A logo for college computing

Description automatically generated

**Assessment Cover Page**

|  |  |
| --- | --- |
| *Student Full Name* | Santhosh Shanmugasundaram |
| *Student Number* | Sba24100 |
| *Module Title* | AI Concepts to Implementation |
| *Assessment Title* | CA1 |
| *Assessment Due Date* | 10th November 2024 |
| *Date of Submission* | 10th November 2024 |

**Declaration**

By submitting this assessment, I confirm that I have read the CCT policy on academic misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source.

I declare it to be my own work and that all material from third parties has been appropriately referenced.

I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution.

CA1 – AI Concepts to Implementation

Contents

[1. Task1: Two Artificial Intelligence Strategy that are used to play Chess. 2](#_Toc182156774)

[1.1 Introduction: 2](#_Toc182156775)

[1.2 Why Chess for AI Research: 2](#_Toc182156776)

[1.3 Why Artificial Intelligence for Chess Game: 2](#_Toc182156777)

[1.4 Working Principles of AI Chess Engine: 2](#_Toc182156778)

[1.5 Min-Max Searching 3](#_Toc182156779)

[1.5.1 Key Terminologies 3](#_Toc182156780)

[1.5.2 Steps 3](#_Toc182156781)

[1.5.3 Example 4](#_Toc182156782)

[1.5.4 Key Properties of Min-Max Algorithm 4](#_Toc182156783)

[1.6 Alpha-Beta Pruning 5](#_Toc182156784)

[1.6.1 Key Terminologies 5](#_Toc182156785)

[1.6.2 Key Steps 5](#_Toc182156786)

[1.6.3 Condition for Pruning 5](#_Toc182156787)

[1.6.4 Example 6](#_Toc182156788)

[2 Task2: 7](#_Toc182156789)

[2.1 Find the Best Route from Tipperary to Sligo 7](#_Toc182156790)

[2.1.1 Route Finding Using Depth-First Search (DFS) 7](#_Toc182156791)

[2.1.2 Route Finding Using Dijkstra’s Algorithm 7](#_Toc182156792)

[2.1.3 Route Finding Using A\* Algorithm 8](#_Toc182156793)

[2.1 Conclusion 8](#_Toc182156794)

[2.2 Two AI Strategies Used in Modern Games 8](#_Toc182156795)

[2.2.1 Monte Carlo Tree Search (MCTS) in AlphaGO Zero: 9](#_Toc182156796)

[2.2.2 Behavior Tree Algorithm in Halo 2 10](#_Toc182156797)

[3 Task3: 10](#_Toc182156798)

[3.1 Characteristics of AIBO 11](#_Toc182156799)

[3.2 AIBO – Scope for Improvement 12](#_Toc182156800)

[4 References 12](#_Toc182156801)

[5 GitHub Link 15](#_Toc182156802)

# Task1: Two Artificial Intelligence Strategy that are used to play Chess.

## Introduction:

Chess is a strategic board game played between two opponents on an 8x8 grid of alternating light and dark squares. Chess demands foresight, calculation, and tactical prowess, making it a timeless test of intellect and skill.

Chess is a closed system with total moves between 10^111 & 10^123 and total legal moves of 10^40, which is more than the number of atoms in the universe. (Rolls)

## Why Chess for AI Research:

Chess has a well-defined rule structure, it has been useful for researchers to use it as testing ground for different algorithms and machine learning approaches. We see approaches piloted in chess that are later applied to many different areas of the modern world including natural language processing, reinforcement learning, game theory etc. (Rolls)

## Why Artificial Intelligence for Chess Game:

Artificial Intelligence based chess engine has increased the overall level of chess skill. From beginner to Super Grand Master the standard of chess has improved as AI is an invaluable teaching aid. It allows players to rapidly evaluate the strength of a position and investigate possible lines of play, as well as to review their games. This is particularly true with opening theory where now, with the help of chess engines, players will often have “book” (the best) moves memorized for the first 15-20 moves. (Rolls)

## Working Principles of AI Chess Engine:

AI is fundamental in chess engines because it allows the computer to evaluate positions and make informed decisions based on its analysis. Chess engines use various algorithms and techniques, including machine learning, to evaluate positions and carry out the next move. They do this by analysing vast amounts of data to come up with very solid and accurate position choices, allowing them to play much faster than a human could. (Degni)

Modern chess engines are composed of two parts: a search algorithm to provide a restricted selection of good moves, and an evaluation function to choose the move to play.

