Algorithm Based On Reward And Punishment Technique For Checker Player

Alicja Winnicka[a]

[a]Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44-100 Gliwice, Poland

Abstract

Board games always need algorithms for players to be a challenge for people. The only way to increase playability for people is to create an algorithm that will be intelligent enough to beat a human. In our paper we decided to implement an algorithm based on reward and punishment, which decides every turn which move is the best, considering the situation on the board.

Keywords

Game, algorithm, reward and punishment, board, checkers, artificial intelligence

1. Introduction

Game development is important due to many different factors, and above all entertainment. It is noteworthy that players drive their development by placing greater and greater requirements regarding the story as well as the quality of the game. This means that newer games need more computing power, so the development of equipment that allows them to run is driven very fast. The plot, the world surrounding the hero are also important and quite often both elements are generated by algorithms that must be constantly developed and improved. In addition, the behavior of opponents is important for the whole game, which is commonly called as artificial intelligence. All these elements are constantly being developed in order to improve the quality of multimedia games production. Improvements in network for remote gaming are also envisaged [1].

The development of game analysis strategies and artificial intelligence activities point to milestones in this field. One of the biggest is the algorithm that defeated the world champion in the game of Go, which is considered much more difficult than chess [2]. Other types of games are also analyzed by the researcher what can be seen in [3]. The authors proposed a new strategy for solving fuzzy matrix games. In [4], the authors described a sequential model for the prediction of poker moves. Again in [5], the idea of using eye gaze to play cards with the use of artificial neural networks was presented. In fact the neural networks have proved to be a very powerful tool for predictions in different fields [6, 7, 8] .

Last year brought different types of reality games like virtualcit [9] or augmented [10, 11] which are still improved. In [12], the authors presented the results of their analysis that games help students learn. Again in [13], the idea of gamification is described with some results based on conducted experiments.

In this paper, we propose a solution for playing checkers, where the current iteration of the game is analyzed on the basis of the reward and punishment technique.

2. Checkers' rules

Checkers are one of the most known classic board games based on strategic thinking. They were invented probably in the XII century and since then there were created many variants of board and play. The basic and the only one being sport discipline is International Checkers also called Polish Checkers, where the board is 10×10 tiles and each player has 20 checkers. However, the most played by people is the English variant with 8×8 board and 12 checkers per player.

The board is divided into black and white tiles arranged alternately, which in our program is grey and yellow to better recognition of checkers, which also are black-white. We used an English variant with checkers set on black tiles. According to the assumption each player has 12 checkers set on the opposite edges of the board in three rows. This arrangement of position is shown in Fig. 1. The goal of this game is to take off all enemy's checkers – then the winner is the player who has at least one checker on the board.

This game requires strategic thinking and considering possible movements of the enemy, and, which is the most important, ability to change a plan depending on the situation on the board.

We assume that checkers can move only diagonally

IVUS 2020: Information Society and University Studies, 23 April 2020, KTU Santaka Valley, Kaunas, Lithuania

Alicja.Lidia.Winnicka@gmail.com (A. Winnicka)

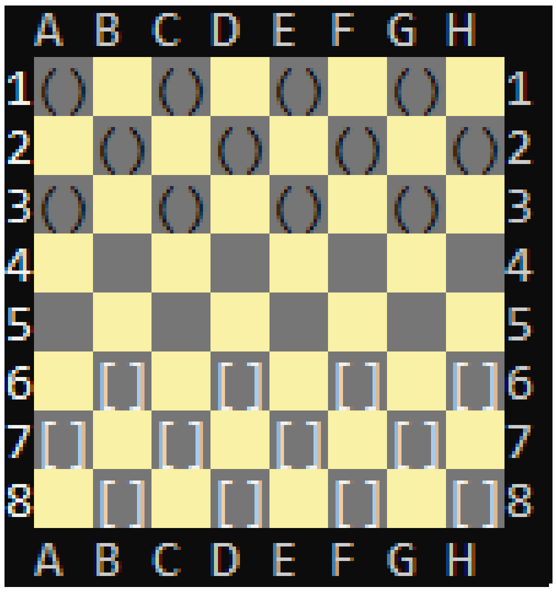

Figure 1: Starting positions of the checkers.

