CS440: Intro to Artificial Intelligence, Fall 2017, Homework 2

Name: Nestor Alejandro Bermudez Sarmiento (nab6)
Credit hours: 4

Worked individually

In this assignment we have implemented general-purpose search algorithms to solve well know puzzles. In the first section we'll use such algorithms to 'guide' a 'Pacman' through the maze, finding all the food pellets. In the second section we'll solve the Sokoban puzzle and describe the chosen heuristic and why it is valid.

We've decided to use Python for this assignment. The source code is provided with this report and will also be available under the following GitHub account right after the submission deadline had passed. See Nestor's GitHub CS440 repository.

Part 1

Part 2: Breakthrough

Just like in the previous assignment we will use bitmaps to represent the state of the board. Having two bitmaps, one per player, of size equals to the size of the board. A value of 1 indicates that the piece of that player is in a given position.

We implemented a **Game** class inspired in the code for the textbook[1]. The class is a simple empty skeleton that specifies which methods a game must implement. We then implemented a **Breakthrough** class that extends **Game** and implements all its methods. There is a function to generate all the possible actions for a given player given the current state of the board and, more importantly, there is a function that determines when a state is terminal. The **Breakthrough** class can handle any board size and we use it for the extended rule of using a rectangular board.

For the game with the extended rule to take 3 workers home we created a new class called **Breakthrough3WorkersToBase** which inherits everything from the previous **Breakthrough** class but overrides the *terminal_test* function.

From now on whenever we may refer to the *state* as the *board* and vice versa. Note that the board has been implemented as a class called **BreakthroughBoard** and it is responsible of actually moving the pieces/capturing based on an original and final position.

Heuristic

All the heuristic functions can be found under the **Heuristics** class. Each of the methods is expected to receive two arguments, the first one is the game instance and the second one is the current state.

The implementation of the two given heuristics is rather simple. For **Defensive 1** we simply count how many ones are in the corresponding bitmap of our state representation. Once we have the count we just apply the formula provided. The same goes for **Offensive 1** with the only difference that you need to look at the other bitmap.

For **Defensive 2** we implemented a weighted function that considers these features of the board:

1. number of protected cells (both ours and the opponent's): this is the number of cells that can be reached from one of the pieces in a single move and that are not occupied by another piece.
2. row formations: this is when at least two pieces are next to each other.
3. delta formations: this is when a given piece has two pieces diagonally behind it.
4. base cells: how many pieces are still in their base. We value this because they are our last line of defense.

[1] https://github.com/aimacode/aima-python

We played a little bit with the weight of each of the features and we ended up with the best results when:

1. the score is punished for each unprotected cell. An unprotected cell is one that is under attack by the opponent but is it defended by one of your own pieces. For each of those cells we reduce our score by 100.
2. for each of the delta formations we add 75 points to our board score and reduce 75 for each enemy delta formation. We value these formations this high because if a piece is captured we can always retaliate.
3. protected cells add 65 points each. Cells protected by the enemy reduces 65 points.

Besides those features we also check for pieces that are about to win, on both sides, and add or subtract from the score accordingly.

For **Offensive 2** we also implemented a weighted function of features. These are the features:

1. proximity to the goal: how far into the enemy territory a piece is.
2. arrow formations: created by a piece and all the cells behind it being occupied by friendly pieces.
3. almost win position: increase or decrease the score if board can be won/lost on the next move.

Just like for **Defensive 2** we played with the weights and settled with:

1. proximity: 5 points for each row closer to the goal
2. arrow formation: 20 points
3. almost win/lose: 1000 points

Analysis

Why **Defensive 2** works?
Our heuristic emphasizes coverage of the board and delta formations; since **Offensive 1** only depends on the remaining enemy pieces it is expected that it will try to capture whenever possible. This will immediately be punished if there is a delta formation unless it is about to win. Also, since we value protected cells, we will have more opportunities to capture a piece if it treating us.

