**GROUP ASSIGNMENT 2**

Expert Systems with Uncertainty

**ARTIFICIAL INTELLIGENCE**

CSC 3206

**TUESDAY P3 1**

Muhammad Awad Luckhoo (15042278)

Choong Kai Wern (15053648)

Teh Cuok Syen (15061567)

Ong Li Shen (15071863)

Mu Chun Khang (13079272)

Mah Qi Hao (16080111)

# TABLE OF CONTENTS

[TABLE OF CONTENTS 2](#_Toc485895675)

[QUESTION 1 3](#_Toc485895676)

[Preface 3](#_Toc485895677)

[Problem Space 3](#_Toc485895678)

[Search Strategy Recommendation 4](#_Toc485895679)

[QUESTION 2 6](#_Toc485895680)

[Literature Review 6](#_Toc485895681)

[PEAS of Turtle Agent 6](#_Toc485895682)

[FOL Rules 7](#_Toc485895683)

[Knowledge Representation Scheme Recommendation 7](#_Toc485895684)

[Estimated Path Cost 9](#_Toc485895685)

[NetLogo Applet 9](#_Toc485895686)

[QUESTION 3 10](#_Toc485895687)

[Literature Review 10](#_Toc485895688)

[Generation of Game Tree 10](#_Toc485895689)

[AI Implementation 11](#_Toc485895690)

[Heuristic Function 11](#_Toc485895691)

[Effectiveness of Pruning 12](#_Toc485895692)

[Java Applet 13](#_Toc485895693)

[QUESTION 4 14](#_Toc485895694)

[Contributions of Members 14](#_Toc485895695)

[Gantt Chart 15](#_Toc485895696)

[REFERENCES 16](#_Toc485895697)

# QUESTION 1

## Literature Review

Write here…

## Comparative Analysis

## Justification

Topic

## Expert System Design

Topic

# QUESTION 2

## Literature Review

## Quality of Recommendations

# QUESTION 3

## Literature Review

Reversi is strategy board game played by two players on an eight-by-eight board. To develop an Artificial Intelligence (AI) to play Reversi, research was done on the minimax algorithm. Minimax is an algorithm used to search for the optimal path to a two-player game of perfect information given enough time and space (Korman, 2013).

Minimax might be the solution to search for an optimal path, but it takes a long time to execute in real-life scenarios because there are too many depths to explore. In this case, alpha-beta pruning can be used to significantly reduce the execution time of the minimax algorithm (Korman, 2013). The way the pruning works is by eliminating branches that do not contain the optimal path (Korman, 2013). To enhance the response time of a Reversi AI, the implementation should include the minimax algorithm with alpha-beta pruning. A heuristic function is required to determine how well a specific game state is (Korman, 2013).

## Generation of Game Tree

To generate the game tree for the implementation of the minimax algorithm, the right data structure is required. ArrayList was the first data structure considered. However, it quickly became obsolete because of how difficult it was to access child nodes and visualize the process. As the tree grows, keeping track of the parent and child nodes became even more troublesome. Hence, a class needs to be written to especially manage the game tree.

A Node class is created to store each possible game state in the game tree to better visualize the game tree structure. Herein, the class contains several properties like an array to store the child nodes, the current parent node, the game state, the current game state, the coordinates of the current move and the utility value. Besides that, the Node class also contains an important method called addChildNode to add nodes to the current node.

Victor Allis (1994) mentioned that Reversi has an approximate state space complexity of 1028. Therefore, it is impossible to generate all the game states because the regular computer simply does not possess the required memory. At this juncture, the game tree depth was limited to 6. In other words, the Reversi AI implemented can only look 6 steps ahead to determine the most optimal move.

## AI Implementation

In the implementation of the Reversi AI, the minimax algorithm returns a Node object containing both the move and the score of the game state instead of returning the score or the move. The minimax function does not accept depth as the parameter contrary to usual implementations. The implementation also sees the base case for the minimax algorithm checking if the node is a leaf node. In other words, the last node in the game tree contains no child node. As for the alpha-beta pruning, the Noe object created is also utilized in the implementation. The minimax algorithm with alpha-beta pruning is akin to the minimax algorithm. However, for the minimax with alpha-beta pruning, an alpha and beta node are passed in as parameters before the algorithm is called recursively.

## Heuristic Function

The heuristic function uses a couple of factors to evaluate the game state and thus the utility value of the leaf nodes of game tree. Firstly, the score difference between the black and white player is calculated for after taking the move. Coin parity is the heuristic approach that refers to the score difference between the max and min player (Sannidhanam & Annamalai, 2015). However, this approach alone is too simple as it does not take into account the move’s stability. Therefore, the position of the move also needs to be considered.