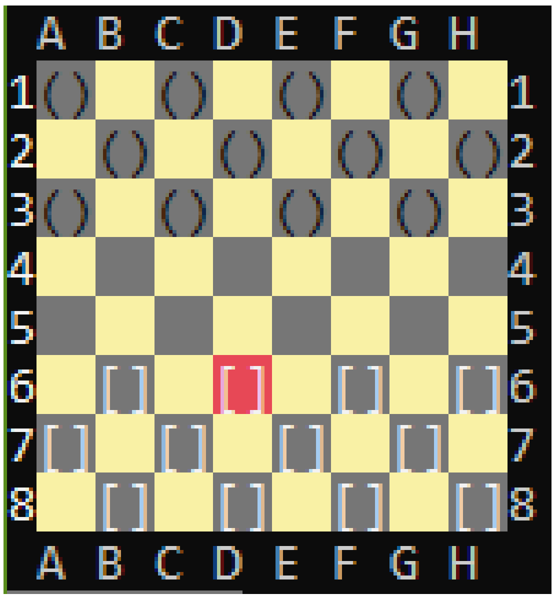

Figure 2: Board with marked checker.

on the black tiles and they can move one tile per turn. The situation, when we want to take an enemy's checker in the neighborhood. Then tile behind this tile must be empty and our checker jumps over enemy's one, so it moves over two tiles. Changing the checker to "King" is a second exception. King can move over the whole board diagonally on black tiles.

One of these movements is treated as one turn, but after taking an enemy's checker player can move again in the same turn. A second player's turn starts when the first player will move without taking his checker.

According to the description above, we had to set a few basic assumptions:

- checkers can move backward,
- checkers can take others checkers backward,
- taking checkers is not obligatory,
- checkers can take kings,
- king can move in diagonal line through whole board,
- board is 8×8,
- checkers move on black tiles,
- player with white checkers starts.

3. Algorithm

The purpose of our algorithm is to find the best option to move during one turn. We may divide it into a few parts.

3.1. First Step

Let us assume that a checker is on (x, y). In this situation, depending on the position, it can have maximally four directions to move, which are checked to classified as possible or not possible to move. It will be classified as not possible when our checker is on the edge of the board or a checked tile is not empty. For example, the checker marked as red in Fig. 2 has two options to move – left-up and right-up – because two tiles below him are used by other white checkers. Because our coordinates starts in top left corner, these tiles are on $(x - 1, y - 1)$ and $(x + 1, y - 1)$.

3.2. Second Step

After finding all the options for a turn, we have to calculate which of them is the best option. This calculating is based on the sum of rewards and punishment connected to these movements. As a reward, we treat being close to changing into the king and taking the enemy's checker. Alternatively, punishments mean being possibly taken in the next enemy's move.

Let us set $s(x, m, n)$ as this sum for the one movement. According to our function, it will be a sum of three values:

- reward for being closer to be a king:

  A basic assumption in this option is the fact, that being closer to being a king means being closer to the enemy's edge of the board. Additionally, it is important, how many checkers have the player. It will be more urgent to have a king when the number of checkers is smaller – it would be very useful to have a king when there is only one left checker on the board.

  We needed a function, which will increase with increasing tile counter x – a number of tiles between player's edge of the board and a checker – and which will have bigger values for a smaller number of checkers m. We defined a function:

$$d(x, m) = |\frac{\sin(x)}{m}| \tag{1}$$

- reward for taking enemy's checker:

  The second reward is the situation when a player can take the enemy's checker. Because it is a goal of the game player should be focused on this – so if the player can take enemy's checker it must be his priority - unless his winning is in danger. Additionally, this move causes the possibility of the next move, so another value to the reward – all these rewards are calculated as a reward of this turn.