Two key strategies that are used in modern chess engines are:  
1. Min-Max Searching   
2. Alpha-Beta Pruning

## Min-Max Searching

The Mini-Max algorithm is a decision-making algorithm used particularly in game theory and computer games. In a two-player game, one player is the maximiser, aiming to maximize their score, while the other is the minimizer, aiming to minimize the maximiser’s score. The algorithm operates by evaluating all possible moves for both players, predicting the opponent's responses, and choosing the optimal move to ensure the best possible outcome.

### Key Terminologies

* Maximizing Player (Max):   
  Chooses the move that leads to the highest possible utility value, assuming the opponent will play optimally.
* Minimizing Player (Min):  
  Selects the move that results in the lowest possible utility value for the maximiser, assuming the opponent will play optimally.

The interplay between these two players is central to the Min-Max algorithm, as each player attempts to outthink and counter the other's strategies.

### Steps

The Min-Max algorithm involves several key steps, executed recursively until the optimal move is determined. This algorithm applies Depth First Search (DFS); therefore, we must go till the terminal node.

**Step1**: Generate the Game Tree  
Create a tree structure representing all possible moves at the current state of game.

**Step2**: Evaluate Terminal States  
Assign utility values to the terminal nodes of the game tree.

**Step3**: Propagate Utility Values Upwards  
Starting from the terminal nodes, propagate the utility values upwards through the tree.  
For each non-terminal node, if it's the maximizing player's turn, select the maximum value from the child nodes.  
If it's the minimizing player's turn, select the minimum value from the child nodes.

**Step4**: Select Optimal Move  
At the root of the game tree, the maximizing player selects the move that leads to the highest utility value. (GeeksforGeeks)

### Example

A simple game where each player can choose between two moves at each turn. Here's a basic game tree:

Max

/ \

Min Min

/ \ / \

+1 -1 0 +1

At the leaf nodes, the utility values are +1, -1, 0, and +1.

The minimizing player will choose the minimum values from the child nodes: -1 (left subtree) and 0 (right subtree).  
The maximizing player will then choose the maximum value between -1 and 0, which is 0.

Thus, the optimal move for the maximizing player, considering optimal play by the minimizer, leads to a utility value of 0.

### Key Properties of Min-Max Algorithm

**Complete**- Min-Max algorithm is Complete. It will find a solution (if exist), in the finite search tree.

**Optimal**- Min-Max algorithm is optimal if both opponents are playing optimally.

**Time complexity**- As it performs DFS for the game-tree, so the time complexity of Min-Max algorithm is O(bm), where b is branching factor of the game-tree, and m is the maximum depth of the tree.

**Space Complexity**- Space complexity of Mini-max algorithm is also like DFS which is O(bm).

## Alpha-Beta Pruning

Alpha-beta pruning is a technique used to improve the efficiency of the minimax algorithm by reducing the number of nodes that need to be evaluated in a game tree. In minimax, every possible move and countermove is evaluated, but many of these moves don’t affect the final decision.   
Alpha-beta pruning works by eliminating branches that are guaranteed to not influence the outcome. If the algorithm finds that a particular branch can’t improve the final outcome for either player, it “prunes” that branch, meaning it stops further evaluation of that part of the tree. By doing so, the algorithm avoids unnecessary calculations, allowing it to search deeper into the tree more quickly. (Saxena)

### Key Terminologies

* Alpha:   
  Represents the best value the maximizing player can guarantee so far. The initial value of alpha is -Infinity (-∞).
* Beta: Represents the best value the minimizing player can guarantee so far. The initial value of Beta is +Infinity (+∞)

### Key Steps

Since Alpha-Beta Pruning is an extension of Min-Max algorithm, following steps are in addition to the steps followed by min-max algorithm.

