Ludo AI

Gregor Trefs & Dominique Ritze

I. Introduction

Artificial Intelligence Ludo (ai-ludo) is an environment for artificial agents. Its purpose is to compare several approaches in the area of artificial intelligence. On the one hand, Ludo is rather a simple game and fully observable but, on the other hand contains a few challenges due to the stochastic and multi agent environment. Therefore it offers a good balance between simplicity and complexity and is able to attract a wide audience and not only professionals. In addition, it is very common and well-known around the world.

Our goal was to provide a simple platform on which everybody could add an own player and check its strength compared to several other ones. By providing a possibility to create game statistics it is feasible to analyze the various players in detail. As first players we implemented the random player and the genetic learning player. The random player randomly decides which pawn to move, while the genetic learning player bases this decision on a utility function to evaluate all possible following game states. In this case, the utility functions is composed of several terms and their weights are determined beforehand with the help of a genetic algorithm. We will show that the approach based on a genetic algorithm is applicable and profitable and significantly outnumbers the random player in wins.

First this paper introduces the ludo game rules players must adhere to be comparable with other players. Ongoing an overview of artificial players in general is given and in addition an example of a similar project. Afterwards the theory about utility functions and genetic algorithms as well as the concrete application on the Ludo game is described. In the end, our experimental results as well as their evaluation is presented.

II. Game Rules

There are several ludo games available. We used the German 'Mensch ärgere dich nicht!' (coarsely translation: 'Don't get bothered !'). The game consists of a board with 40 fields, 1 begin field, 4 start positions, 4 pawns and 4 end fields for each player. A pawn is moveable if the dice count added to the pawn's current position would not lead to a field which is already possessed by another player's pawn and it would not lead to hypothetical field which is behind the end fields. The game is won by a player, if he is the first to have all his pawns on his end fields. The rules of this version are as follows:

- Any player who throws a 6 has to take an own pawn from the start position to a begin field.
  - If there is no pawn left on the start position, the player is allowed to move any moveable pawn.
- Afterwards, the player is allowed to throw the dice again and to move the according number of points.
  - The begin field has to be freed as soon as possible.
  - If there is no pawn left on the start position the pawn is allowed to stay on its begin field.
- If a player's pawn A is moved to a field which is possessed by an opponent's pawn B the opponent's pawn is thrown. This means, pawn B has to be moved back to the corresponding start position.
  - If the field is possessed by another player's pawn C, the player is not allowed to move the pawn A to this field.
- If there are several pawns of a player on the board, the player is allowed to choose which pawn to move.
  - Only one pawn per player is allowed to be moved in one round.
- If a player has no pawns on the field he has 3 attempts per round to throw the necessary 6.

The implementation is based on Java and can be found here: http://code.google.com/p/ai-ludo/

III. Related Work

From the first stirrings in the area of artificial intelligence until today, games have always taken an important position. Often it is easier to test new algorithms on closed environments with a fixed set of rules, like games, before applying them on real world problems. In many cases the first idea is to use search algorithms in order to find a solution, for example like Deep Blue[1] the chess playing agent does it. If a search cannot be used because of the large search space or space/time restrictions, heuristic methods, like the A*-Search or genetic algorithms, may provide a basis for building artificial players. Beside the search strategies, a second huge class of algorithms is concerned with learning. These ones, like neutral networks (e.g. for checkers[2]) or reinforcement learning[3], require some training data and compute the best moves for an artificial player beforehand and not directly during the game. A whole list of possible approaches and their areas of applications together with advantages and disadvantages

[1] 'A new era. How Garry Kasparov changed the world of chess.', Michael Khodarkovsky, Leonid Shamkovich, Ballantine Books, 1997

[2] 'Evolving neural networks to play checkers without relying on expert knowledge', K. Chellapilla, K., D.B. Fogel, Neural Networks, IEEE Transactions, Vol 10, Issue 6, 1999

[3] 'Reinforcement Learning: A Survey', L. Kaelbling, M. Littman, A. Moore, Journal of Artificial Intelligence Research, Vol 4, 1996