  It is worth noticing that the less enemy's checkers, the better is the player's situation. According to this, we assumed that it is more urgent to take enemy's checkers' when there are fewer of them – to finish the game quickly. We needed a function dependent on a number of enemy's checkers n and player's checkers m and we defined simple increasing function:

$$f(m, n) = \frac{1}{mn}, \qquad m, n \in [1, 12] \tag{2}$$

- being taken:

  Being killed must be punished – we have a similar situation to the taking enemy's checker. If the move will cause the possibility of being taken by the enemy's checker, a value of s has to be smaller to avoid this. To balance the difference between taking and being taken we used the same function as punishment, but we will subtract it:

$$k(m, n) = \frac{1}{m * n}, \qquad m, n \in [1, 12] \tag{3}$$

Algorithm 1 Proposed Player's Movement Algorithm

```
1:  Start,
2:  for each player's checker do
3:      for each diagonal direction from the checker do
4:          calculating rewards according to Eq. 4,
5:      end for
6:  end for
7:  set $best$=0,
8:  for each $s$ do
9:      if $s > best$ then
10:         $best = s$
11:     end if
12: end for
13: Return $best$ as final movement.
14: Stop.
```

In this situation our s will be defined with following equation:

$$s(x, m, n) = d(x, n) + f(n) - k(m, n) + r \tag{4}$$

where $x \in [0, 8]$ is number of tiles from player's edge, $m \in [1, 12]$ and $n \in [1, 12]$ are respectively number of player's and enemy's checkers left on the board and $r \in (0, \frac{1}{mn})$ is random value.

Value of s is calculated for each possible move in this turn (so each possibility for each checker) and the next algorithm finds the best of them, which is equivalent to the biggest one. This value is treated as the best option to move considering the situation on the board.

Random value r was added to make small differences between possibilities which are simple moves towards the end of the board and where x is the same. Then s would be the same value. Without r algorithm became predictable, because it would choose the first of the option with the same value. r allowed to make small differences between them.

The whole algorithm of one turn is shown above.

4. Experiments

During experiments, we focused on the impact of a number of enemy's and player's checkers and additionally on the position of checked tile. Fig. 3 shows us the dependence of reward and position of checked tile.

On the axis OX, we located tiles from the player's edge to the enemy's edge and on the OY is calculated to reward s. On this figure we can see two considered options for the move:

- move towards enemy's edge without taking or being taken,

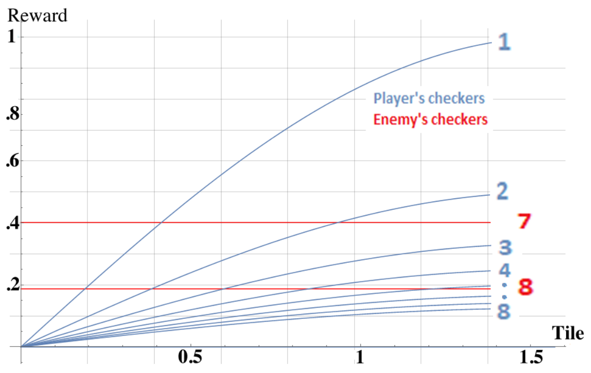

Figure 3: Comparison of reward depending on player's and enemy's checkers.

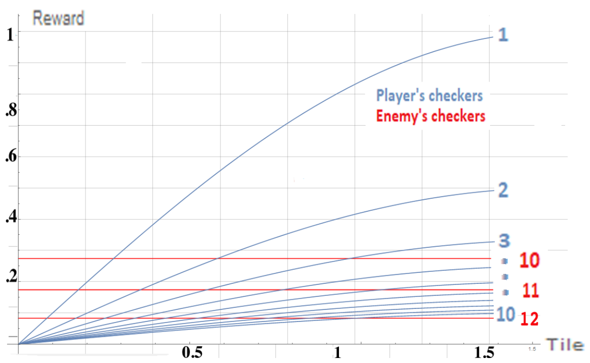

Figure 4: Example of choosing not the best option.

- taking the eighth enemy's checker.

First, one is our $d(x, m, n)$ and is assigned to blue functions on the Fig. 3 They are changing in the connection to a number of player's checkers. This function is a function corresponding to the need of having a king and increases with a decreasing number of player's checkers. Red function means the reward for taking the eighth enemy's checker.

We noticed that taking the enemy's checker is more rewarded in any place of the board for more than five player's checkers. The situation is more complicated when a player has five or fewer checkers – for five checkers and for the fifth tile algorithm will choose to go further to be a king.

This whole algorithm decides about priorities during the play but is not good enough to choose always the best option from a human's point of view. An example of a bad decision is shown in Fig. 4.

