Comp 424: Artificial Intelligence

Hus Board Game Playing Agent

Overview & Motivation

The simplicity and speed was the focus in the design of the agent given several constraints in the competition. At the initialization step, the program maps adjacent pits of the inner row and the outer row using an array such that the Nth element of the array contains the adjacent pit of Nth pit. This initial setup enables the program to calculate the number of seeds that can be captured much faster. The program looks at most 2 moves ahead to determine the best move to choose. Each move is evaluated by a heuristic function involving three variables: total number of seeds after the move (**tts**), the number of all seeds that can be captured after the move (**cps**), and the number of pits with seeds less than or equal to 1 (**sp**). A cost of move is calculated to each player as follow: the number of pits with seeds less than or equal to 1 + (the number of all seeds that can be captured / total number of seeds) equivalent to **sp + (cps / tts)**. The agent considers the best move as having the smallest cost difference between the player and the opponent. In other words, smaller the player's moving cost than the opponent's moving cost, better the move is. The program also looks one more move ahead when deciding the move. When calculating these future costs, the agent uses the same formula for the cost and considers two variables in evaluating them: average cost and maximum cost. The two variables are weighed such that the average cost weighs 10% and the maximum cost weighs 90%. Finally, by considering the cost of one move ahead and the cost of one more moves ahead, the agent makes the decision in this fashion: the best move is chosen 80% of the time; the second-best and third-best moves are chosen at 30% of the time each. This approach of decision-making is employed in the hope of preventing infinite loops and disturbing the opposite player who is usually expecting my player to choose the best move.

Theoretical Basis of the approach

The agent utilizes best-first search with heuristic function calculating the cost of one move ahead and the cost of one more move ahead. Each cost is calculated using the following formula: **sp + (cps / tts)** where **sp** is the number of pits with seeds less than or equal to 1, **cps** is the number of seeds that can be captured, and **tts** is total number of seeds. Each variable represents only one player's side. Generally, the ratio of capturable seeds to all seeds would negatively affect the situation to the player and also the number of pits on which the player cannot start a move from would imply how far the player is from the GameOver condition. The cost calculation is performed to both the player and the opponent and the final result is the difference from the player's cost to the opponent's cost. Being partly inspired by Minimax, the agent also looks at all potential moves of the opponent after each of player's all valid moves so that the accuracy of its prediction could be improved. When inspecting the moves of the opponent, the program keeps track of maximum cost and the average cost of each move and eventually takes 10% of the average and 90% of the maximum into account. Also, it may not be the best idea to always pick the best move since it may already have been foreseen by the opponent so that the opponent already knows its next move accordingly. In addition, there is a risk of getting into infinite loops if the agent always tries to pick the best move. For these reasons, the agent, being partly inspired by simulated annealing, chooses the best move only 80% of the time and allows the second-best and the third-best at a chance of 30% each.

Pros & Cons

First of all, the agent is extremely fast in making its move thanks to the simple heuristic function. Surely it would depend on the computing power of the hardware, yet it barely exceeds 1 second to finish its job and on average, it only takes about 0.1 second or less on Trottier machines for making each move. Also, the program does not require any file I/O activities for its initialization steps nor heavy use of data structures: The agent only initializes one array mapping the adjacent pits as a basic setup. The absolute satisfaction of time and memory constraints would be the main strength of the agent. However, there are several disadvantages involved in this approach. Basically, the agent is not using given resources

fully efficiently. In fact, the agent is performing its search only up to the depth of 2, which is shallow given the extra amount of time and memory spares. Also, the best move is only picked at probability of 80% which means the agent may have excessive amount of freedom to choose a non-optimal move. Moreover, the optimality and the accuracy of the heuristic function is not guaranteed since it is solely based on the knowledge and assumptions which are too simple to be mature. On the other hand, I believe that the heuristic is nearly admissible in most cases.

Other Approaches

Before coming up with the final approach, there was another approach which was later opted out due to incorrectness. The other approach has a different cost calculation formula: cost = **sp * (cps / tts)** where **sp** is the number of pits with seeds less than or equal to 1, **cps** is the number of seeds that can be captured, and **tts** is total number of seeds. This approach failed to win the game most of the time because of the inaccurate cost calculation.

Future Improvement

Since the agent is not inspecting the moves after inspecting two future moves, it would be better to inspect the moves further while still respecting the time and memory constraints. In this case, exponentially growing game search tree may result in a memory constraint violation and alpha-beta pruning would significantly slow down the growth of memory use. Multithreading would be useful to respect the time constarint as it could keep track of the elapsed time so that the agent may stop the search and make a move before the deadline. Also, it would be necessary to investigate the game rule and more case scenarios in order to improve the heuristic function for cost calculation. One ambitious improvement would be to learn how the opponent is choosing its best move at each game and enlarge the knowledge base gradually. This way, the agent would be able to better predict the bahaviour of the opponent game after game.

Conclusion

Although my agent is primitive and somewhat immature in some aspects, at least it does fulfill all the basic requirements: it respects all the competition constraints; it always beats the randomly generated player. Actually, the real power of the agent comes from the combination of best-first search, optimization inspired by simulated annealing, and game tree search inspired by minimax of depth 2. In the effort of implementing a simpler and faster agent, the full efficiency of using the resources such as memory, time, and multithreading might have been sacrificed, yielding a plenty of room for improvement. There will be many diverse agent playing Hus Board Game in the competition and it will prove if the focus on the simplicity and speed in building an agent is indeed worthwhile.