COSC 322 – 001

Introduction to Artificial Intelligence

2020 Winter Term 2

# Progress Report

**Group Members:**

Jean-Philippe Abadir

Jaden Balogh

Rylan Cox

Guy Kaminsky

Logan Parker

## Current Accomplishments

We have reached several milestones as of the writing of this report. Firstly, we set up a GitHub team account and added every team member to it, allowing the group to work on the project together in a convenient manner through a version control platform.

Next, we were able to get both warmup tasks done and get the project to run on everyone's machines. Completing these warm-up tasks enabled us to have two players participate in a game that contains no rules. Also, the players can enter any of the available rooms they want. In addition, a spectator can also join the room and watch the gameplay unfold.

Of course, having this run on every team member's machine is beneficial, as it allows everyone to familiarize themselves with the code on their own before attending group meetings and working on the project. In addition, everyone in the group familiarized themselves with the API.

Once we finished the warmup tasks, we took time to understand the rules of the game. We wanted to ensure that everyone can understand how the game is meant to be played before implementing a working algorithm. Currently, the group can play the game following the proper rules.

Next, we finished the research on what algorithm we want to use. Based on that research, we collectively decided that we want to implement the Monte Carlo Tree Search (MCTS) algorithm.

## Task Separation and Responsibilities

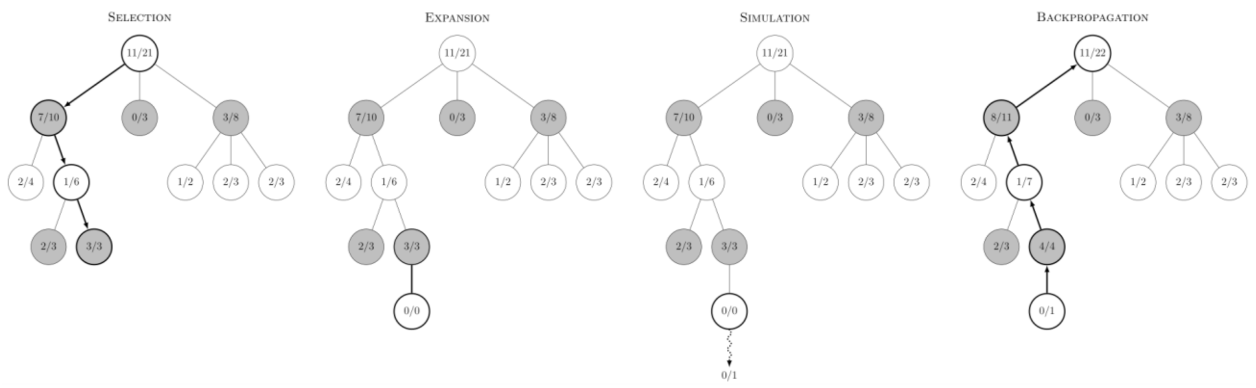
We have split up several tasks among five people to reach the first project milestone. Firstly, Logan and Jean-Philippe started by completing both warmup tasks. This was a crucial point as it allowed the group to make sure we can run the program. Next, Jaden led a peer programming session as we started to make modifications to the player class. After getting more familiar with the program, JP and Rylan started to compile research and compare which algorithm to use. Lastly, Guy took all the information and put it together in this report.

## Algorithm Analysis

### About Monte Carlo Tree Search

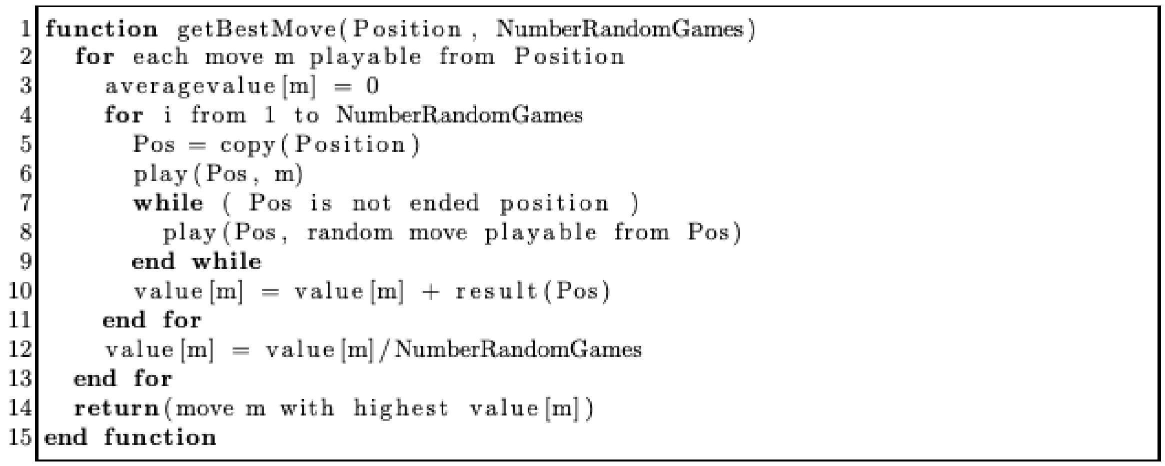
Monte Carlo Tree Search is a technique based on “the analysis of the most promising moves, expanding the search tree based on random sampling of the search space” (https://en.wikipedia.org/wiki/Monte_Carlo_tree_search). This technique seems promising because it “has been used successfully to play games such as Go, Tantrix, Battleship, Havannah, and Arimaa." (https://en.wikipedia.org/wiki/Monte_Carlo_method). All these games are strategy board games which resemble, to a varying extent, the game of Amazons. The “Principle of Operation” section of the page https://en.wikipedia.org/wiki/Monte_Carlo_tree_search gives a good overview of how this technique works. The basic idea is to traverse the current search tree until we reach a node from which no simulated game (called playout) has been run yet. This means that node has no child, making it a leaf. Then, the leaf is expanded. The process of expansion is the following: many times (each time being one playout), the outcome of the game is simulated by picking random legal moves for each opponent starting at the node being expanded, until the game ends. Then, depending on the outcome, a step of backpropagation is run, where the value of each node is updated. The value of the nodes is increased if the simulation resulted in a win for that player, and decreased if it was a loss.

