Project Part B Report

GAME PLAYING APPROACH

The game under consideration is an adversarial type of game with two players. As such, two game-playing strategies that could be considered are the mini-max algorithm and the Monte Carlo Tree Search (MCTS) algorithm. Each algorithm has its own advantages and disadvantages. And after careful consideration, we believe that the mini-max algorithm is better suited for this scenario.

As the game of Infexion has multiple aspects that could affect the outcome of a game, we believe that the mini-max algorithm will perform better by allowing us to design an evaluation function that best represents our game-playing strategy. We are also able to test and substitute multiple evaluation functions against each other in order to determine the best choice. Furthermore, an approach using the MCTS algorithm requires multiple playouts in order to determine the optimal move to make. The more playouts the algorithm can complete, the better the chosen move will be. However, given the time constraints placed upon our agent, we chose not to utilise an MCTS-based agent.

Our agent always makes a spawn move on the centre of the board in the event that it makes the first move. Otherwise, our agent simulates multiple spawn and spread moves using the mini-max algorithm to a depth of three and chooses the best move to make based on the result of the evaluation function and the mini-max algorithm.

Spread moves are simulated fully and every possible grid configuration resulting from all possible spread moves is generated, whereas spawn moves are not fully explored. Spawn moves are generated from a pool of possible safe spawn locations. Safe spawn locations are locations on the grid where if a player were to spawn, the opponent cannot convert the newly spawned hex in the next move. And if there are no safe spawn locations in a particular grid configuration, then no spawn moves are simulated.

EVALUATION FUNCTION

What makes a significant impact on the performance of our algorithm is the chosen evaluation function. The evaluation function is used to determine the desirability of different grid configurations at different points during the game. A good evaluation function ideally takes into account as much information about the grid as possible in order to produce a value depending on the chosen strategy.

The evaluation function used in our game-playing agent consists of a linear weighted sum of different features. In order to construct this evaluation function, we have explored a few aspects of the game that we believed were essential in determining grid configuration desirability.

In the following subsections, the features explored and our reasonings for choosing or discarding them are discussed.

AVERAGE POWER

In a game where individual pieces are associated with a numeric representation of their worth, an approach maximising average value is one that came to mind. The average power is defined as the total power controlled by a player divided by the total number of hexes controlled by the player.

The idea behind such an approach is that if any individual hex has a higher power, the number of hexes that will be converted to the player's colour as a result of a spread move would be larger. In terms of distance, it would mean that a player could close the distance between their hex and an opponent's hex more rapidly with a single move. A player would also be able to convert more hexes with a single move, regardless if the hex was neutral or belonged to the opponent.

The evaluation approach aims to maximise the ability of a player to make significant changes to the grid in a single move, but therein lies a major point of contention. By favouring moves that lead to grid configurations with higher average power, merge moves would almost always be chosen when the board is sparse of coloured hexes. Such as during the early stages of the game. And the agent would hardly play spawn moves, as that would decrease the average power, even when it may be more beneficial in the long run to do so. The agent also opens itself to more detrimental losses in the event that the opponent converts one of its hexes.

NUMBER OF CONVERSIONS

A game of inflexion ends when one player has no more hexes under their control, and this resulting state must be achieved through a spread move. Given that the terminal state is clearly defined, a possible method of choosing moves could be to maximise the number of opponent hexes converted with each move. This is an extremely aggressive strategy that aims to end the game as soon as possible by means of offence.

This method of evaluating moves would heavily favour spread moves and give almost no consideration to spawn moves. This is because as long as at least a single opponent hex can be captured by means of a spread move in a particular grid configuration, a spread move is definitely going to be chosen. The only time a spawn move would be considered is during the start of the game, as a player’s first move has to be spawning a hex, Or if no spread moves result in any conversions.

A strategy that invests completely into conversions may inadvertently expose a player’s hexes into being converted by the opponent in the next move. For example, choosing to make a particular spread move may result in a considerable number of hexes being added to a player’s collection or a significant increase in power. However, the spread move may lead to a player’s hexes being within range of the opponent’s hexes. Such that in the next move the opponent is able to convert a few of the agent’s hexes as well. The net benefit of such a move is then unclear and varies depending on the circumstances.

Furthermore, should the agent find itself with much fewer hexes or much less power an offensive strategy may be detrimental to the survival of the agent. Under such circumstances, a more defensive strategy such as favouring spawn moves may increase the agent’s chances of winning.

