Adversarial Game Playing Agent for Isolation

Submitted by: Vadym Serpak, v.serpak@gmail.com

Course: Artificial Intelligence Nanodegree, Udacity

Date: 06/21/2017

1. Synopsis

In this project, we will develop an adversarial search agent to play the game "Isolation". Isolation is a deterministic, two-player game of perfect information in which the players alternate turns moving a single piece from one cell to another on a board. Whenever either player occupies a cell, that cell becomes blocked for the remainder of the game. The first player with no remaining legal moves loses, and the opponent is declared the winner.

This project uses a version of Isolation where each agent is restricted to L-shaped movements (like a knight in chess) on a rectangular grid (like a chess or checkerboard). The agents can move to any open cell on the board that is 2-rows and 1-column or 2-columns and 1-row away from their current position on the board. Movements are blocked at the edges of the board (the board does not wrap around), however, the player can "jump" blocked or occupied spaces (just like a knight in chess).

Additionally, agents will have a fixed time limit each turn to search for the best move and respond. If the time limit expires during a player's turn, that player forfeits the match, and the opponent wins.

2. Implementation

2.1. Adversarial Search

We will first implement the *minimax* algorithm, then implement *alpha-beta pruning for minimax*, and finally incorporate *iterative deepening*.
We will implement them in the `game_agent.CustomPlayer` class under the `minimax()`, `alphabeta()`, and `get_moves()` methods, respectively. The class constructor takes arguments that specify whether to call *minimax* or *alphabeta* search, and whether to use fixed-depth search or iterative deepening. These will be used, along with a function that you will use to determine how much time remains before the search will time out, in the `get_moves()` method.

2.2. Evaluation Functions

We will develop heuristic functions to inform the value judgements our AI will make when choosing moves. We were provided with implementations in sample_players.py of the heuristic functions in the `OpenMoveEval` and `ImprovedEval` classes, as well as a `NullEval` class to use in testing. We will experiment with the `custom_score` to come up with at least three distinct heuristic functions that we will compare in the report. Please note that we are not required to find a *better* heuristic than those provided.
We were also provided with a script called `tournament.py` that we will use to evaluate and compare our heuristic functions by testing our agent & heuristic against agent configurations that have been specified in the tournament script. The script plays our agent against each one of the test agents - which have all been

ranked with a calibrated Elo score (a skill rating system used in many games) - to determine the relative strength of our heuristic and search algorithm.

3. Heuristic Evaluation Functions

The heuristic functions provided to us are described as follows:

1. **Null Score:** This heuristic presumes no knowledge for non-terminal states, and returns the same uninformative value for all other states.
2. **Open Move Score:** The basic evaluation function that outputs a score equal to the number of moves open for your computer player on the board.
3. **Improved Open Move:** The "Improved" evaluation function that outputs a score equal to the difference in the number of moves available to the two players.

As we can see, these functions are based on the difference in the number of moves, available to the active player and the opponent player in a given board state. Let's improve this strategy by some 'aggression' – introduce heuristic constant chosen empirically as 1.5 to minimize opponent moves. We will get the function **aggressive_heuristic**. For the **Tournament#1** we will assign:

custom_score - aggressive_heuristic (game, player)
custom_score_2 - open_move_score (game, player)
custom_score_3 - improved_score(game, player):

```
                    *************************
                         Playing Matches
                    *************************

Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                       Won | Lost   Won | Lost   Won | Lost   Won | Lost
   1       Random       9  |   1    10  |   0    8  |   2    9  |   1
   2       MM_Open      4  |   6     6  |   4    6  |   4    6  |   4
   3      MM_Center     9  |   1     6  |   4    9  |   1   10  |   0
   4     MM_Improved    4  |   6     4  |   6    4  |   6    5  |   5
   5       AB_Open      5  |   5     4  |   6    6  |   4    4  |   6
   6      AB_Center     6  |   4     7  |   3    5  |   5    5  |   5
   7     AB_Improved    5  |   5     5  |   5    3  |   7    5  |   5
--------------------------------------------------------------------------
          Win Rate:      60.0%        60.0%        58.6%        62.9%
```

We can see that the implementation of aggressive_heuristic did not improve the result significantly.

