**CS4386 Assignment 1 (Semester B, 2022-2023)**

**Name: LUO Peiyuan SID: 56642728**

Table of Contents

[Utility Functions 2](#_Toc132400220)

[**Board Class** 2](#_Toc132400221)

[**makeMove function** 2](#_Toc132400222)

[**check\_total\_distance\_from\_sheep\_to\_wolf** 3](#_Toc132400223)

[**calculate\_shortened\_distance\_by\_current\_move** 3](#_Toc132400224)

[**check\_num\_of\_ways\_wolf\_trapped\_and\_num\_of\_wolf\_trapped** 3](#_Toc132400225)

[**check\_num\_of\_sheep\_and\_num\_of\_to\_be\_killed** 4](#_Toc132400226)

[**calculate\_sheep\_scores** 4](#_Toc132400227)

[**calculate\_wolf\_scores** 5](#_Toc132400228)

[**check\_wolf\_trapped\_in\_this\_way** 5](#_Toc132400229)

[**getWolfMoves** 5](#_Toc132400230)

[**getSheepMoves** 5](#_Toc132400231)

[**game\_ends function** 6](#_Toc132400232)

[Methodology 6](#_Toc132400233)

[**AB\_Minmax** 6](#_Toc132400234)

[**Objective** 6](#_Toc132400235)

[**Implementation** 6](#_Toc132400236)

[**Heuristic Evaluation** 8](#_Toc132400237)

[**Wolf Heuristic Function** 8](#_Toc132400238)

[**Sheep Heuristic Function** 10](#_Toc132400239)

[APPENDIX 11](#_Toc132400240)

# **Utility Functions**

## **Board Class**

This class stores some utility function related to the game board checking as follows. And each board object stores the current player of the board, state matrix, and winner of this board(if applicable).

Text

Description automatically generated

### **makeMove function**

This method would take the move as input, change the game state, update the next player and return a new board object.

Text

Description automatically generated

### **check\_total\_distance\_from\_sheep\_to\_wolf**

Return the total distance from sheep to wolf of current state.

**Objective:**  for the later heuristic evaluation function

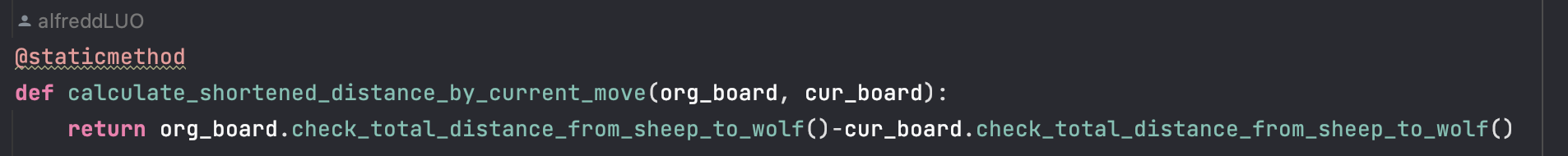
Text

Description automatically generated

### **calculate\_shortened\_distance\_by\_current\_move**

Return the difference of the total distance of all wolves and sheep between the original board and current board. If the returned value is positive, then the distance between wolf and sheep are shortened. Otherwise, the distance increased.

**Objective:**  for the later heuristic evaluation function



### **check\_num\_of\_ways\_wolf\_trapped\_and\_num\_of\_wolf\_trapped**

**Implemntation:** check the current state, and output the number of ways that wolf’s next move are trapped (max 4 for each wolf) and number of wolf that is already trapped(max 2).

\*trapped: the wolf can’t move in this direction

**Objective:**  for the later heuristic evaluation function

Text

Description automatically generated

### **check\_num\_of\_sheep\_and\_num\_of\_to\_be\_killed**

return the number of sheep at this state and the number of sheep that can be killed at next step(one empty column away from wolf).

**Objective:**  for the later heuristic evaluation function

Text

Description automatically generated

### **calculate\_sheep\_scores**

calculate the heuristic value of the sheep when the board is not yet end but reach the max depth.

**Objective:**  for the later heuristic evaluation function

Text

Description automatically generated

### **calculate\_wolf\_scores**

calculate the heuristic value of the wolf when the board is not yet end but reach the max depth.

**Objective:**  for the later heuristic evaluation function

Text

Description automatically generated

### **check\_wolf\_trapped\_in\_this\_way**

check whether wolf is trapped in this way.

Objective: used in check\_num\_of\_ways\_wolf\_trapped\_and\_num\_of\_wolf\_trapped function.

Text

Description automatically generated

### **getWolfMoves**

Return the valid moves for wolves.

### **getSheepMoves**

Return the valid moves for sheep.

### **game\_ends function**

Return whether the game ends or not.

Text

Description automatically generated

# **Methodology**

## **AB\_Minmax**

### **Objective**

I use AB\_Minmax algorithm as the framework of my code structure.

To simulate several rounds of the game and use the max value of the evaluation result for my turn and find the minimum of the evaluation result for opponent’s turn. And by using this, we would see a bigger picture when we make decision. However, with limited time, we should limit the max depth of the ab minmax, and also use alpha beta pruning approach to reduce running time.