DIFFERENCE IN TOTAL POWER

A simple and straightforward method of determining the desirability of grid states in such a game as inflexion would be the difference between the total power of all hexes controlled by each player. The goal of such an evaluation approach is to maximise the difference in power, and favour states where the player has more power compared to their opponent.

However, there are multiple issues related to using such an approach. One example would be that the total power controlled by a player tells nothing of the distribution of power in the hexes. There could be two grid configurations that result in the same difference in total power between players. One configuration could have one player having many low-powered hexes and the opponent having few high-powered hexes. The other configuration could be the opposite.

As such, we concluded that the difference in total power alone is insufficient as an evaluation method. And while this method would not be used, it is still an aspect worth considering when building a final evaluation method.

DIFFERENCE IN THE NUMBER OF HEXES CONTROLLED

Having a large number of hexes on the board means that there are fewer neutral spaces where the opponent can spawn and a larger range of spread activity and more spread options that a player can consider. An agent that tries its best to widen the gap in the number of hexes aims to establish as much control over the grid as possible.

This evaluation approach was initially considered based on the premise that the game ends when one player no longer has any hexes, regardless of how many hexes the opponent has. By favouring a larger difference, the agent will either extend its own control over the grid or push the opponent closer to defeat.

Some advantages of such an approach are that by favouring a larger difference, the agent will either extend its own control over the grid or push the opponent closer to defeat with every move.

There are some obvious weaknesses to this approach as well. For example, the agent does not take into account the power of the cells on the grid. Sometimes it may be more beneficial to merge cells in order to extend the spread reach in future moves, but this agent would almost never consider that as merging decreases the number of cells controlled by the player.

After more exploration and consideration, we acknowledged that while this aspect brings us in the right direction it is far from being our desired agent. However, its performance relative to the other evaluation methods lead us to decide that it could still be partially utilised.

FINAL EVALUATION FUNCTION

The exploration and discussion of the previously mentioned evaluation method had led us to realise that an evaluation method considering only one aspect will not suffice. As such we decide to utilise a linear weighted sum of certain aspects as our chosen evaluation function.

After looking through the multiple evaluation methods, we concluded that our chosen evaluation function should be a weighted sum of the difference in the number of hexes and the difference in total power. We believe that these two aspects of the game are the most significant when considering a move to be made. Furthermore, we gave the difference in total power a higher weight as having more power leads to more opportunities in future moves.

Using the following definitions:
NumDiff = The difference in the number of hexes
PowDiff = The difference in total power

Our agent's linear weighted sum evaluation function is as follows:
*Desirability = 3*PowDiff + 2*NumDiff*

Once we had our final evaluation function decided, we tested it against agents utilising the single-aspect evaluation methods previously mentioned and recorded the results in the table below for comparison.

Using the following definitions:
AvgPow = Average Power
NumDiff = The difference in the number of hexes
NumConv = The number of conversions
PowDiff = The difference in total power
LinSum = The linear weighted sum evaluation function

Average win rates of agent

	AvgPow	NumDiff	NumConv	PowDiff
LinSum	100%	90%	98%	95%

DATA STRUCTURES

A large portion of code in our agent has been taken and adapted from our code in Part A of the project. The data structure used for state representation has largely been unchanged except for a new flag representing if the action is a spawn move or a spread move. The flag is used to determine the return type of the chosen action.

ALGORITHMIC OPTIMISATIONS

The mini-max algorithm generates child states from a parent repeated for a specified depth. The number of total states generated increases exponentially and causes a significant increase in the time spent simulating player moves. In order to reduce overheads where possible, we chose to implement the alpha-beta pruning algorithm alongside the mini-max algorithm to prune as many undesirable nodes as possible.

In addition, we have also chosen to sort the states generated before they are explored to increase the efficiency of alpha-beta pruning. The order by which states are sorted is the number of conversions resulting from the move and the shortest distance between two hexes of opposing colours.

SUPPORTING WORK

In order to determine the suitability of our evaluation function, we need to run out agent against the different evaluation methods for multiple games. To facilitate testing, we made modifications to the referee program given. Instead of the referee just simulating one game and displaying the results, we had the referee run 50 games consecutively and record the result of each of the 50 games.

We ran each pair of agents against each other twice, alternating which agent goes first and second. Using the results of the two sets of simulations, we calculated the average win rate of our agent against every other evaluation function. From this summary of performances, we concluded that our final evaluation function is the best.