Problem 3: Adversarial search for m,n,k-games

Working group assignment

Submit:

- Your group report (within a zip file) here as TP2

  including:

  i) The problem id, your group number, and its elements;

  ii) Your description of the problem and the used algorithm(s)

  iii) The unit tests developed

  iv) All design options taken

  v) Code

  vi) Javadoc

  vii) The implementation UML class diagram

  viii) Results, analysis, and discussion

  ix) Main conclusions or remarks

  x) Bibliographic references used, if any

  up to:
  December 4, 2023 – 17h30

The problem

From Wikipedia [1] we learn that

> "An m,n,k-game is an abstract board game in which two players take turns in placing a stone of their color on an m-by-n board, the winner being the player who first gets k stones of their own color in a row, horizontally, vertically, or diagonally. Thus, tic-tac-toe is the 3,3,3-game (…)"

Fig. 1, also from Wikipedia [1], shows a terminal state of a 11, 10, 5-game.

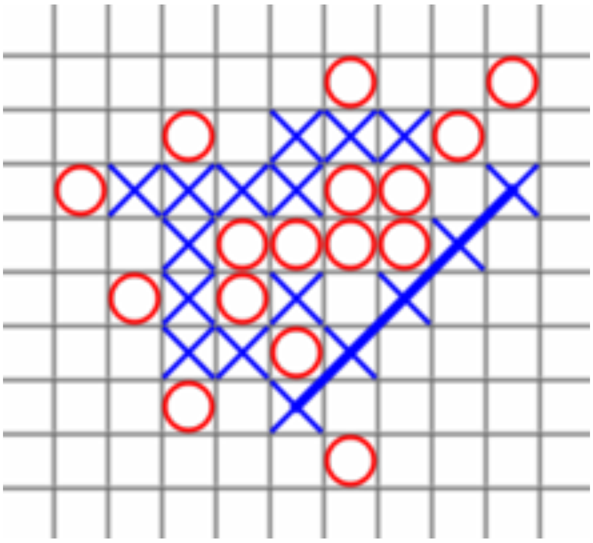

Fig. 2 – An example of a completed 11, 10, 5-game [1]

Task

The task is to select and implement an adversarial search algorithm that can be either the Minimax with alpha-beta cutoffs or the MCTS-Monte Carlo Tree Search algorithm following the approach and design pattern presented in Lab tutorial 1. Any other approach, regardless of its merit, will be quoted with 0 (zero).

See the companion skeleton code to guide your development. You see that the interface Ilayout has been defined with a minimum set of functions. You can extend this interface if needed. The class Board that implements the interface Ilayout is almost ready to use, just add the necessary code where the comment /* YOUR CODE HERE */ is. Once the test for winner is complete it will be possible to play the game using the Console class.

Your agent should implement a function that accepts an Ilayout and return an integer representing the position where the agent wants to play. This function does not change its argument. The int position starts in 0 and increases from left to right and from top to bottom. A reactive agent that plays randomly is given as an example.

If you select MCTS remember that the *tree policy* to select a leaf node *i* is governed by the upper limit of the confidence interval of merit:

$$\frac{w_i}{n_i} + c\sqrt{\frac{\ln t}{n_i}} \qquad (1)$$

where ω_i is the number of wins scored in *i*; n_i is the number of simulations counted in *i*; *t* is the total number of simulations counted at the parent node of *i* and *c* is a user-defined exploration parameter. Also, be aware that the tree policy to be used is the one illustrated in Fig. 2.

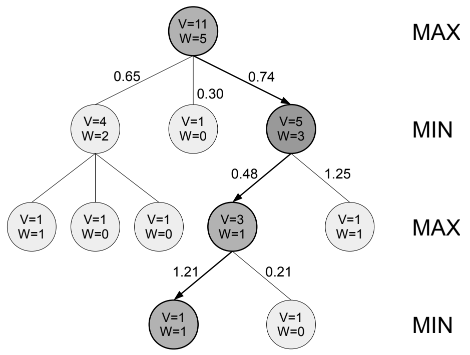

Fig. 2 – Tree policy to be used in 2 agent competitive games [2]. Inside each node there are two counters: the number of simulations or visits *v* and the number of wins w. Edges are labelled with values of expression (1).

Once the search algorithm is implemented it should be applied to a 4,4,4-game (i.e., to a 4x4 Tic-Tac-Toe).

Game on

Each group is invited to take part in a competition which will take place in the last lab class. For competition *any* approach is valid as long as the approach and design pattern presented in Lab tutorial 1 is followed.

Competition
The competition is won by the group with the highest average score taken over the three runs. Should a draw occur, the total duration time of the runs will be used as performance measure. The lower the better.

Rules
1 – The competition has two stage;
2 – All groups are invited to the first stage of the competition, under the following conditions:

- 2 (two) runs as X plus 2 (two) runs as O; each one of them not exceeding 20 seconds;
- After 20 seconds the game stops with a draw.

3 – A group ranking will be produced based on the score of the three simulations.
4 – Only the top three groups will have the chance to participate in the second phase (finals), which will take place under to following conditions:

- The finalist agents will play against each other for
- 3 (three) runs; each one of them not exceeding 20seconds;
- After 20 seconds the game stops with a draw.s

5 – The competition rank will be that of the finals followed by the group rank obtained in the first stage.
6 – No code changes are allowed during the competition. Code should be included within a single file and sent to the instructor before the competition.
7 – Unspecified events will be judged by the instructor;
8 – Appeals, complaints or grievance claims are not accepted.

Rewards

The following bonus for the lab class grades will be award to the final rank, *i.e.*, a max of eight groups will be rewarded.

Place	1st	2nd	3rd	4th	5th	6th	7th	8th
Points	20	16	12	10	8	6	4	2

Group in the top 3 are invited to present their work to the class.

References

[1] https://en.wikipedia.org/wiki/M,n,k-game

[2] Hendrik Johannes Sebastian Baier, *Monte-Carlo Tree Search Enhancements for One-Player and Two-Player Domains*, PhD thesis, Maastricht University, 2015