Why **Offensive 2** works?
Note that our heuristic does not weight in the number of pieces on either side, this offensive strategy relies to getting as far without necessarily capturing pieces, this should allow us to reduce the change of triggering a response from **Defensive 1**.

Results 2.1

For each of the match-ups requested we played the game four times. The averages are shown below but the results for the individual match-ups can be found in the summary spreadsheet delivered with this report and also online[2].
In the zip file accompanying this report you will find the whole sequence of moves for every game summarized in this report, as well as the stats for each player. See the *results* directory. Any time measurement is assumed to be in seconds.

1. Minimax (Offensive 1) vs Alpha-beta (Offensive 1)
Minimax used depth 4 and Alpha-beta used depth 5. Game results:

1. Average time per move for whites: **16.03**
2. Average time per move for blacks: **0.91**
3. Average # nodes expanded per move for whites: **413,445.82**
4. Average # nodes expanded per move for blacks: **21,177.43**
5. Total # nodes expanded by whites: **2,796,884.5**
6. Total # nodes expanded by blacks: **143,064**
7. Total # of moves: **13.5**
8. White pieces captured: **0.25**
9. Black pieces captured: **1.5**
10. Winner: Whites

```
[B, W, W, W, W,  , W, W]
[W,  ,  ,  , W, W,  , W]
[ , W, W,  ,  ,  , W, W]
[ ,  ,  ,  ,  ,  ,  ,  ]
[ ,  ,  ,  ,  ,  ,  ,  ]
[ ,  , B,  ,  ,  ,  ,  ]
[ ,  , B, B, B, B, B, B]
[B, B, B, B, B, B, B, B]
```

Final state of the board.

[2] https://docs.google.com/a/illinois.edu/spreadsheets/d/1304kvtHACh1N3kYYjGNfJnYtcIc5_p1jHPop4sn356I/edit?usp=sharing

Trends: in all the games the Alpha-beta player won. Which makes since it can see one more move ahead.

2. Alpha-beta (Offensive 2) vs Alpha-beta (Defensive 1)
This was run with depth of 4 for both players.
Game results:

1. Average time per move for whites: **0.66**
2. Average time per move for blacks: **0.25**
3. Average # nodes expanded per move for whites: **7,544.1**
4. Average # nodes expanded per move for blacks: **4,403.9**
5. Total # nodes expanded by whites: **189,886.75**
6. Total # nodes expanded by blacks: **102,276.5**
7. Total # of moves: **49.5**
8. White pieces captured: **0**
9. Black pieces captured: **2.75**
10. Winner: Whites

```
[ ,  ,  ,  ,  ,  ,  ,  ]
[ ,  ,  ,  ,  ,  , W,  ]
[ ,  , B, W, W, W, W, W]
[ ,  , W, W, W, W, W, W]
[W, W, W,  ,  , B,  , B]
[ , B, B,  ,  , B,  ,  ]
[ ,  ,  ,  ,  ,  ,  , B]
[ ,  , W,  ,  ,  , B,  ]
```

Final state of the board.

Trends: Our heuristic won 3 out of the 4 games played. It was nice to see that the formation that we intended to see in the board actually appeared. Here is an example:

```
[W,  ,  ,  ,  ,  ,  , W]
[W, W,  , W, W, W, W,  ]
[ , W, W, W, W, W, W, W]
[ , W,  ,  , B, B,  ,  ]
[ ,  ,  ,  ,  ,  , B,  ]
[B,  , B,  , B,  ,  , B]
[B,  , B, B,  ,  ,  , B]
[B,  ,  ,  , B, B, B, B]
```

See how the white pieces advance but don't try to go on their own and always have "backup".