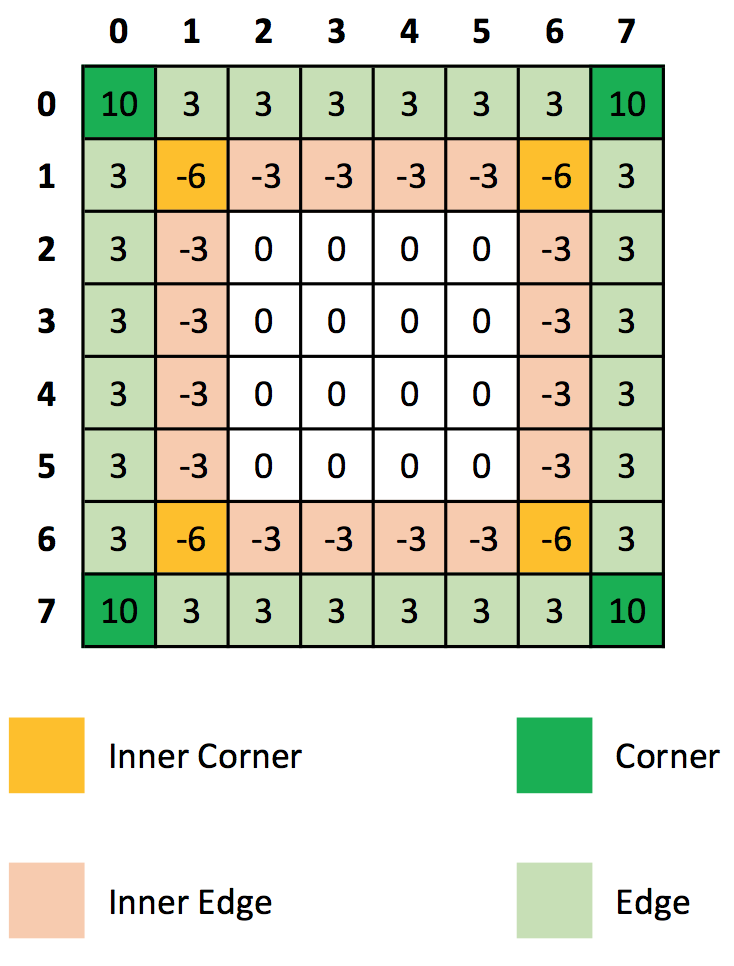


Figure 5: Weights assigned to positions on the board

Different weights are assigned to different positions on the game board. Moves at the corners or edges increases the utility values because they cannot be flanked by the opponent, safeguarding the stability of the specific region (Sannidhanam & Annamalai, 2015). In contrast, moves placed on the inner corners and edges create opportunities for the opponent to counter.

## Effectiveness of Pruning

To calculate the effectiveness, the Reversi AI was tested with human players for a total of 5 rounds, comparing minimax algorithm and minimax algorithm with alpha-beta pruning. The time for when the AI starts to generate the game tree and when it has found the optimal move. To ensure consistency of results, the same human player was used for all the game rounds.

We save the initial time just before the AI starts to generate the game tree and save the end time after the AI manage to generate the game tree and execute the algorithm. To ensure the consistency of the result, the same computer is used for all the tests and the same individual is chosen to play all the games. Depths of 5 and 6 were tested respectively to better compare the efficiency of pruning.

Game Tree of Depth 5

|  |  |  |  |
| --- | --- | --- | --- |
| **Round** | **Average (ms)** | **Black** | **White (AI)** |
| 1 | 386.03 | 23 | 41 |
| 2 | 671.83 | 28 | 36 |
| 3 | 357.53 | 19 | 45 |
| 4 | 390.73 | 16 | 48 |
| 5 | 435.60 | 13 | 51 |
| **Total Average (ms)** | 448.34 | | |

Table 3: Average time for minimax algorithm (depth 5)

|  |  |  |  |
| --- | --- | --- | --- |
| **Round** | **Average (ms)** | **Black** | **White (AI)** |
| 1 | 271.66 | 17 | 47 |
| 2 | 277.41 | 24 | 40 |
| 3 | 258.77 | 21 | 43 |
| 4 | 379.30 | 15 | 49 |
| 5 | 387.97 | 17 | 47 |
| **Total Average (ms)** | 315.02 | | |

Table 4: Average time for minimax algorithm with alpha-beta pruning (depth 5)