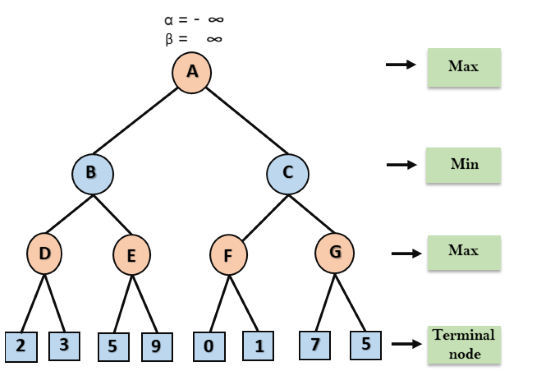
* For all maximizing players turn, update the Alpha value with node value.
* For all minimizing players turn, update the Beta value with node value.
* When backtracking the tree, node value will be passed to Alpha and Beta of upper node.
* When moving from top to bottom in the tree, pass the alpha and Beta value to all child nodes.

### Condition for Pruning

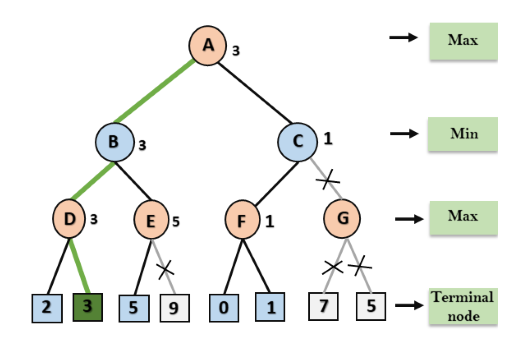
* The minimizing player finds a value that is lower than or equal to alpha (β ≤ α) then prune the branch.
* The maximizing player can prune a branch when they find a value higher than or equal to beta (α ≥ β)

### Example

Consider the below search tree with root node and 2 child nodes and terminal node.



Following the conditions and steps mentioned above, we arrive at the below result. Final result will be same after pruning four branches.



# Task2:

## Find the Best Route from Tipperary to Sligo

### Route Finding Using Depth-First Search (DFS)

The depth-first search or DFS algorithm explores data structures, such as trees and graphs. The algorithm starts at the root node or a given node and examines each branch as far as possible before backtracking.

The order in which DFS algorithm traverses is based on the order in which data is maintained in the stack.

Using python program, a dictionary list is maintained in which the locations are stacked in the below mentioned order.

If the order of Tipperary is modified, then the result could vary. This proves that result may not be optimal when there is more than one solution.



Route taken from Tipperary to Sligo is as below with a run time in the range of 0.1 millisecond.



### Route Finding Using Dijkstra’s Algorithm

Dijkstra’s Algorithm is used to find the shortest path between 2 given points in a positive weighted graph.

For the given map scenario, an optimal result is returned using Dijkstra’s algorithm and the execution time is negligible proving very efficient then DFS.



### Route Finding Using A\* Algorithm

A\* Search Algorithm is a simple and efficient search algorithm that can be used to find the optimal path between two nodes in a graph. It is an extension of Dijkstra’s shortest path algorithm (Dijkstra’s Algorithm). The A\* Search Algorithm uses a heuristic function that provides additional information regarding how far away from the goal node we are.

A major drawback of this algorithm is its space and time complexity. It takes a large amount of space to store all possible paths and a lot of time to find them. (Ravikiran AS)

For the given map scenario, the result of A\* algorithm matches with Dijkstra’s result but the actual run time to complete this execution is greater than DFS and Dijkstra’s.



## Conclusion

When the scope is small and non-optimal results are accepted, DFS algorithm can be used. However, if optimal result is expected, then the recommendations is to use Dijkstra’s algorithm or A\* algorithm if there is no negative weight in the graph.  
To use A\* algorithm, additional information namely heuristic value is required. It must be noted that inefficient heuristic value could result in poor performance.  
For the given scenario, considering the information available and performance, it is best to use Dijkstra’s algorithm.

## Two AI Strategies Used in Modern Games

**Introduction:**

AI for gaming refers to the integration of artificial intelligence techniques and technologies into video games to create more dynamic, responsive, and immersive gameplay experiences. It involves programming computer-controlled characters (non-player characters or NPCs) and entities within the game environment to exhibit intelligent behaviours, make decisions, and interact with the player and the game world in a lifelike manner.  
You know those opponents in a game that seem to adapt and challenge you differently each time? That's AI at work, crafting opponents that can think on their feet. (Engati)

### Monte Carlo Tree Search (MCTS) in AlphaGO Zero:

**What is MCTS:**  
Monte Carlo tree search is a method that relies on intelligent tree search that balances exploration and exploitation. It performs random sampling in the form of simulations and stores the statistics of actions to make more educated choices in each subsequent iteration.

