Introduction to Artificial Intelligence

HW4

Almountassir Bellah Aljazwe

Results

1. Without Heuristic
   1. Minimax with no pruning: Too many game possibilities for any result.
   2. Minimax with pruning: Too many game possibilities for any result.
2. With Heuristic
   1. Minimax with no pruning:

|  |  |  |
| --- | --- | --- |
| AI Move # | States Evaluated | Time Used (secs) |
| 1 | 18832 | 1.12 |
| 2 | 43996 | 2.24 |
| 3 | 74042 | 4.42 |
| 4 | 34014 | 1.86 |
| 5 | 74042 | 4.26 |
| 6 | 34014 | 1.96 |
| 7 | 74042 | 5.80 |
| 8 | 34014 | 2.80 |
| 9 | 74042 | 4.99 |
| 10 | 34014 | 1.92 |
| Averages | 21611 | 3.14 |

* 1. Minimax with pruning

|  |  |  |
| --- | --- | --- |
| AI Move # | States Evaluated | Time Used (secs) |
| 1 | 1426 | 0.08 |
| 2 | 2440 | 0.14 |
| 3 | 3474 | 0.20 |
| 4 | 2100 | 0.12 |
| 5 | 3474 | 0.20 |
| 6 | 2100 | 0.11 |
| 7 | 3474 | 0.23 |
| 8 | 2100 | 0.15 |
| 9 | 3474 | 0.27 |
| 10 | 2100 | 0.14 |
| Averages | 1308 | 0.16 |

Discussion

The results display how the results differ from minimax without and with alpha-beta pruning; minimax with alpha-beta pruning is executed in far less time due to a lesser number of states evaluated. Unfortunately, the heuristic applied is not a good one. My initial plan was to use a Monte Carlo Tree Search to simulate from a particular state and return the ratio of wins to games played. The problem was that it could not generate the tree due to the requirement of a huge memory storage (what I concluded was the problem). For the last plan heuristic, the heuristic calculates the differences between positions; it then returns the reciprocal values of the differences.