COMP30024 Artificial Intelligence Part B Report

Implementation of the Chosen Search Algorithm

For the two-player version of Infexion, an implementation of the Monte Carlo tree search (MCTS) algorithm is used, noting that we chose the A* search algorithm for the single-player version before identifying the shortest paths. However, as the complexity of the game increases, a modified version with a different algorithm is used, as mentioned. For example, the positions of the blue and red cells are expected to be moved around throughout the game, and finding a reliable evaluation function, in this case, is challenging. Hence, MCTS would be suitable as it does not depend on a heuristic evaluation function. Instead, we can use the playouts, focusing on moves with higher win rates while still allowing the flexibility of exploring other potential moves.

MCTS goes through four stages: selection, expansion, simulation, and backpropagation, in which a playout policy is customised accordingly. For example, the simulation is set to be more defensive in this case.

First, we accumulate a list of cells in danger, i.e., cells with more opposing cells than safe ones surrounding them. Then, we attempt to make it "safe" by either spreading it offensively or defensively OR spawning other cells beside our cell. If there are no endangered cells, the opponent cells can be attacked by spreading our cells. If not, another cell is spawned and spread to accumulate the cell power, looking for further opportunities to attack the opponent in the following turns.

We sampled 30 moves (15 on each side) since it is time-consuming to simulate every single playout to the terminal state.

Selection

An upper confidence bound applied to trees is used as the selection policy. For node n, the formula is:

$$UCB1(n) = \frac{U(n)}{N(n)} + C \times \sqrt{\frac{logN(Parent(n))}{N(n)}}$$

in which the node with the highest $UCB1$ value is selected. In this particular case, we have selected a "C" of value 2, a good balance between exploration and exploitation.

Expansion and Simulation

From the selected leaf node, the node is expanded by adding a new child in which it applies the customised playout policy from the new node until it reaches its terminal state. During expansion, we expand all possible SPREAD moves by each cell and SPAWN cells around our existing cells. By SPAWN-ing cells beside existing cells, we set it up so that it is easier to trade1 (further explained in creative aspects) and also make it easier to spread to our own

cells to accumulate power. A trade-off of such an expansion algorithm is the exponential growth of the number of child nodes. Each additional cell adds 6 SPREAD child nodes and potentially 5 SPAWN nodes. In the later stages of the simulation, this may become a weakness as the time needs to be increased to simulate with the necessary accuracy. We attempted to implement an algorithm to limit the number of nodes by eliminating duplicate nodes (due to the mirroring and infinite nature of the board). Still, we ultimately were unable to complete it within the time frame. This can be taken into consideration for further optimisation of the game design.

Backpropagation

Since it is difficult to simulate the outcome (win or loss), we change the heuristic during the backpropagation to totalNumOfCells and ourCells, better reflecting the advantage we hold on the board. For instance, we can draw a fair comparison using the number of wins where we have 5 cells versus 4 cells (5v4) and 50 cells versus 4 cells (50v4). In the former system, both would reflect a winning case, whereas using our newer system, 5v4 would have a value of 5/9 = 0.55, and 50v4 would have a value of 50/54 = 0.93.

As a result, our scoring system will also better reflect the difference compared to just the win-loss state. Another reason is that it is more difficult to achieve an end state through the playout within the set amount of time and challenging to determine whether a state is a win or loss.

Creative Aspects

A strategy similar to chess “trading” is employed (Chess Chivalry, 2022). Specifically, as long as the node is surrounded by more friendly cells than opposing cells, the cell is considered “safe”. This means that, even if the opponent puts in all their resources, the spot is still in reasonable control due to its safe surroundings, allowing better simulation during the simulation stage in MCTS.

References

1. Chess Chivalry. (2022). *Trading Chess Pieces: The Art of Attacking.* https://www.chess-chivalry.com/en/blogs/strategy-and-tactics/trading-chess-pieces-the-art-of-attacking