3. Alpha-beta (Defensive 2) vs Alpha-beta (Offensive 1)
This was run with depth of 4 for both players.
Game results:

1. Average time per move for whites: **55.39**
2. Average time per move for blacks: **3.34**
3. Average # nodes expanded per move for whites: **10,067.28**
4. Average # nodes expanded per move for blacks: **2,514.29**
5. Total # nodes expanded by whites: **160,601.75**
6. Total # nodes expanded by blacks: **45,653.75**
7. Total # of moves: **36**
8. White pieces captured: **0**
9. Black pieces captured: **2**
10. Winner: Whites

```
[W, W, W, W, W, W, W, W]
[ ,  ,  , W, W,  , W, W]
[ , W, W,  ,  ,  ,  ,  ]
[ ,  ,  ,  ,  ,  , W,  ]
[ ,  ,  ,  ,  ,  ,  ,  ]
[B,  ,  , B,  ,  ,  , B]
[B,  , B,  ,  ,  , B,  ]
[B, B, B, B, B, B, W,  ]
```

Final state of the board.

Trends: Our heuristic won 4 out of the 4 games played. The delta formations we were expecting to see showed up most of the times. Here is an example:

```
[W,  , W,  , W, W, W, W]
[W, W, W, W,  , W,  , W]
[W,  , W,  , W,  , W,  ]
[ ,  ,  ,  ,  ,  ,  ,  ]
[ ,  ,  ,  ,  ,  , B,  ]
[B,  ,  ,  , B, B,  ,  ]
[ , B, B,  ,  ,  , B, B]
[B, B, B, B, B, B, B, B]
```

We realize this formation would be bad for a smarter opponent (and it is confirmed later on in this report) because of the "holes" in the delta formation but it turns out to work well against the provided defensive heuristic.

One thing to note is that for this match-up the second game took way longer than the other games. Time between moves was around 2 minutes. Our only guess is that it was caused by the fact that we were running a script for our data mining course at the same time. We could never replicate it after that.

4. Alpha-beta (Offensive 2) vs Alpha-beta (Offensive 1)
This was run with depth of 4 for both players.
Game results:

1. Average time per move for whites: **0.51**
2. Average time per move for blacks: **0.17**
3. Average # nodes expanded per move for whites: **6,463.52**
4. Average # nodes expanded per move for blacks: **3,183.71**
5. Total # nodes expanded by whites: **197,461.5**
6. Total # nodes expanded by blacks: **95,030.75**
7. Total # of moves: **60.25**
8. White pieces captured: **0.25**
9. Black pieces captured: **4.75**
10. Winner: Whites

```
[ ,  ,  ,  ,  ,  ,  ,  ]
[ ,  ,  ,  ,  ,  ,  , W]
[ ,  , W, W, B, W, W, W]
[W,  , W, B,  ,  , W,  ]
[ ,  , W,  ,  ,  ,  ,  ]
[ ,  ,  ,  ,  ,  ,  ,  ]
[ , W,  , W,  , W,  , W]
[W,  ,  ,  ,  ,  ,  ,  ]
```

Final state of the board.

Trends: at first we didn't expect **Offensive 2** to beat **Offensive 1** but after looking at the results it does make sense. The reason is that almost always when **Offensive 1** would take a piece the opponent would have a way to respond because of the value we gave to "moving as a front". As a result of this, this match-up has the highest number of pieces captured on average.

5. Alpha-beta (Defensive 2) vs Alpha-beta (Defensive 1)
This was run with depth of 4 for both players.
Game results:

1. Average time per move for whites: **1**
2. Average time per move for blacks: **0.22**
3. Average # nodes expanded per move for whites: **6,111.88**
4. Average # nodes expanded per move for blacks: **3,524.07**
5. Total # nodes expanded by whites: **136,086.25**
6. Total # nodes expanded by blacks: **69,990.5**
7. Total # of moves: **41.5**
8. White pieces captured: **0**
9. Black pieces captured: **2**
10. Winner: Whites

```
[W,  , W,  , W,  , W, W]
[W, W, W, W,  , W,  , W]
[W,  , W,  , W,  , W,  ]
[ ,  ,  ,  ,  ,  ,  ,  ]
[ , B, B,  ,  , B,  ,  ]
[ , B,  ,  , B, B,  ,  ]
[B,  ,  ,  ,  , B, B,  ]
[ , B, B,  , B, B, W, B]
```

Final state of the board.

