Heuristic analysis

All heuristics below grades move to a win position with a highest score and to loose position to a lowest score. It's a general note for all heuristics to don't put it in all descriptions.

Aggressive reducing number of an opponent open moves

Implemented in `scores.py#score_aggressive_diff`

Description

First I tried to use heuristic offered in lectures with an aggressive strategy implemented by weights in simple heuristic contains difference between number of open moves of our player and an opponent's open moves.

Result

Result that I get is, in fact, this heuristic underperformes our ID_Improved baseline:

```
*************************
 Evaluating: ID_Improved
*************************

Playing Matches:
----------
  Match 1: ID_Improved vs   Random      Result: 8 to 12
  Match 2: ID_Improved vs   MM_Null     Result: 12 to 8
  Match 3: ID_Improved vs   MM_Open     Result: 11 to 9
  Match 4: ID_Improved vs MM_Improved   Result: 10 to 10
  Match 5: ID_Improved vs   AB_Null     Result: 9 to 11
  Match 6: ID_Improved vs   AB_Open     Result: 10 to 10
  Match 7: ID_Improved vs AB_Improved   Result: 11 to 9


Results:
----------
ID_Improved         50.71%

*************************
   Evaluating: Student
*************************

Playing Matches:
----------
  Match 1:   Student   vs   Random      Result: 10 to 10
  Match 2:   Student   vs   MM_Null     Result: 8 to 12
  Match 3:   Student   vs   MM_Open     Result: 5 to 15
  Match 4:   Student   vs MM_Improved   Result: 3 to 17
  Match 5:   Student   vs   AB_Null     Result: 10 to 10
  Match 6:   Student   vs   AB_Open     Result: 12 to 8
  Match 7:   Student   vs AB_Improved   Result: 12 to 8
```

```
Results:
----------
Student             42.86%
```

Explanation

From analyzing this modified Isolation game rules it's clear that very aggressive strategy can be good only it will lead to winning immediately. Reason - in the canonical game from lectures by forcing aggressive strategy and decrease number of open moves helps us to partition the board and lock an opponents on his partition. In modified game it doesn't work this way and not so important. If you don't close your opponent completely by an aggressive strategy you will only leave him one move less, but in his next move he will change his surrounding cells and cells for next move significantly and your aggressive tactic would nearly not affect his next turns.

Safe increasing number of our open moves and consider open moves of an opponent less

Implemented in `scores.py#score_safe_diff`

Description

Based on previous heuristic I decided to try the "safe" version, where a number of our open moves has weight 2, while a number of opponent's open moves has weight 1. So, we mostly trying to keep our open moves better, but if, on the same time, we could reduce option for our opponent and have same amount of open moves - we would prefer to reduce options of an opponent too.

Result

New heuristic has significantly better results:

```
*************************
 Evaluating: ID_Improved
*************************

Playing Matches:
----------
  Match 1: ID_Improved vs   Random      Result: 10 to 10
  Match 2: ID_Improved vs   MM_Null     Result: 10 to 10
  Match 3: ID_Improved vs   MM_Open     Result: 9 to 11
  Match 4: ID_Improved vs MM_Improved   Result: 10 to 10
  Match 5: ID_Improved vs   AB_Null     Result: 13 to 7
  Match 6: ID_Improved vs   AB_Open     Result: 11 to 9
  Match 7: ID_Improved vs AB_Improved   Result: 15 to 5


Results:
----------
ID_Improved         55.71%

*************************
```

```
   Evaluating: Student
*************************

Playing Matches:
----------
  Match 1:   Student   vs   Random      Result: 10 to 10
  Match 2:   Student   vs   MM_Null     Result: 16 to 4
  Match 3:   Student   vs   MM_Open     Result: 10 to 10
  Match 4:   Student   vs MM_Improved   Result: 14 to 6
  Match 5:   Student   vs   AB_Null     Result: 16 to 4
  Match 6:   Student   vs   AB_Open     Result: 15 to 5
  Match 7:   Student   vs AB_Improved   Result: 16 to 4


Results:
----------
Student             69.29%
```

Explanation

```
2 * our_open - op_open
```

our_open	op_open	metric
3	2	4
3	1	5
4	2	6

So, as you can see, we will take a number of opponent open moves into account, but - we can't reduce it more that by one - reducing it by one doesn't overweight increasing number of our open moves by one.