Another heuristic idea is based on the 'weights' of each position of a player. We assume that the bigger number of the possible moves the better. We will modify the **Open moves score** heuristic provided to us by weighting each open move by a weight that depends on the player position in relation to the center of the board. The measure of each position weight is the amount of open moves in it. For the tournament board 7x7 we will get the table of the weights:

2	3	4	4	4	3	2
3	4	6	6	6	4	3
4	6	8	8	8	6	4
4	6	8	8	8	6	4
4	6	8	8	8	6	4
3	4	6	6	6	4	3
2	3	4	4	4	3	2

As we could suppose, the worst result in the tournament belongs to the **Open_move_score**, so we will implement the new heuristic function ***weighted_open_move_score(game, player)*** *in the custom_score_2 in the* ***Tournament#2****:*

custom_score - aggressive_heuristic (game, player)
custom_score_2 - *weighted_open_move_score* (game, player)
custom_score_3 - improved_score(game, player):

```
                    ************************
                         Playing Matches
                    ************************

Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                       Won | Lost   Won | Lost   Won | Lost   Won | Lost
   1       Random       9  |   1     9  |   1    10  |   0     8  |   2
   2       MM_Open      6  |   4     4  |   6     6  |   4     4  |   6
   3      MM_Center     6  |   4     8  |   2     8  |   2     7  |   3
   4     MM_Improved    6  |   4     3  |   7     2  |   8     6  |   4
   5       AB_Open      4  |   6     7  |   3     5  |   5     5  |   5
   6      AB_Center     6  |   4     5  |   5     6  |   4     6  |   4
   7     AB_Improved    6  |   4     7  |   3     5  |   5     3  |   7
--------------------------------------------------------------------------
          Win Rate:      61.4%        61.4%        60.0%        55.7%
```

We can see that result became a little bit better. The worst result belongs to the **improved_score(game, player)** as we could suppose**.** Let's implement 'weight' technique for it and calculate the difference in the weighted open moves scores between the current player and its opponent with **weighted_improved_score in the custom_score_3** in the **Tournament#3:**

custom_score - aggressive_heuristic (game, player)
custom_score_2 - *weighted_open_move_score* (game, player)
custom_score_3 - weighted_improved_score (game, player):

```
                    ************************
                         Playing Matches
                    ************************

Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                       Won | Lost   Won | Lost   Won | Lost   Won | Lost
   1       Random       9  |   1     8  |   2     9  |   1     9  |   1
   2       MM_Open      9  |   1     7  |   3     7  |   3     7  |   3
   3      MM_Center     8  |   2     7  |   3     8  |   2     9  |   1
   4     MM_Improved    5  |   5     8  |   2     3  |   7     6  |   4
   5       AB_Open      6  |   4     4  |   6     5  |   5     5  |   5
   6      AB_Center     6  |   4     5  |   5     5  |   5     5  |   5
   7     AB_Improved    6  |   4     5  |   5     3  |   7     4  |   6
--------------------------------------------------------------------------
          Win Rate:      70.0%        62.9%        57.1%        64.3%
```

We will continue with the implementation of the heuristic evaluation function **diff_my_moves_opp_moves_one_ply_lookahead(game, player)**, which calculate the difference in the number of available moves between the current player and its opponent one ply ahead in the future and use it as the score of the current game state in the **Tournament#4**:

custom_score - diff_my_moves_opp_moves_one_ply_lookahead(game, player)
custom_score_2 - weighted_improved_score(game, player)
custom_score_3 - aggressive_heuristic(game, player):

```
                    **************************
                        Playing Matches
                    **************************

Match #    Opponent     AB_Improved    AB_Custom    AB_Custom_2  AB_Custom_3
                         Won | Lost   Won | Lost   Won | Lost   Won | Lost
   1        Random       10  |  0     9  |  1     9  |  1     9  |  1
   2        MM_Open       7  |  3     7  |  3     7  |  3     5  |  5
   3       MM_Center      5  |  5     8  |  2     9  |  1     9  |  1
   4     MM_Improved      6  |  4     6  |  4     4  |  6     3  |  7
   5        AB_Open       6  |  4     6  |  4     5  |  5     7  |  3
   6       AB_Center      5  |  5     5  |  5     5  |  5     6  |  4
   7     AB_Improved      5  |  5     6  |  4     6  |  4     8  |  2
-------------------------------------------------------------------------
          Win Rate:      62.9%        67.1%        64.3%        67.1%
```

