Isolation Game(v.Knight) Heuristic Analysis

Tournament result

Each result is average of 6 tournaments each 10 matches against opponents Random, MM_Null, MM_Open, MM_Improved, AB_Null, AB_Open, AB_Improved . In total 280 games.

Heuristic	Win Probability	Avg. Depth	End Game Hit Rate
ID_Improved	0.61	5.690	0.228
ID_null	0.55	N/A	N/A
random_score	0.78	5.530	0.289
random_score(w/o util)	0.34	N/A	N/A
imp_score_w_neg_dist	0.71	N/A	N/A
imp_score_w_pos_dist	0.71	N/A	N/A
imp_score_w_neg_legal_moves_int	0.65	N/A	N/A
imp_score_w_pos_legal_moves_int	0.60	N/A	N/A
open_space_score	0.64	N/A	N/A
longest_path_score	N/A	N/A	N/A
variable_heuristic_score	0.77	5.467	0.285

- **Avg.Depth** is the sum of deepest depth reached during iterative deepening / total get_move count
- **End Game Hit Rate** is sum of get_move calls which return either a +inf or -inf / total get_move count

ID_Improved vs random_score

Results of total 100 games

Heuristic	Win Probability
ID_Improved	0.30
random_score	0.70

random_score vs variable_heuristic_score

Results of total 200 games

Heuristic	Win Probability
variable_heuristic_score	0.23
random_score	0.77

Heuristics

random_score

- **function:** random value between 0 and 1
- **purpose:** live the moment till you can see the end game
- **analysis:** after all time spend on to find a good heuristic this surprisingly simple heuristic has the best win rate. Either I have a bug I can not find or for isolation game with night move this is a really good heuristic.

random_score(w/o util):

- **function:** random value between 0 and 1, does not check for end game
- **purpose:** don’t even care about the end game just live the moment :)
- **analysis:** finding the end game utility is the secret essence to random score without it this heuristic has no value.

imp_score_w_neg_dist:

- **function:** improved_score - euclidean distance from the opponent
- **purpose:** in addition to improved score keep close to the opponent
- **analysis:** this heuristic scored above the improved one but still not good enough as random.

imp_score_w_pos_dist:

- **function:** improved_score + euclidean distance from the opponent
- **purpose:** in addition to improved score keep far away from to the opponent
- **analysis:** surprisingly using distance both negative and positive has the same effect on win rate.

imp_score_w_neg_legal_moves_int:

- **function:** improved_score - length of same legal moves with opponent
- **purpose:** in addition to improved score try to minimize the same legal moves with opponent
- **analysis:** I was trying to create partition by penalizing same legal moves of the players. It seems to work better than the sole improved_score but still far behind the random score

imp_score_w_pos_legal_moves_int:

- **function:** improved_score + length of same legal moves with opponent
- **purpose:** in addition to improved score try to maximize the same legal moves with opponent
- **analysis:** The negative of the above. This did not perform any better than the improved heuristic.

open_space_score:

- **function:** Calculate longest path of both players and return (my longest path - opponent longest path)
- **purpose:** Try to maximize the available space for player. This does not guarantee a win since the open space may not be traversable and opponent can cross the path anytime.
- **analysis:** It seemed so obvious in the beginning however there was always the probability of the opponent crossing the open space of the player and making most of the empty spaces unavailable. The computation time for this heuristic was around 0.3ms on my pc and I had an average best depth of 4.568 on iterative deepening. According to test results it made slightly better than the improved heuristic.

longest_path_score:

- **function:** Calculate longest path of both players and return (my longest path - opponent longest path)
- **purpose:** The player with the longest path wins, right?
- **analysis:** This seems to be the ultimate scoring function but it does not guarantee a win since the opponent can cross the path anytime. Finding the longest path is a NP-Hard problem and I wrote a very naive recursive
  implementation for the purposes of this assignment. I couldn't use this heuristic in the beginning of the game because of the time limit so I had to combine this with other heuristics and use

it close to end game.

variable_heuristic_score:

- **function:**
  if game is empty space percentage is above 0.3 run random_score
  if game is empty space percentage is above 0.2 below 0.3 run open_space_score
  if game is empty space percentage is above 0.0 below 0.2 run longest_path_score
- **purpose:** Computationally expensive heuristics like longest_path_score and open_space_score are taking too much time
  when the number of empty spaces is high and this reduces the depth that can be explored with iterative deepening.
  However close to end game when the number of empty spaces are low they both operated fast enough to not affect the average depth
- **analysis:** In theory it should outperform random but tests show that they score almost the same. I think the test setup is not good enough to compare these two high performing(random and this) heuristics.

Conclusion

I had lots of difficulties trying to come up with a conclusion. First my iterative deepening was flawed so that all fixed depth algorithms were

performing very poor. Since this was not covered in unit tests I did not realise this until I told about my score on slack. After fixing the bug I had to do all the tests again and came up with this table above. According to my tests the maximum winning heuristic is random moving with end game utility function which won 0.78 percent of the time in average against all the opponents. First I thought this was a bug and wrote different implementations of get_move, minimax and alphabeta however it did not change the score or I reproduced the bug somehow.

According to tests my best performing heuristic is random_score and variable_heuristic_score however variable_heuristic_score internally uses random score for most of the beginning game. After running one on one tests random score won against variable_heuristic_score on 0.77 percent of the time. This made it obvious my afvanted heuristic is a haux however after all the effort I choose to submit my project with variable_heuristic_score since it has similar high avg depth and end game hit rate. Finally I think the randomized first move nature of the tournament setup makes it very hard to compare well performing heuristics.