Game Tree of Depth 6

|  |  |  |  |
| --- | --- | --- | --- |
| **Round** | **Average (ms)** | **Black** | **White (AI)** |
| 1 | 1953.81 | 18 | 46 |
| 2 | 3088.52 | 23 | 41 |
| 3 | 2437.33 | 21 | 43 |
| 4 | 4081.60 | 29 | 35 |
| 5 | 5191.69 | 0 | 30 |
| **Total Average (ms)** | 3350.59 | | |

Table 5: Average time for minimax algorithm (depth 6)

|  |  |  |  |
| --- | --- | --- | --- |
| **Round** | **Average (ms)** | **Black** | **White (AI)** |
| 1 | 1435.67 | 18 | 46 |
| 2 | 1583.61 | 31 | 33 |
| 3 | 995.90 | 14 | 50 |
| 4 | 2074.93 | 20 | 44 |
| 5 | 1715.00 | 30 | 34 |
| **Total Average (ms)** | 1561.02 | | |

Table 6: Average time for minimax algorithm with alpha-beta pruning (depth 6)

The results from tables 3 – 6 show that for a game tree of depth 5, the minimax algorithm with alpha-beta pruning yielded an improvement of 29.7% over the minimax algorithm. Meanwhile, for the game tree of depth 6, there was an improvement of 53.4% when the minimax algorithm with alpha-beta pruning was used as compared with the minimax algorithm.

Thus, it can be concluded that minimax algorithm with alpha-beta pruning delivers significant improvement over minimax algorithm. Besides that, the results indicate that the efficiency of minimax algorithm with alpha-beta pruning betters with increased depth.

## Java Applet

The Java Applet below uses the implementation for the Reversi game is obtained from Assignment 2 of ICS 23 Spring 2012 Course of University of California. Alex Thornton and Norman Jacobson (2012) implemented the graphical user interface and game logic.

INSERT JAVA APPLET

# QUESTION 4

## Contributions of Members

Muhammad Awad Luckhoo

Gathered information, formulated problem space and purposes, formulated PEAS, implemented minimax algorithm and alpha-beta pruning, prepared Gantt chart, and compiled report.

Choong Kai Wern

Gathered information, researched informed search strategies, implemented minimax algorithm and alpha-beta pruning, and evaluated effectiveness of pruning.

Teh Cuok Syen

Gathered information, researched informed search strategy, formulated PEAS, discussed knowledge representation scheme for agent, and compiled report.

Ong Li Shen

Gathered information, formulated problem space and purposes, discussed knowledge representation scheme for agent, implemented minimax algorithm and alpha-beta pruning, evaluated effectiveness of pruning.

Mu Chun Khang

Gathered information, researched uninformed search strategy, added turtle agent to maze and estimated path cost, generated FOL rules for agent operations, and compiled report.

Mah Qi Hao

Gathered information, researched uninformed search strategy, added turtle agent to maze and estimated path cost, and discussed knowledge representation scheme for agent.