We can see that some forecasting improved our results significantly. Along all tournaments we could see the good result of the 'aggressive' strategy constantly. Let's implement it for the **diff_my_moves_opp_moves_one_ply_lookahead(game, player)** and combine aggression in the difference in players moves and some forecasting. But in this case our heuristic constant chosen empirically as 1.5 will be maximizing player's moves. As usually we will change the weakest strategy on a new one **aggressive_diff_moves_one_ply_lookahead (game, player)** in **Tournament#5**:

custom_score - diff_my_moves_opp_moves_one_ply_lookahead(game, player)
custom_score_2 - aggressive_diff_moves_one_ply_lookahead (game, player)
custom_score_3 - aggressive_heuristic(game, player):

```
                    **************************
                        Playing Matches
                    **************************

Match #    Opponent     AB_Improved    AB_Custom    AB_Custom_2  AB_Custom_3
                         Won | Lost   Won | Lost   Won | Lost   Won | Lost
   1        Random        9  |  1     9  |  1     9  |  1     9  |  1
   2        MM_Open       5  |  5     7  |  3     9  |  1     5  |  5
   3       MM_Center      9  |  1     7  |  3    10  |  0     9  |  1
   4     MM_Improved      6  |  4     5  |  5     7  |  3     6  |  4
   5        AB_Open       5  |  5     4  |  6     6  |  4     4  |  6
   6       AB_Center      6  |  4     5  |  5     6  |  4     4  |  6
   7     AB_Improved      6  |  4     5  |  5     4  |  6     3  |  7
-------------------------------------------------------------------------
          Win Rate:      65.7%        60.0%        72.9%        57.1%
```

```
                    **************************
                        Playing Matches
                    **************************

Match #    Opponent     AB_Improved    AB_Custom    AB_Custom_2  AB_Custom_3
                         Won | Lost   Won | Lost   Won | Lost   Won | Lost
   1        Random        7  |  3     9  |  1     9  |  1     9  |  1
   2        MM_Open       6  |  4     5  |  5     7  |  3     4  |  6
   3       MM_Center      6  |  4     8  |  2     8  |  2    10  |  0
   4     MM_Improved      3  |  7     3  |  7     5  |  5     6  |  4
   5        AB_Open       5  |  5     5  |  5     5  |  5     5  |  5
   6       AB_Center      7  |  3     4  |  6     3  |  7     6  |  4
   7     AB_Improved      5  |  5     6  |  4     7  |  3     4  |  6
-------------------------------------------------------------------------
          Win Rate:      55.7%        57.1%        62.9%        62.9%
```

```
                    *************************
                         Playing Matches
                    *************************

Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                       Won | Lost   Won | Lost   Won | Lost   Won | Lost
   1       Random       8  |  2     9  |  1     10 |  0     10 |  0
   2       MM_Open      4  |  6     6  |  4      8 |  2      5 |  5
   3      MM_Center     8  |  2     7  |  3      8 |  2     10 |  0
   4     MM_Improved    6  |  4     6  |  4      8 |  2      5 |  5
   5       AB_Open      5  |  5     7  |  3      5 |  5      5 |  5
   6      AB_Center     5  |  5     5  |  5      5 |  5      6 |  4
   7     AB_Improved    6  |  4     4  |  6      3 |  7      7 |  3
--------------------------------------------------------------------------
          Win Rate:      60.0%        62.9%        67.1%        68.6%
```

After several tournaments we can estimate the average results of the best heuristics for a while:

	Win rate	Won	Draw	Lost
aggressive_diff_moves_one_ply_lookahead (game, player)	- 67.6%	1	1	1
aggressive_heuristic(game, player)	- 62.9%	1	1	1

It means, that 'aggressive' heuristic could be estimated as very perspective. Let's check this in the combination of 'aggressive' and 'weighted improved' heuristics by implementing **aggr_weighted_improved_score(game, player)** in the **Tournament#6:**

custom_score - aggr_weighted_improved_score(game, player)
custom_score_2 - aggressive_diff_moves_one_ply_lookahead (game, player)
custom_score_3 - aggressive_heuristic(game, player):

```
                    *************************
                         Playing Matches
                    *************************

Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                       Won | Lost   Won | Lost   Won | Lost   Won | Lost
   1       Random       9  |  1     9  |  1     10 |  0     10 |  0
   2       MM_Open      5  |  5     6  |  4      7 |  3      4 |  6
   3      MM_Center     8  |  2     7  |  3      8 |  2      9 |  1
   4     MM_Improved    4  |  6     6  |  4     10 |  0      5 |  5
   5       AB_Open      6  |  4     6  |  4      6 |  4      4 |  6
   6      AB_Center     5  |  5     5  |  5      7 |  3      4 |  6
   7     AB_Improved    5  |  5     6  |  4      4 |  6      5 |  5
--------------------------------------------------------------------------
          Win Rate:      60.0%        64.3%        74.3%        58.6%
```

