CR1

Benoit Viguier

06/10/2014

1 Monte Carlo Tree Search Algorithm

1.1 Introduction

Monte Carlo Tree Search (MCTS) is an algorithm used for making optimal decisions in Artificial Intelligence (AI) problems such as solving games or decision making in project managment. It is based on making a big number of random simulations in order to get trustfull datas.

1.2 How does it works ?

The Algorithm create a tree with all possible solution with a small depth. Then it start to run random simulations starting from the leaves in order to test the odds of the outcome. Once we got enough the results (usually we are using time based simulations) we feed back the results and make the decision depending on the odds of each subsequent leaves.

1.3 Example

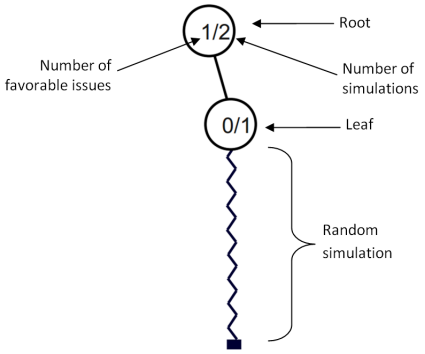


Figure 1: Legend of the following figures.

Figure 2: Run a first simulation from the root, get a favorable issue (will be considered as a *win*).

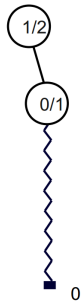


Figure 3: Create a first leaf at depth 1 and run the simulation, get an unfavorable issue (considered as a *loss*).

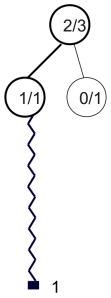


Figure 4: Create a second leaf at depth 1 and run the simulation (*win*).

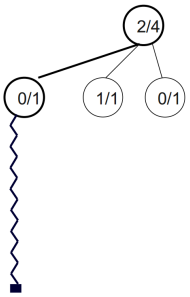


Figure 5: Create a third leaf at depth 1 and run the simulation (*loss*).

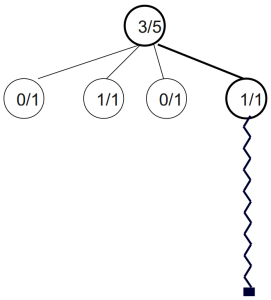


Figure 6: Create a fourth leaf at depth 1 and run the simulation (*win*).

Right now the odds of winning are 3/5. Now that we tested all the possible outcomes at depth 1, we will expend the tree on the favorable leaves (here the second and fourth).

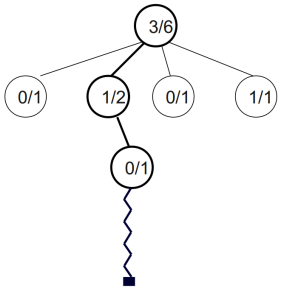


Figure 7: Create a leaf at depth 2 with parent the 2nd leaf at depth 1 and run the simulation (*loss*), update the odds value of the node and making it less interesting than the fourth node. Therefore the algorithm will now work on the fourth node.

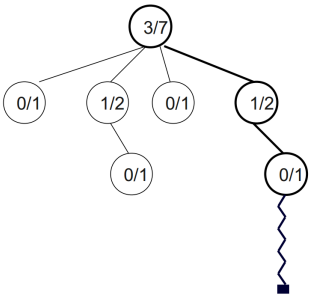


Figure 8: Create a leaf at depth 2 with parent the fourth leaf at depth 1 and run the simulation (*loss*), update the odds value of the node and making it as interesting as the second node. The algorithm will now work on the second node.

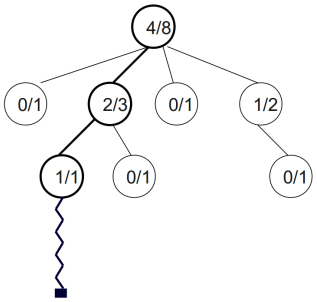


Figure 9: Create a second leaf at depth 2 with parent the second leaf at depth 1 and run simulation (*win*), update the odds value and continue to develop this leaf.

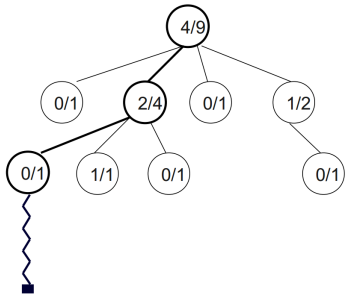
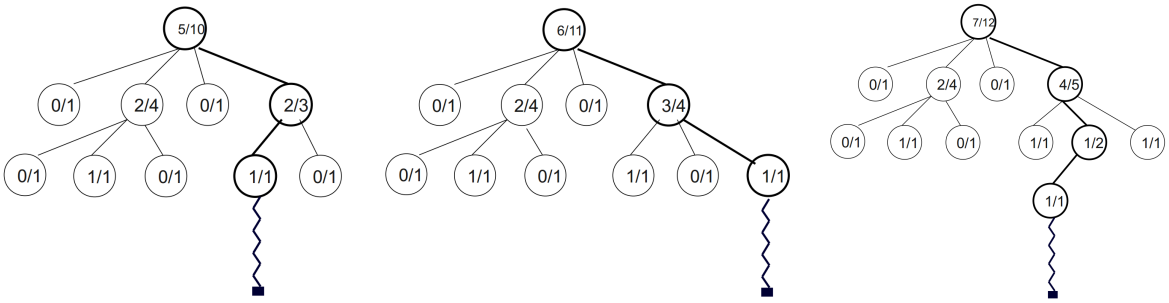


Figure 10: Create a third leaf at depth 2 with parent the second leaf at depth 1 and run simulation (*loss*), update the odds value and switch to the fourth leaf.

Continue the Algorithm until a decent about of simulation are run and/or the time limit is .



Make a decision : here we chose the fourth leaf.

1.4 How to select the leaves to develop ?

In the previous exemple, we chose to not expend leaves without winrates. But depending on the results of the simulations, wins can vary greatly. Therefore we will run more simulations on each leaf before chosing the ones to develop. For practical purpose we will use a cost function in order to select the leaves to expend. The more a leaf is developed, the less it's cost is worth it. This way we can be sure that a leaf with low winrate isn't completely forgotten.

1.5 Why using the Monte Carlo Tree Search ?

Compared to other algorithm like minimax, this one is generic, once you set the rules of the games, given enough time, it will solve it. The advantage of MCTS with it's basic form is that you don't need to implement functions to improve the researches. Based on its random simulations, it will determine by itself which are the good options and which aren't.
The more you run simulations, the more accurate the results will be.

1.6 How much power do we need ?

The more the game has possible moves, the more power it require to solve. In order to get plausible decisions, it needs to go deeper in the tree and to search enough leaves. If the time or number of simulations is not sufficient, the algorithm might miss some important branches and fail to give plausible results. Therefore in order to get decent results, using high-end computer is mandatory, it allows us to get access to multi-threading technology in order to parallelize the simulations.