## Gantt Chart

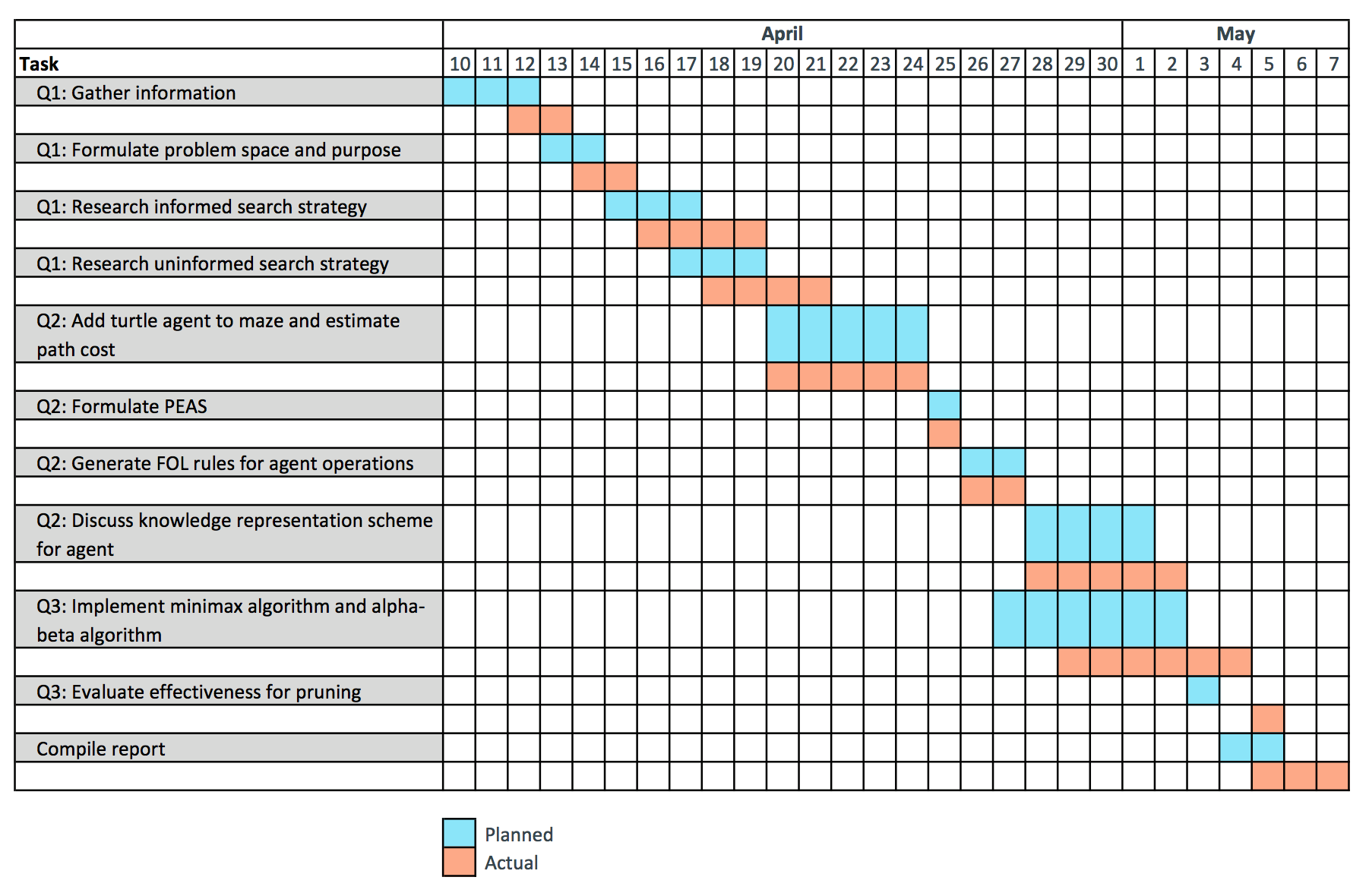


Figure 6: Gantt Chart

# REFERENCES

Allis, L. V. (1994). *Searching for solutions in games and artificial intelligence*. Ponsen & Looijen.

Amit (2017). Introduction to A\*. Retrieved from http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

Babula, M. (2009). *Simulated maze solving algorithms through unknown mazes*. Organizing and Program Committee, *13*.

Barnouti, N. H., Al-Dabbagh, S. S. M., & Naser, M. A. S. (2016). Pathfinding in Strategy Games and Maze Solving Using A Search Algorithm. *Journal of Computer and Communications*, *4*(11), 15.

Beck, L. (2014). A Case Study on the Search Topology of Greedy Best-First Search. From http://ai.cs.unibas.ch/papers/theses/beck-master-14.pdf.

Chand, M., Goel, M., & Rathore, S. (2017). *Maze Solving Algorithms*. From https://pdfs.semanticscholar.org/6c0b/a9533602e32555888b31d0747801dcd51af7.pdf.

Dorin, A. (2017). A-Life and Virtual Environments. From http://users.monash.edu/~cema/courses/FIT3094/lecturePDFs/lecture6a\_Astar.pdf.

Eckroth, J. (2017). *Informed search*. CSCI 431. Csci431.artifice.cc. Retrieved from http://csci431.artifice.cc/notes/informed-search.html#tocAnchor-1-6.

Korman, M. J. (2003). Playing Othello with Artificial Intelligence.

Plesník, J. (1981). *A bound for the Steiner tree problem in graphs*. Mathematica Slovaca, *31*(2), 155-163.

Reba, M. A., & Shier, D. R. (2014). *Puzzles, Paradoxes, and Problem Solving: An Introduction to Mathematical Thinking*. CRC Press.

Robin. (2009). *A Star Algorithm*. Intelligence. Retrieved from http://intelligence.worldofcomputing.net/ai-search/a-star-algorithm.html.

Sannidhanam, V., & Annamalai, M. (2015). An Analysis of Heuristics in Othello.

Teahan, W. J. (2010). Hampton Court Maze Wall Following NetLogo model. Artificial Intelligence. Ventus Publishing Aps.

Thornton A., & Jacobson N. (2012). Project #2: Black and White. Retrieved from http://www.ics.uci.edu/~thornton/ics23/LabManual/BlackAndWhite/.