We received the same result without any improvement for 'improved' score. Let's combine 'aggressive' with 'weighted' and 'forecasted' in **aggress_diff_weight_moves_one_ply_lookahead(game, player)** in the **Tournament#7:**

custom_score - aggress_diff_weight_moves_one_ply_lookahead(game, player)
custom_score_2 - aggressive_diff_moves_one_ply_lookahead (game, player)
custom_score_3 - aggressive_heuristic(game, player):

```
                    ************************
                         Playing Matches
                    ************************

Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                      Won | Lost   Won | Lost   Won | Lost   Won | Lost
   1       Random      9  |  1     9  |  1     9  |  1     10 |  0
   2       MM_Open     6  |  4     8  |  2     8  |  2     7  |  3
   3      MM_Center    5  |  5     8  |  2     8  |  2     7  |  3
   4     MM_Improved   6  |  4     8  |  2     7  |  3     5  |  5
   5       AB_Open     6  |  4     5  |  5     4  |  6     6  |  4
   6      AB_Center    4  |  6     6  |  4     5  |  5     6  |  4
   7     AB_Improved   4  |  6     7  |  3     4  |  6     4  |  6
--------------------------------------------------------------------------
         Win Rate:      57.1%        72.9%        64.3%        64.3%
```

```
                    ************************
                         Playing Matches
                    ************************

Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                      Won | Lost   Won | Lost   Won | Lost   Won | Lost
   1       Random     10  |  0    10  |  0     9  |  1     9  |  1
   2       MM_Open     8  |  2     8  |  2     7  |  3     7  |  3
   3      MM_Center    7  |  3     9  |  1     7  |  3     9  |  1
   4     MM_Improved   6  |  4     6  |  4     7  |  3     3  |  7
   5       AB_Open     4  |  6     8  |  2     6  |  4     5  |  5
   6      AB_Center    8  |  2     6  |  4     5  |  5     4  |  6
   7     AB_Improved   4  |  6     6  |  4     6  |  4     5  |  5
--------------------------------------------------------------------------
         Win Rate:      67.1%        75.7%        67.1%        60.0%
```

```
                    ************************
                         Playing Matches
                    ************************

Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                      Won | Lost   Won | Lost   Won | Lost   Won | Lost
   1       Random      9  |  1     9  |  1    10  |  0     8  |  2
   2       MM_Open     8  |  2     7  |  3     6  |  4     6  |  4
   3      MM_Center    9  |  1     7  |  3     8  |  2     8  |  2
   4     MM_Improved   5  |  5     7  |  3     6  |  4     8  |  2
   5       AB_Open     6  |  4     7  |  3     7  |  3     5  |  5
   6      AB_Center    5  |  5     7  |  3     2  |  8     4  |  6
   7     AB_Improved   5  |  5     5  |  5     6  |  4     5  |  5
--------------------------------------------------------------------------
         Win Rate:      67.1%        70.0%        64.3%        62.9%
```

After several tournaments we can estimate the average results of the best heuristics for a while:

	Win rate	Won	Draw	Lost
aggress_diff_weight_moves_one_ply_lookahead(game,player)	- 72.9%	3	0	0
aggressive_diff_moves_one_ply_lookahead (game, player)	- 65.2%	0	0	3

4. Results

Total results of implemented heuristics are shown in the summary table:

	Heuristic	T#1	T#2	T#3	T#4	T#5	T#6	T#7	Average
1	*open_move_score*	58.6							**58.6**
2	*improved_score*	62.9	55.7						**59.3**
3	*aggressive_heuristic*	60.0	61.4	62.9	67.1	62.9	58.6	62.5	**62.2**
4	*weighted_open_move_score*		60.0	57.1					**58.5**
5	*weighted_improved_score*			64.3	64.3				**64.3**
6	*diff_my_moves_opp_moves_one_ply_lookahead*				67.1	60.0			**63.5**
7	*aggressive_diff_moves_one_ply_lookahead*					67.6	71.3	65.2	**68.0**
8	*aggr_weighted_improved_score*						64.3		**64.3**
9	***aggress_diff_weight_moves_one_ply_lookahead***							72.9	**72.9**

As we can see, the **best heuristic is last one** - ***aggress_diff_weight_moves_one_ply_lookahead.*** It considers all the possible elements: number of available moves for the player and opponent, estimation of the position quality on the board, acceptable level of aggression in decision making and ply lookahead provides a more accurate evaluation.

So, ***aggress_diff_weight_moves_one_ply_lookahead*** does perform better in practice and hence can be recommended.