 and *Blue* has bonus 9. With this said, picking countries becomes very straightforward as the player must try to get *Blue* or at least stop the opponents from getting it. In a simple two player game, the optimal strategy for both players would end up with each player controlling one country of each continent so that the starting player does not have any bonus at his first turn. Moreover, if given the chance, a player should try to secure the *Blue* continent.

We started the evaluation of the models by looking at the evolution of the probabilities they assigned to each country when having to choose first. Results in Figure 3 show that the model is indeed learning that the best choices are situated in the *Blue* continent, followed by the *Yellow* one. It is interesting to see that the model is not able to decide

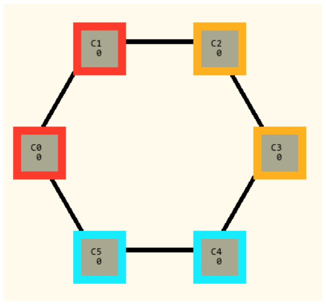

Fig. 2: Hex map. Country background colors are related to continents. Country names are $C0$ and $C1$ in red, $C2$ and $C3$ in yellow, $C4$ and $C5$ in blue

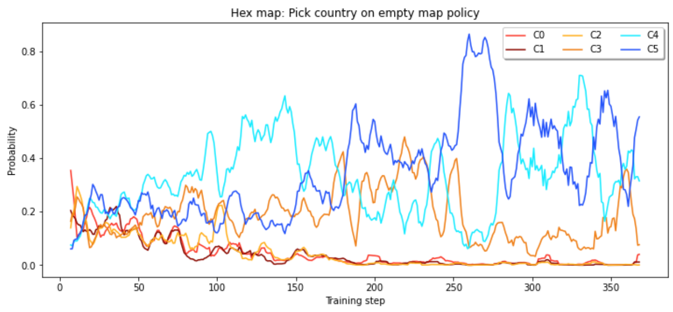

Fig. 3: Evolution of the policy for choosing a country with an empty board.

whether $C4$ or $C5$ is best, which is reasonable given the simplicity and shape of the map.

Next we wanted to know what happened if instead of choosing a country on an empty map, the model had to choose second and the best options were not available. Figure 4 shows that if the *Blue* country $C5$ was not available, the model is able to realize that the best option is either the other *Blue* country $C4$ to stop the enemy from getting the whole continent, or to start taking the *Yellow* continent. Again we see that the model changes its opinion through time between these two options and on the final steps it seems to encourage more the *Blue* country, which is indeed a better choice.

These first results are encouraging and suggest that the model is learning common sense strategies for the country drafting phase guided by the continent bonus. We then evaluated the player on the following phase where, once the countries have been drafted, the players must allocate armies by turns. Given an optimal country drafting (each player has one country in each continent), the player places its armies on the most important countries (See Figure 5).

As a sanity check we ran 1000 games between the neural network player using just the policy (no search) and a random player, half playing first and half playing second, and the neural network player consistently defeated the random player (90% when playing first, 75% when playing second). This confirms that the model was playing better

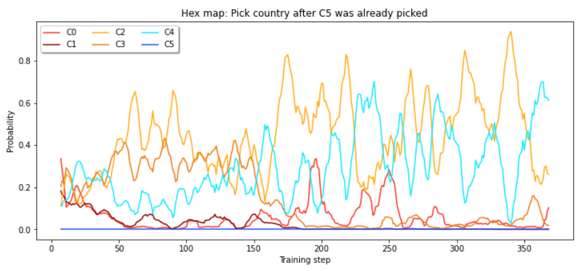

Fig. 4: Evolution of the policy for choosing a country after the best choice is not available

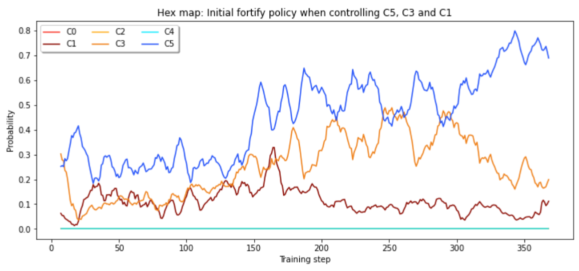

Fig. 5: The optimal game position where every player controls one country of each continent is achieved almost every time if the second player chooses $C4$, then $C2$ and lastly $C0$. In this position, the model prefers fortifying $C5$ which is the most important country.

than random. It is interesting to notice that if the player is given the first decision it will play better, reinforcing what we saw about its ability to draft countries and fortify them at the beginning of the game.

5.2 Mini world map

The Mini world map is a sub graph of the classic map without Asia, Australia and leaving only connecting countries on the continents left. Regarding the continent bonuses, North America and Europe have 5, Africa 3 and South America 2.