Trends: It makes sense for heuristic **Defensive 2** to beat **Defensive 1** because our heuristic still considers the value of the pieces as **Defensive 1** does but it also considers other features of the board.

6. Alpha-beta (Offensive 2) vs Alpha-beta (Defensive 2)
This was run with depth of 4 for both players.
Game results:

1. Average time per move for whites: **0.41**
2. Average time per move for blacks: **0.22**
3. Average # nodes expanded per move for whites: **6,010.24**
4. Average # nodes expanded per move for blacks: **1,680.6**
5. Total # nodes expanded by whites: **294,502**
6. Total # nodes expanded by blacks: **80,669**
7. Total # of moves: **97**
8. White pieces captured: **1**
9. Black pieces captured: **14**
10. Winner: Whites

```
[ ,  ,  ,  ,  ,  ,  ,  ]
[ ,  ,  ,  ,  ,  ,  , W]
[ ,  , W, W, B, W, W, W]
[W,  , W, B,  ,  , W,  ]
[ ,  , W,  ,  ,  ,  ,  ]
[ ,  ,  ,  ,  ,  ,  ,  ]
[ , W,  , W,  , W,  , W]
[W,  ,  ,  ,  ,  ,  ,  ]
```

Final state of the board.

Trends: as mentioned earlier, **Defensive 2** heuristic isn't very smart because the delta formations leave an open path for the enemy.
Although we ran the match-up four times, the results were the same. This makes sense because there is no randomness on either side of the board.

Extended rules

For part 2.2 we were asked to add additional rules to the board. All the code was written thinking about these extended rules since the beginning so in order to play a match in a 10x5 board the only thing we needed to do was to change the size of the board argument in our **Breakthrough** class.
For the 3 workers to base rule we created a new class (**Breakthrough3WorkersToBase**) that extends the normal **Breakthrough** class and overrides the *terminal_test* method.

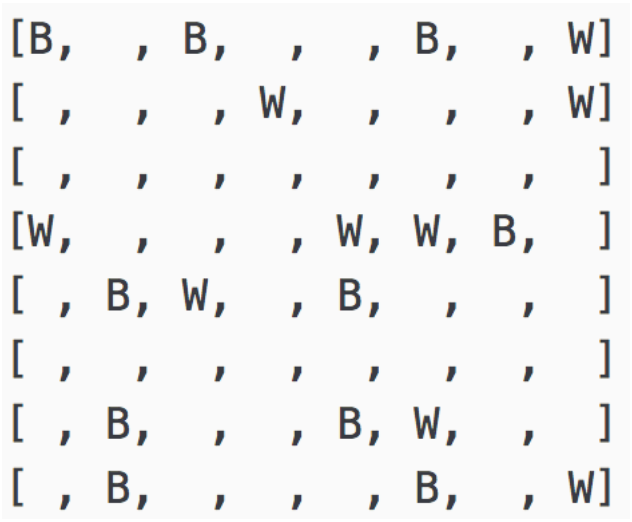
Lets look at the results.

Results 2.2

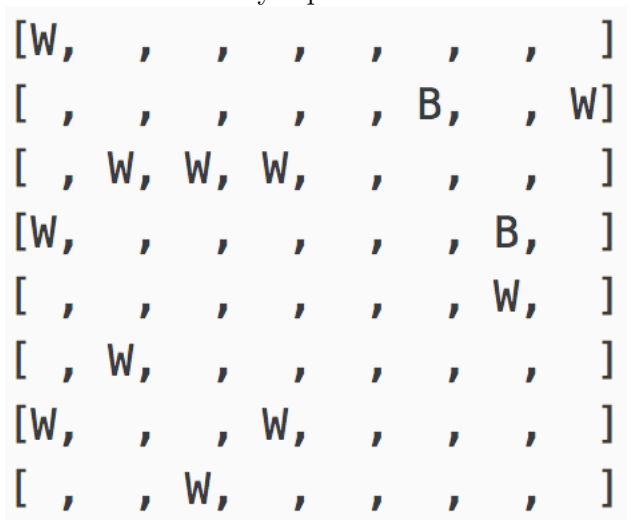
1. 3 workers to base
We decided to use **Offensive 1** and **Defensive 1** for the match-ups.