Here we have the same two options which are mentioned above, but this time after taking one enemy's checker, the player can take another two, decreasing the number of enemy's checkers to 10. It will do it if player's checkers are mostly above 3, but for 1, 2 or 3 checkers, it will still go to the edge of the board, ignoring the opportunity to take three enemy's checkers.

Another situation is visible on the bottom of the Fig. 4 – when the player has even his full set of 12 and the enemy has 12 too, the player will choose to go further instead of taking one enemy's checker.

One of the most popular algorithms for games is the minimax algorithm. This algorithm finds the best option for the chosen player, considering possible moves for both players and finds the best option from every combination of moves. It ensures that literally the best of all moves will be found, however simultaneously requires much more computing power during each turn to check if the best option is still the best one – because an enemy may do the move which was not included in chosen series of moves.

It means, that our algorithm is more exact than for example minimax algorithm, but is faster and better for a quick game.

5. Conclusions

In this paper, we showed a reward and punishment based algorithm as a possible algorithm deciding how to move in checkers. Our algorithm turned to be good enough to play against an intelligent person, however with a trend to lose checkers at the beginning of the game. It is much better in choosing how to move when a number of player's and enemy's checkers changes dynamically during the game.

In future papers, we will focus on increasing the effectiveness of the algorithm at the beginning of the game.

References

[1] G. Ciccarella, R. Giuliano, F. Mazzenga, F. Vatalaro, A. Vizzarri, Edge cloud computing in telecommunications: Case studies on performance improvement and tco saving, IEEE Int. Conference on Fog Mobile Edge Computing (FMEC 2019) (Rome, Italy, Jun. 2019) 113–120.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., Mastering the game of go without human knowledge, Journal of Artificial Intelligence and Soft Computing Research 550 (2017) 354—-359.

[3] A. Mansoori, M. Eshaghnezhad, S. Effati, Recurrent neural network model: a new strategy to solve fuzzy matrix games, IEEE transactions on

neural networks and learning systems 30 (2019) 2538––2547.

[4] R. Radziukas, R. Maskeliūnas, R. Damaševičius, Prediction of poker moves using sequential model and tensorflow, International Conference on Information and Software Technologies (Springer, 2019) 516––525.

[5] Q. Hu, S. Yean, J. Liu, S. Lee, D. Rajan, R. Phattharaphon, Using eye gaze to play mind card-game using neural network, Proceedings of The Fifth International Conference on Electronics and Software Science (ICESS2019) (Japan, 2019) 54.

[6] F. Beritelli, G. Capizzi, G. Lo Sciuto, C. Napoli, F. Scaglione, Rainfall estimation based on the intensity of the received signal in a lte/4g mobile terminal by using a probabilistic neural network, IEEE Access 6 (2018) 30865–30873.

[7] F. Bonanno, G. Capizzi, G. Sciuto, C. Napoli, Wavelet recurrent neural network with semiparametric input data preprocessing for microwind power forecasting in integrated generation systems, 2015, pp. 602–609.

[8] G. Capizzi, C. Napoli, S. Russo, M. Woźniak, Lessening stress and anxiety-related behaviors by means of ai-driven drones for aromatherapy, in: CEUR Workshop Proceedings, volume 2594, 2020, pp. 7–12.

[9] D. Połap, K. Kęsik, M. Woźniak, Accident prevention system during immersion in virtual reality, in International Conference on Multimedia and Network Information System. (Springer, 2018) 565––573.

[10] D. Połap, Human-machine interaction in intelligent technologies using the augmented reality, Information Technology and Control 47 (2018) 691––703.

[11] D. Połap, M. Woźniak, C. Napoli, E. Tramontana, Real-time cloud-based game management system via cuckoo search algorithm, International Journal of Electronics and Telecommunications 61 (2015) 333–338.

[12] J. Hamari, D. J. Shernoff, E. Rowe, B. Coller, J. Asbell-Clarke, T. Edwards, Challenging games help students learn: An empirical study on engagement, flow and immersion in game-based learning, Computers in human behavior 54 (2016) 170—-179.

[13] M. Sailer, J. U. Hense, S. K. Mayr, H. Mandl, How gamification motivates: An experimental study of the effects of specific game design elements on psychological need satisfaction, Computers in Human Behavior 69 (2017) 371—-380.