Figures 7 is analogous to the one showed in Section 5.1 about the policies for country drafting. Again, the model has learned that the best continents are Europe and North America. This is true because they have the best bonuses while keeping only two entry points. Moreover, it seems to prefer the two countries that are next to each other (Greenland and Iceland). This makes sense because they allow an attack on the other most valuable continent.

Regarding the attack phase, after performing tests on different game positions we concluded that the player has not yet developed a consistent nor intelligent attacking

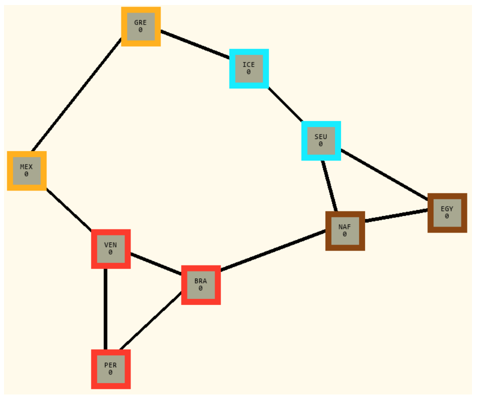

Fig. 6: Mini world map

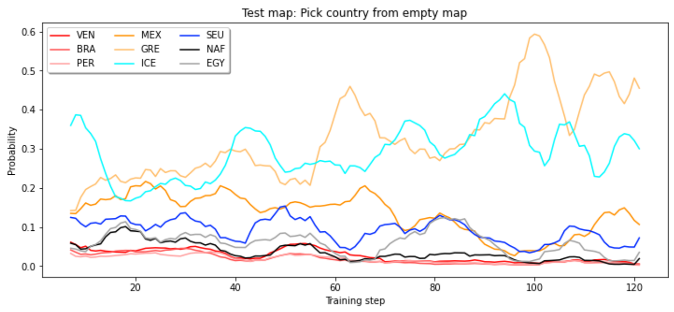

Fig. 7: The player has learned that the best countries to pick at the beginning are either in Europe or in North America.

style. The player appears to have a certain fixation with some territories instead of attacking based on a good reading of the board.

The extreme situation presented in Figure 8 where the player (blue) is cornered in Peru was designed to test its capacity of choosing the best attack target. We proceeded to test two positions, one with 20 armies in Venezuela and 1 in Brazil and vice versa, expecting the player to always attack the country with 1 army. If the player passes, the next turn the opponent will have 13 bonus armies, so it is almost a certain defeat. Therefore the objective of the pinned player should be to attack the weak flank and continue attacking in chain to try to reduce the opponents bonus as much as possible for the next turn, creating a more even position. Unfortunately the player not only prefers passing in both scenarios, the second highest action of the three possible is to attack Brazil, no matter the number of armies it has. This kind of behavior was noticed on different positions, elucidating flaws that need to be dealt with in the future.

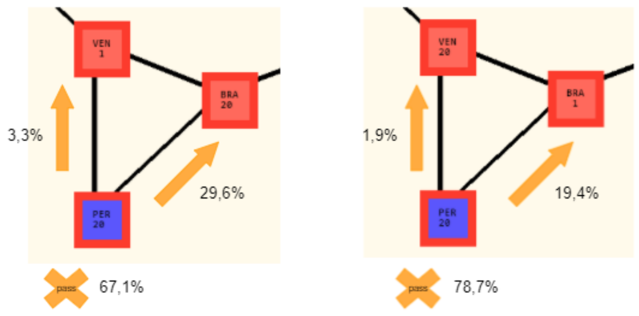

Fig. 8: Player shows no sensitivity to the number of armies on the countries it can attack

6 Future work

We think there are many possible improvements that could stem from this prototype. First, the game engine could allow faster or even parallel simulations to accelerate self-play and specially the expert labeling phase that involves the MCTS.

On the other hand there are many possible customizations for the neural network design to test. The training pipeline has also numerous parameters to study including the use of the value network in the MCTS [24], [1] or the way past experience is sampled [25]. In the paper presenting Katago [28], authors present interesting improvements to the *AlphaZero* pipeline that should also be beneficial for Risk like *Forced Playouts* and *Policy Target Pruning*.

Another natural step is to generalize the player to the full version of the game without simplifications. Some of the game phases would need to be reinterpreted, for example how to define a move in the army placing phase where N armies must be placed.

7 Conclusion

We created a first prototype of a Risk player using graph neural networks and MCTS following the *Expert Iteration* and *AlphaZero* frameworks. On small synthetic maps this model was able to learn optimal strategies for the first game phases like country drafting, confirming that UCT can indeed produce a good country drafting policy [9] that can be learned by neural networks. We conjecture that this game phase might be somehow easier to learn because chance nodes appear deeper in the tree.