The diagram below helps understand the mechanics of that technique. (Source: https://en.wikipedia.org/wiki/Monte_Carlo_tree_search)



### Monte Carlo Tree Search, as applied to the Game of Amazons

We found this paper: https://dspace.jaist.ac.jp/dspace/bitstream/10119/8867/7/paper.pdf which seems to be especially well-suited to our needs. In it, the author, Julien Kloetzer, dedicates a whole section to applying the MCTS method to the game of Amazons. In that section, before detailing his algorithm, he states that: "[I]t is possible to apply Monte-Carlo with a good confidence to compute a broad number of values" (20). This gives us hope as to the potential of using the MCTS method for our project.



```
function getBestMove(Position , NumberRandomGames)
  for each move m playable from Position
    averagevalue[m] = 0
    for i from 1 to NumberRandomGames
      Pos = copy(Position)
      play(Pos, m)
      while ( Pos is not ended position )
        play(Pos, random move playable from Pos)
      end while
      value[m] = value[m] + result(Pos)
    end for
    value[m] = value[m]/NumberRandomGames
  end for
  return(move m with highest value[m])
end function
```

Figure 3.1: Pseudocode for a basic Monte-Carlo playing engine

The author then explains his algorithm. While the details of it are beyond the scope of this report, we expect to pull inspiration for our final algorithm from the pseudocode provided on page 21, which we have attached above (Figure 3.1).

What the algorithm above does is the following. It starts at a given position, which is a game state. Then, it iterates over all possible moves m from that position. For each move, which results in a position P, it then plays N number of games, where N is an input to the algorithm. For each playout, it picks random moves starting from the same position P. Every time it reaches the end of a playout, it increments the value of the move m. The amount by which the value of the move m is incremented depends on whether the playout resulted in a win, a loss, or a draw. Finally, it divides the value of the move m by the number of simulated games, so that the value of the move m becomes a probability of winning. Finally, it plays the move with the highest probability of winning.

## Other techniques we considered using, and why we decided to go with MCTS

Another technique we have considered using is the min-distance evaluation function detailed in Dr. Martin Muller's paper. This technique is interesting and, according to the author, effective: "A similar pattern became apparent in the Second Computer-Amazons championship in which [our program] participated and took third place". As the author explains, however, this algorithm also has disadvantages: "However, against aggressive human players and programs, such as Jens Lieberum's Amazong, Arrow gets into trouble quickly". For this reason, we thought it would be interesting to explore using Monte Carlo Tree Search.

Additionally, there does not seem to be a very large amount of information to be found online when searching for uses of the Monte Carlo Tree Search method to play the Game of Amazons. We managed to find two papers in addition to the aforementioned one: Amazons Discover Monte-Carlo (https://link.springer.com/chapter/10.1007/978-3-540-87608-3_2) and Analysis of a Monte Carlo Tree Search in Knight-Amazons (https://www.sciencedirect.com/science/article/pii/S1877050915025417), and that second one actually applies the MCTS method to a variant of the game called Knight-Amazons. Therefore, we thought it would be interesting to give this technique a try.

## Unexpected Cases/Difficulties

At this current stage, we encountered a major difficulty when implementing a random player class into the project. When we got together to start working on the project, the server was down. Nevertheless, we continued to attempt to make as many changes to the code as we could. We got to a certain point where we needed a debugger to continue developing our AI. To alleviate this issue, we contacted the professor right away. Although he responded quickly and the server is now working, we could not manage to get our team of five to meet again before the due date of this report, meaning we were not able to finish implementing our first, experimental step: building an AI that makes a random legal move when it is its turn to play.

## Plans Going Forward

First, we want to finish our random move AI step. Then, the plan going forward will be to build on the current progress we will have – with the next step being to expand upon our random player class in order to implement the basic MCTS algorithm. We plan on doing this through the peer programming method in order to ensure all group members can contribute equally and stay on the same page with the progress we make on the project.