

3 agentplayer

Random agent

It is an agent that chooses randomly. As it does not use heuristic, it is not executed with any.

Minimax Agent (MM)

It is an agent that implements a fixed-depth minimax search algorithm. It is executed by all following heuristic functions.

Alpha-Beta agent (AB)

It is an agent that implements the fixed-depth Alpha-Beta pruning search algorithm. It is executed by all following heuristic functions.

Different heuristic functions in `sample_players.py`

- `open_move_score`
 - how many legal moves are there for player. With this score, player will only consider itself.
- `center_score`
 - the square of the distance between the center of the board and the location of the active player.
 - this will let the player move away from the center.
 - this doesn't consider the situation of the opponent.
- `improved_score`
 - the difference between own moves and opponent's moves. With this score, player will consider both. Since legal moves will be less and less with the game going, the difference will generally be less and less too. When the difference is positive, the active player gets a better situation than the other.

Different heuristic functions in `game_agent.py`

- `custom_score`
 - difference of the sum of available moves for each legal move of the two players.
 - this considers situation of both players and more precisely than the `improved_score`. In some meaning, this seems like to add the depth by 1.
 - this will block the opponent.
 - this will outplay the `improved_score`.
- `custom_score_2`
 - some kind of distance of two players' locations: the sum of absolute difference of each element of the location tuple.
 - this will let the active player away from the other.
 - the `improved_score` will outplay this score.
- `custom_score_3`
 - linear combination of `own_moves` (number of own legal moves) and `opp_moves` (number of opponent's legal moves): $\text{own_moves} - 1.5 * \text{opp_moves}$
 - this almost gets a tie with the `improved_score`.

Result picture

Here is the result of 400 matches. AB_Custom gets the best winning probability.

```
This script evaluates the performance of the custom_score evaluation function
against the `ID_Improved` agent baseline. `ID_CustomScore` is an agent using
Iterative Deepening and the custom_score function defined in game_agent.py.

*****
Playing Matches
*****

Match #   Opponent   AB_Improved   AB_Custom   AB_Custom_2   AB_Custom_3
          Won | Lost   Won | Lost   Won | Lost   Won | Lost
1         Random    190 | 10    186 | 14    183 | 17    187 | 13
2         MM_Open   96 | 104    147 | 53    105 | 95    119 | 81
3         MM_Center 148 | 52    168 | 32    139 | 61    147 | 53
4         MM_Improved 99 | 101    130 | 70    94 | 106    114 | 86
5         AB_Open   114 | 86    150 | 50    97 | 103    119 | 81
6         AB_Center 141 | 59    156 | 44    141 | 59    165 | 35
7         AB_Improved 101 | 99    134 | 66    85 | 115    102 | 98
-----
Win Rate:    63.5%      76.5%      60.3%      68.1%
```

Conclusion

My best score is the difference of the sum of available moves for each legal move of the two players which can outplay improved_score.