Monte Carlo tree searches a few layers deep into the tree and prioritizes which parts of the tree to explore. It then simulates the outcome rather than exhaustively expanding the search space. In doing so, it limits how many evaluations it must make.

It is known for its ability to effectively handle complex and strategic video games with massive search areas, in which traditional algorithms may additionally struggle due to the full-size number of feasible actions.  
(Radke)

**MCTS in AlphaGo Zero:**  
In a Go game, AlphaGo Zero uses MC Tree Search to build a local policy to sample the next move.  
MCTS searches for possible moves and records the results in a search tree. As more searches are performed, the tree grows larger as well as its information. To make a move in Alpha-Go Zero, 1,600 searches will be computed. Then a local policy is constructed. Finally, we sample from this policy to make the next move.

In MCTS, a node represents a board position, and an edge represents a move.  
And for a given position, we can compute:  
The policy p (p is a probability distribution scoring each action), and  
The value function v (how likely to win at a board position)  
using a deep network f. (Hui)

### Behavior Tree Algorithm in Halo 2

**What is Behavior Tree:**

A Behavior Tree is a data structure that can make decisions based on a set of predefined conditions. They were originally invented for making AI behavior more modular in video games but have also started to get more popular in the robotics field as well.  
A Behavior Tree is formed using nodes that are organized in a hierarchical tree structure. Each node encapsulates a specific behavior that is not dependent on other nodes, allowing modularity and fast iteration. However, they share one main functionality: their status. All behavior trees use, at least, the following statuses:

* Success
* Failure
* Running

(Denis)

**Behavior Tree for Decision Making in Halo 2:**  
In the game Halo 2, each AI player has four states of the combat cycle namely,  
Idle  
Guard/Patrol  
Attack/Defend  
Retreat.

Using various gaming condition, behavior tree selects the right state for the AI player.

(Valdes)

# Task3:

Robot AIBO

**Introduction**:

The Aibo robot dog is one of the most advanced AI-powered robotic pets available today. Created by Sony, this interactive robot pet is designed to mimic the behaviour of a real dog, offering a lifelike experience with the power of artificial intelligence. Aibo comes equipped with advanced sensors, cameras, and artificial intelligence and thereby responds to touch, voice commands, and can even recognize faces.

## Characteristics of AIBO

**Machine Learning**:  
Just like humans and animals, Aibo has capabilities to understand where it is or whom it is with, and acts based on its feelings and instincts. Like a puppy, it may not behave as it is told by the owner, because it makes its own decisions through regular interaction, that is why the bond with people is so important.

**Natural Language Processing(NLP):**  
Aibo can understand a range of voice commands, recognizing basic phrases, and reacting to them appropriately. It can learn common commands like "Sit down" or "Come on" and associate them with specific actions.

**Environment Mapping and Navigation:**   
Aibo is equipped with advanced cameras and sensors that allow it to navigate its environment without bumping into objects. Aibo can create a map of its environment there by explore the surrounding intelligently and it can also detect a safe pathway.

**Recognize Emotion:**  
Aibo can detect emotional cues through voice tone and facial expressions. For example, When Aibo see a smile on its owner's face or it is petted on head, back, or under the chin, Aibo takes it as a compliment and looks delighted. While it may behave more gently if it detects stress or sadness.  
When you tap your Aibo on the back, Aibo takes it as “a scolding.”

**Cloud Connectivity:**  
Aibo can keep records of everything it experiences in day-to-day life, uploads the data to the cloud, and creates a database of memories that you can browse with the My Aibo app. You can even ask Aibo to take a picture—and you'll be able to preserve that moment for posterity.

(Sony Group Corporation)

## AIBO – Scope for Improvement

**Unsupervised Learning**:  
Aibo’s adaptability is restricted by its programmed algorithms. It lacks unsupervised learning capability which limits its ability to react to unforeseen situations.

**Lacks Advanced Conversational Abilities**:  
Though Aibo has the characteristics to process natural language, it is limited to specific commands and cannot follow more than what is in the manual. Thereby lacking the real NLP capability.   
(Sony Group Corporation)

# References

(Rolls)

Rolls, Chris. “LinkedIn.” *Linkedin.com*, 14 July 2024, www.linkedin.com/pulse/thoughts-ai-chess-quality-engineering-chris-rolls-5eehc/. Accessed 8 Nov. 2024.