On more complex phases like attacking there were no clear signs of learning. The stochastic nature of this phase might be at the root of the problem and either more simulations have to be used to get better action value estimates or other techniques have to be thought of to deal specifically with this chance nodes when they appear directly at the root.

Acknowledgment

This work was supported in part by the French government under management of Agence Nationale de la Recherche as part of the “Investissements d’avenir” program, reference ANR19-P3IA-0001 (PRAIRIE 3IA Institute).

References

1. Anthony, T., Tian, Z., Barber, D.: Thinking fast and slow with deep learning and tree search. arXiv preprint arXiv:1705.08439 (2017)
2. Anthony, T.W.: Expert iteration. Ph.D. thesis, UCL (University College London) (2021)
3. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine learning **47**(2), 235–256 (2002)
4. Blomqvist, E.: Playing the game of risk with an alphazero agent (2020)
5. Browne, C.B., Powley, E., Whitehouse, D., Lucas, S.M., Cowling, P.I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S.: A survey of monte carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in games **4**(1), 1–43 (2012)
6. Carr, J.: Using graph convolutional networks and td (λ) to play the game of risk. arXiv preprint arXiv:2009.06355 (2020)
7. Cazenave, T.: Residual networks for computer go. IEEE Transactions on Games **10**(1), 107–110 (2018)
8. Cazenave, T., Chen, Y.C., Chen, G.W., Chen, S.Y., Chiu, X.D., Dehos, J., Elsa, M., Gong, Q., Hu, H., Khalidov, V., Cheng-Ling, L., Lin, H.I., Lin, Y.J., Martinet, X., Mella, V., Rapin, J., Roziere, B., Synnaeve, G., Teytaud, F., Teytaud, O., Ye, S.C., Ye, Y.J., Yen, S.J., Zagoruyko, S.: Polygames: Improved zero learning. ICGA Journal **42**(4), 244–256 (December 2020)
9. Gibson, R., Desai, N., Zhao, R.: An automated technique for drafting territories in the board game risk. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. vol. 5 (2010)
10. Johansson, S.J., Olsson, F.: Using multi-agent system technologies in risk bots. In: Proceedings of the Second Artificial Intelligence and Interactive Digital Entertainment Conference,(AIIDE), Marina del Rey (2006)
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
12. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
13. Kocsis, L., Szepesvári, C.: Bandit based monte-carlo planning. In: European conference on machine learning. pp. 282–293. Springer (2006)
14. Li, G., Muller, M., Thabet, A., Ghanem, B.: Deepgcns: Can gcns go as deep as cnns? In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9267–9276 (2019)
15. Li, G., Xiong, C., Thabet, A., Ghanem, B.: Deepergcn: All you need to train deeper gcns. arXiv preprint arXiv:2006.07739 (2020)
16. Lütolf, M.: A Learning AI for the game Risk using the TD (λ)-Algorithm. Ph.D. thesis, BS Thesis, University of Basel (2013)
17. Melkó, E., Nagy, B.: Optimal strategy in games with chance nodes. Acta Cybernetica **18**(2), 171–192 (2007)
18. Nijssen, J., Winands, M.H.: Search policies in multi-player games1. ICGA journal **36**(1), 3–21 (2013)
19. Olsson, F.: A multi-agent system for playing the board game risk (2005)

20. Rosin, C.D.: Multi-armed bandits with episode context. Annals of Mathematics and Artificial Intelligence **61**(3), 203–230 (2011)
21. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE transactions on neural networks **20**(1), 61–80 (2008)
22. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. nature **529**(7587), 484–489 (2016)
23. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science **362**(6419), 1140–1144 (2018)
24. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al.: Mastering the game of go without human knowledge. nature **550**(7676), 354–359 (2017)
25. Soemers, D.J., Piette, E., Stephenson, M., Browne, C.: Manipulating the distributions of experience used for self-play learning in expert iteration. In: 2020 IEEE Conference on Games (CoG). pp. 245–252. IEEE (2020)
26. Sturtevant, N.: A comparison of algorithms for multi-player games. In: International Conference on Computers and Games. pp. 108–122. Springer (2002)
27. Wolf, M.: An intelligent artificial player for the game of risk. Unpublished doctoral dissertation). TU Darmstadt, Knowledge Engineering Group, Darmstadt Germany. http://www. ke. tu-darmstadt. de/bibtex/topics/single/33 (2005)
28. Wu, D.J.: Accelerating self-play learning in go. arXiv preprint arXiv:1902.10565 (2019)
29. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems **32**(1), 4–24 (2020)