### **Implementation**

**Max Depth:** 4 for Wolf, 3 for Sheep

**Location:** I use ab minmax function inside getBestMove() function to find the best move using ab minmax method.

**All\_moves:** I generate the valid moves list accoding to player, and if board.currentPlayer() ==2 it would return all valid moves of wolves, else return all valid moves of sheep.

**Ending condtion:** to end the recursion function of ab minmax, the triggering condition is either game ends(one of the player wins) or currentDepth==maxDepth.

Text

Description automatically generated

Text

Description automatically generated

## **Heuristic Evaluation**

I used the heuristic evaluation method for the evaluation function of the game.

Exception: Under the condition that the game ends at this state: if the winner is current player, then return 100000, otherwise -100000. If the game won’t end at this state, then return the heuristic function for the player.

Text

Description automatically generated

### **Wolf Heuristic Function**

#### **Overview**

The below is the main calculation related to Wolf Heuristic Function.

Text

Description automatically generated

Text

Description automatically generated

#### **Parameters**

##### **Parameters’ Importance Ranking**

num\_of\_sheep\_killed>num\_of\_to\_be\_killed\_sheep>shorten\_distance\_from\_sheep\_to\_wolf

##### **How to distinguish their differences**

By providing different weight to these parameters when do calculation, it would make the influence of each parameter differs a lot.

##### **Parameters that are taken into account**

###### **num\_of\_sheep\_killed**

* meaning: the number of sheep that are killed between the previous taken move and current depth, calculated by the number of the sheep before this turn minus the number of sheep in current state.
* Weight: 500
* Importance: very high, because the more sheep is eaten by wolf, the higher probability the wolf would win.

###### **num\_of\_to\_be\_killed\_sheep**

* meaning: after all the moves made, the number of sheep that can be killed by the wolf in next turn of wolf, calculated by counting the number of sheep that are only one empty column away from the wolf.
* Weight: 300
* Importance: high, because the more sheep can be killed by wolf in next move, the higher probability the wolf would win under this situation. But the weight should be lighter than the 500.

###### **shorten\_distance\_from\_sheep\_to\_wolf**

* meaning: the shortened distance between all wolves and sheep after all the move is made, calculated by the distance of sheep and wolves before this turn minus the distance of sheep in current state.
* Weight: 20
* Importance: high, as the closer the wolf is to sheep, the higher probability the wolf would win.

### **Sheep Heuristic Function**

#### **Overview**

The below is the main calculation related to Sheep Heuristic Function.

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

#### **Parameters**

##### **Parameters’ Importance Ranking**

num\_of\_sheep\_killed > trapped\_wolf\_num > shorten\_distance\_from\_sheep\_to\_wolf

>num\_of\_trapped\_ways > num\_of\_to\_be\_killed\_sheep

##### **How to distinguish their differences**

By providing different weight to these parameters when do calculation, it would make the influence of each parameter differs a lot.

##### **num\_of\_sheep\_killed:**

* meaning: the number of sheep that are killed between the previous taken move and current depth, calculated by the number of the sheep before this turn minus the number of sheep in current state.
* Weight: -20000
* Importance: very high, because the less sheep is eaten by wolf, the higher probability the sheep would win.

##### **trapped\_wolf\_num:**

* meaning: the number of wolves that are trapped by sheep(can’t move any more). Each wolf has four ways (above, below, right, left). If all four ways of a wolf are trapped, then we say the wolf is trapped.
* Weight: 4000
* Importance: high, the more wolves that are trapped, then the less dangerous for sheep, and the more probability for sheep to trap the wolf and wins.

##### **shorten\_distance\_from\_sheep\_to\_wolf:**

* meaning: the shortened distance between all wolves and sheep after all the move is made, calculated by the distance of sheep and wolves before this turn minus the distance of sheep in current state.
* Weight: -100
* Importance: high, as the farer the sheep is wolves, the higher probability the wolf would win, because the wolf would take more moves to get close to sheep.

##### **num\_of\_trapped\_ways:**

* meaning: the number of ways that the wolf’ move is trapped. Each wolf has four ways (above, below, right, left). If a ways is trapped then the wolf can’t move in this direction.
* Weight: 800
* Importance: high, the more ways that the wolves are trapped, then the more limit for wolves’ move, then the less probability for wolf to win, and the more probability for sheep to trap the wolf and wins.

##### **num\_of\_to\_be\_killed\_sheep:**

* meaning: after all the moves made, the number of sheep that can be killed by the wolf in next turn of wolf, calculated by counting the number of sheep that are only one empty column away from the wolf.
* Weight: 1
* Importance: very low. Because for wolf’s turn, it needs to wait for another turn to kill the to-be-killed sheep, which increase the uncertainty of this value. Hence, I only use this value to differ slightly.

# **APPENDIX**

My source code is already pushed to my personal github:

<https://github.com/alfreddLUO/AI-Algorithm-for-Wolves-Eats-Sheep-Game.git>