Square distance to a board center

Implemented in `scores.py#score_sq_distance_to_center`

Description

This was developed as a helper to next combined heuristic. Idea is that keeping in the center is always better, so let's make a score (our square distance to a center / opponent square distance to a center). Square has been selected mostly to make it cheaper.

Result

```
*************************
 Evaluating: ID_Improved
*************************

Playing Matches:
----------
  Match 1: ID_Improved vs   Random      Result: 8 to 12
  Match 2: ID_Improved vs   MM_Null     Result: 10 to 10
  Match 3: ID_Improved vs   MM_Open     Result: 11 to 9
  Match 4: ID_Improved vs MM_Improved   Result: 10 to 10
  Match 5: ID_Improved vs   AB_Null     Result: 11 to 9
  Match 6: ID_Improved vs   AB_Open     Result: 14 to 6
  Match 7: ID_Improved vs AB_Improved   Result: 15 to 5


Results:
----------
ID_Improved         56.43%

*************************
   Evaluating: Student
*************************

Playing Matches:
----------
  Match 1:   Student   vs   Random      Result: 14 to 6
  Match 2:   Student   vs   MM_Null     Result: 15 to 5
  Match 3:   Student   vs   MM_Open     Result: 12 to 8
  Match 4:   Student   vs MM_Improved   Result: 12 to 8
  Match 5:   Student   vs   AB_Null     Result: 15 to 5
  Match 6:   Student   vs   AB_Open     Result: 14 to 6
  Match 7:   Student   vs AB_Improved   Result: 17 to 3


Results:
----------
Student             70.71%
```

Explanation

This heuristic developed to be combined with safe tactic, but it performs comparably with the safe tactic itself. Reason is that as described above, in modified Isolation rules number of open moves is not a very good metric, because we change our position too significantly each move and can move through occupied cells. So, heuristic to keep near the center works nearly same as heuristic looking on difference of open moves. Closer to center - more possible options, higher to stuck in a corner.

Safe heuristic combined with distance to a board center

Implemented in `scores.py#score_center_plus_safe_diff`

Description

Idea is just to combine this two metric by adding them and see what we get. Square distance took with sqrt here to make numbers in both metrics not normalized, but at least close and comparable to each other.

Result

```
*************************
```

```
 Evaluating: ID_Improved
*************************

Playing Matches:
----------
  Match 1: ID_Improved vs   Random      Result: 12 to 8
  Match 2: ID_Improved vs   MM_Null     Result: 8 to 12
  Match 3: ID_Improved vs   MM_Open     Result: 6 to 14
  Match 4: ID_Improved vs MM_Improved   Result: 10 to 10
  Match 5: ID_Improved vs   AB_Null     Result: 8 to 12
  Match 6: ID_Improved vs   AB_Open     Result: 13 to 7
  Match 7: ID_Improved vs AB_Improved   Result: 16 to 4


Results:
----------
ID_Improved         52.14%

*************************
   Evaluating: Student
*************************

Playing Matches:
----------
  Match 1:   Student   vs   Random      Result: 15 to 5
  Match 2:   Student   vs   MM_Null     Result: 14 to 6
  Match 3:   Student   vs   MM_Open     Result: 13 to 7
  Match 4:   Student   vs MM_Improved   Result: 14 to 6
  Match 5:   Student   vs   AB_Null     Result: 15 to 5
  Match 6:   Student   vs   AB_Open     Result: 13 to 7
  Match 7:   Student   vs AB_Improved   Result: 16 to 4


Results:
----------
Student             71.43%
```

Explanation

As usual with ensembles we get more stable result in terms of win/loose ratio and slight improved results.

Comparison

Heuristic Name	ID_Improved result	Student result	Student/Improved result
aggressive_diff	50.71%	42.86%	0.84
safe_diff	55.71%	69.29%	1.24
sq_distance_to_center	56.43%	70.71%	1.25
center_plus_safe_diff	52.14%	71.43%	1.37

I made two score functions which demonstrates about the same performance - Square distance to center and Safe weighed difference of open moves. Ensemble that just sums this two heuristics with a modification with square root to square distance to make it's values comparable with diff functions outperform both as expected, because forcing to maximize/minimize both of them. I maybe worth to play with normalization (replace square root with a fair normalization) and weights in this ensemble, but it would be a type of just hands tuning, which doesn't affect the general ideas.

Notes

Numbers in this report are a bit hard to compare, their variation is pretty high. Much more experiments required to get more statistically significant and comparable results, especially using laptop CPUs with variable clock speed.