The games were run with depth of 5 for both players.
Game results:

1. Average time per move for whites: **3.21**
2. Average time per move for blacks: **3.1**
3. Average # nodes expanded per move for whites: **83,946.52**
4. Average # nodes expanded per move for blacks: **70,858.69**
5. Total # nodes expanded by whites: **3,938,242.75**
6. Total # nodes expanded by blacks: **3,304,344.7**
7. Total # of moves: **94.75**
8. White pieces captured: **7.75**
9. Black pieces captured: **9**
10. Winner: Blacks (3 out of 4 times)



Game won by 3 pieces at the base.



Game won by 3 pieces at the base.

One of the interesting things about this setup is that the only time Whites won was because they captured all black pieces not because they took 3 pieces to base. In contrast, all the games won by Blacks were won by taking 3 pieces to base and not by capturing all white pieces. In all cases, it seems like the player that captured the most pieces wins.

2. Rectangular board
The games were run with depth of 5 for both players.
Game results:

1. Average time per move for whites: **0.86**
2. Average time per move for blacks: **0.58**
3. Average # nodes expanded per move for whites: **23,419.22**

4. Average # nodes expanded per move for blacks: **14,520.39**
5. Total # nodes expanded by whites: **760,800.25**
6. Total # nodes expanded by blacks: **459,584**
7. Total # of moves: **65.5**
8. White pieces captured: **1.75**
9. Black pieces captured: **6.25**
10. Winner: Whites

```
[W,  , W,  ,  ]
[W, W, W,  ,  ]
[ ,  ,  ,  , W]
[ ,  ,  ,  ,  ]
[B, B,  ,  ,  ]
[ ,  , B, W,  ]
[B,  , W, B,  ]
[ ,  ,  ,  ,  ]
[ ,  ,  ,  , B]
[ , W,  ,  ,  ]
```

Final state of the board for one of the matches.

Potential Extra Credits

As you may have noticed the number of expanded nodes for matches between both alpha-beta players are relatively low. We implemented move ordering based on the board configuration at depth 1. We also tried to do the ordering on each level (inside the min-value and max-value functions) but it was taking too long to finish so we decided to keep it just in the first level.

We found in the book a method called *beam search* which consists of just cutting off some of the search. It relies on a strong move ordering heuristic. Since our games were not taking too long we decided not to implement it. May be worth it if we increase the depth of the search to a point where we can't tolerate to consider all the actions even with alpha-beta pruning. We will try this if time permits.

We also ran some games in a rectangular 10x5 (as opposed to 5x10). The results were this:

1. Average time per move for whites: **3.08**
2. Average time per move for blacks: **1.02**
3. Average # nodes expanded per move for whites: **79,968.26**
4. Average # nodes expanded per move for blacks: **24,608.09**
5. Total # nodes expanded by whites: **1,329,205**
6. Total # nodes expanded by blacks: **317,786.25**
7. Total # of moves: **38.75**
8. White pieces captured: **4.5**
9. Black pieces captured: **14**
10. Winner: Whites (3 out of 4 times)

```
[ ,  ,  , W, W,  , W,  ,  ,  ]
[W,  ,  , W,  ,  , W, W, B, W]
[ ,  , W, W,  , W,  ,  ,  ,  ]
[W, W,  ,  , W,  ,  , W,  ,  ]
[ ,  ,  ,  ,  ,  ,  ,  , W,  ]
```

Final state of the board for one of the matches.

The interesting thing about this board is that it caused the players to "think more", pretty much all the metrics are higher in this board than in the 5x10 board. This may be simply due to the fact that there are twice as many pieces which means that the number of actions at each move is higher.