Available at:   
<https://www.linkedin.com/pulse/thoughts-ai-chess-quality-engineering-chris-rolls-5eehc/>

(Degni)

Degni, Riccardo. “The Ultimate Checkmate: AI and Chess Engines.” *Codemotion Magazine*, 20 Feb. 2023, www.codemotion.com/magazine/ai-ml/the-ultimate-checkmate-ai-and-chess-engines/.

Available at:   
<https://www.codemotion.com/magazine/ai-ml/the-ultimate-checkmate-ai-and-chess-engines/>

(GeeksforGeeks)

GeeksforGeeks. “Mini-Max Algorithm in Artificial Intelligence.” *GeeksforGeeks*, GeeksforGeeks, 27 June 2024, www.geeksforgeeks.org/mini-max-algorithm-in-artificial-intelligence/.

Available at:   
<https://www.geeksforgeeks.org/mini-max-algorithm-in-artificial-intelligence/>

(Saxena)

Saxena, Abhimanyu. “Alpha Beta Pruning in Artificial Intelligence.” *Applied AI Blog*, 25 Sept. 2024, www.appliedaicourse.com/blog/alpha-beta-pruning-in-artificial-intelligence/. Accessed 9 Nov. 2024.

Available at:   
<https://www.appliedaicourse.com/blog/alpha-beta-pruning-in-artificial-intelligence/>

(Ravikiran AS)

Ravikiran AS. “A\* Algorithm in Artificial Intelligence You Must Know in 2022.” *Simplilearn.com*, 22 July 2021, www.simplilearn.com/tutorials/artificial-intelligence-tutorial/a-star-algorithm.

Available at:   
<https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/a-star-algorithm>

(Sony Group Corporation)

Sony Group Corporation. “ERS-1000 | Help Guide | Getting to Know Aibo.” *Sony.net*, 2018, helpguide.sony.net/aibo/ers1000/v1/en-us/contents/TP0001970093.html. Accessed 10 Nov. 2024.

Available at:   
<https://helpguide.sony.net/aibo/ers1000/v1/en-us/contents/TP0001970093.html>

(Engati)

Engati. “What Is AI in Gaming Industry (40+ AI Powered Games in 2024 ).” *Engati*, 11 July 2024, www.engati.com/blog/ai-for-gaming#toc-what-are-the-benefits-of-ai-in-games-. Accessed 10 Nov. 2024.

Available at:   
[https://www.engati.com/blog/ai-for-gaming#toc-what-are-the-benefits-of-ai-in-games-](https://www.engati.com/blog/ai-for-gaming%23toc-what-are-the-benefits-of-ai-in-games-)

(Radke)

Radke, Parag. “Monte Carlo Tree Search: A Guide | Built In.” *Builtin.com*, 20 July 2023, builtin.com/machine-learning/monte-carlo-tree-search.

Available at:   
<https://builtin.com/machine-learning/monte-carlo-tree-search>

(Hui)

Hui, Jonathan. “Monte Carlo Tree Search (MCTS) in AlphaGo Zero.” *Medium*, 17 July 2020, jonathan-hui.medium.com/monte-carlo-tree-search-mcts-in-alphago-zero-8a403588276a.

Available at:   
<https://jonathan-hui.medium.com/monte-carlo-tree-search-mcts-in-alphago-zero-8a403588276a>

(Denis)

Denis, Sion. “The Craft of Behavior Trees - Sion Denis - Medium.” *Medium*, 25 Jan. 2024, medium.com/@sion\_denis/the-craft-of-behavior-trees-ab6b181ce21a. Accessed 10 Nov. 2024.

Available at:   
<https://medium.com/@sion_denis/the-craft-of-behavior-trees-ab6b181ce21a>

(Valdes)

Valdes, Robert. “The Artificial Intelligence of Halo 2.” *HowStuffWorks*, 17 Nov. 2004, electronics.howstuffworks.com/halo2-ai.htm.

Available at:   
<https://electronics.howstuffworks.com/halo2-ai.htm>

# GitHub Link

<https://github.com/santhosh-sba24100/CA1---AI-Concepts-to-Implementation>