HEURISTIC ANALYSIS FOR AIND PROJECT 2 - ISOLATION GAME

Features used

own_moves: Total number of moves the current player has.
opp_moves: Total number of moves the opponent has.
location: Current location of the current player.
normalized_remaining_moves: Total number of blank squares left, divided by total number of squares of the board. This feature is chosen because it can tell whether the game is at early or late stage. The normalizing part ensures that the impact of this feature is scaled so that it does not overpower other features. Multiplying this means that the heuristic is more effective during early games, while dividing this means the heuristic is more effective during late game.

Heuristics examined

lecture_heuristic_improved: Similar to the heuristic given in lecture, but with adjusted weight to punish harder moves that allow more space for opponent. This implies a more offensive strategy than the original heuristic.

*Formular: lecture_heuristic_improved = own_moves - 3*opp_moves*

survival_heuristic: This heuristic does not care about the opponent move. What it does is to try to avoid states where number of legal moves for current player falls below a threshold. This implies a more defensive strategy. I chose this heuristic because being defensive makes a lot of sense in Isolation - after all the way to win the game is to survive longer than the opponent, so being patient is a feasible strategy especially against more aggresive opponents.

Formula: survival_heuristic = 3(own_moves -3)/normalized_remaining_moves*

positional_heuristic: This heuristic simply prefers moves that are closer to the center at the beginning of the game. Since this is most effective during early game, it is multiplied by normalized_remaining_moves to imply the proportional correlation. Due to poor performances and negligible effect on early tests, this heuristic was not used.

Formula: positional_heuristic = normalized_remaining_moves/(distance_to_center)

endgame_heuristic: This is the most expensive heuristic here. Only invoked when the game has past 30 moves, it expands one more level deep to obtain lecture_heuristic_improved score from all the successor states, accumulate them and normalize by normalized_remaining_moves. The incentive is that when the game comes to an end, seeing further becomes far more important, and also a bit more easier since the complexity have been reduced a lot.

Formula: endgame_heuristic = sum(lecture_heuristic_improved)/normalized_remaining_moves
where lecture_heuristic_improved is obtained from each successor states of the current state

composite_heuristic: A linear combination of all the above heuristics with appropriate weight. In this implementation, I decided to chose a rather risk-aversed strategy by placing less weight on lecture_heuristic_improved and more weight on survival_heuristic. Highest weight is reserved for endgame_heuristic since its the most important and most costly and can probably suggest killer moves thanks to its far sight.

*Formula: composite_heuristic = lecture_improved + 2*survival + 3*endgame*

Hardware

Simulation was done initially on a mid-2014 Macbook Pro 2.6 GHz Intel Core i5. As more tests were needed, simulation was run on AWS EC2 C4 instance (Intel Xeon Processor E5-2660 v3, 25M Cache, 2.60 GHz, 8CPUs) to avoid intermittent timeouts caused by inconsisttent CPU clocks and CPU resource being shared by other tasks.

Benchmark

NUM_MATCHES increased to 100 (initially 5) for every pairs of opponent. This means each pair will face each other 400 times instead of 20 initially. This big sample size helps eliminate any bias caused by random moves or inconsistent hardware performance. Below is the result.

lecture_heuristic_improved vs ID_improved: **81.56%** vs **82.96%**

endgame_heuristic vs ID_improved: **44.46%** vs **78.9%**

survival_heuristic vs ID_improved: **84%** vs **82.14%**

(lecture_heuristic_improved + survial_heuristic) vs ID_improved: **87.57%** vs **82.50%**

(lecture_heuristic_improved + endgame_heuristic) vs ID_improved: **86%** vs **84.43%**

composite_heuristic vs ID_improved: **88.11%** vs **83.61%**

Recommendation of evaluation function

composite_heuristic is recommended, for 3 reasons:

1/ From the benchmark results, it can be seen that composite_heuristic achieved the best performance among other stand-alone and even combined heuristics.

2/ By including endgame_heuristic, composite_heuristic ensures that student agent can always look one level deeper when the game is coming to an end. This strategy both gives student an advantage thanks to more information at this critical stage of the game and still makes sure that the complexity is still small enough since end_game_heuristic is only activated near the end of the game.

3/ endgame_heuristic benefits from different kinds of strategies, from more offensive (lecture_heuristic_improved) to more defensive (survival_heuristic) to deeper look up (endgame_heuristic). Even when combining other pairs of standalone heuristics, each combination still lack the other heuristics which can make it too offensive or too defensive or too short-sighted.

Also, for future improvement, a better positional heuristic may be included to find killer moves and allow well-tested strategies such as mimic etc. More experiments can also be run to better optimize the coefficient of composite_heuristic.