Designing and AI for isolation

Coding

The isolation AI was developed by first implementing minimax, then alpha beta pruning and finally iterative deepening as described in the lecture. Minimax and alpha beta were implemented such that the stop early once the game is won. The correct implementation of iterative deepening, minimax and alpha beta pruning AI was tested against the unit tests provided.

Developing a heuristics for isolation

Methodology

The heuristic scoring function of the AI was based on the combination of two concepts, mirroring and maintaining a competitive edge (for details please see below). The heuristics were first tested individually and then in combination, where the heuristics were combined using an non linear estimated for the stage of the game.

Improved base-line scoring function: Implemented as described in the lecture.

```
def baseline_score(game, player):

    if game.is_winner(player): return float("inf")
    if game.is_loser(player): return float("-inf")
    own_moves = len(game.get_legal_moves(player))
    opp_moves = len(game.get_legal_moves(game.get_opponent(player)))

    return float(own_moves - opp_moves)
```

Edge scoring function: This scoring function is a flavor of the improved scoring function, the difference is that using the ratio between own and opponent moves, the agent tries to maintain the pressure constant throughout the game.

```
def custom_score(game, player):
    if game.is_winner(player): return float("inf")
    if game.is_loser(player): return float("-inf")

    own_moves = len(game.get_legal_moves(player))
    opp_moves = len(game.get_legal_moves(game.get_opponent(player)))

    if opp_moves == 0: return float("inf")

    return own_moves/opp_moves
```

Mirroring scoring function: This scoring function reflects an annoying player, which just tries to stay as close as possible to the opponent, thus not giving it to much room.

```
def custom_score(game, player):
    if game.is_winner(player): return float("inf")
    if game.is_loser(player): return float("-inf")

    own_loc = game.get_player_location(player)
    opp_loc = game.get_player_location(game.get_opponent(player))

    # Trying to stay as close as possible to the opponent
    return float(abs(own_loc[0]-opp_loc[0])+abs(abs(own_loc[1]-opp_loc[1])))
```

Pressure scoring function: Combination of the edge and mirroring scoring function. The weights are combined using a non linear proxy for the progress of the game, thus gradually shifting the focus on the edge scorer.

```
def custom_score(game, player):
    if game.is_winner(player): return float("inf")
    if game.is_loser(player): return float("-inf")
    own_moves = len(game.get_legal_moves(player))
    opp_moves = len(game.get_legal_moves(game.get_opponent(player)))
    own_loc = game.get_player_location(player)
    opp_loc = game.get_player_location(game.get_opponent(player))

    if opp_moves == 0: return float("inf")

    # Approximate the stage of the game
    progress = game.move_count/(game.height*game.width)

    # Weighting mirroring (annoyance) vs dominance (edge)
    weight =  progress if progress > 0.3 else 0.3 # Thiny optimization to stay within the time limit

    return weight*own_moves/opp_moves + (1-weight)*(abs(own_loc[0]-opp_loc[0])+abs(abs(own_loc[1]-opp_loc
```

Results

Since, the initial setting of 5 tests to determine the performance of an individual agent were insufficient (> 5 % between runs), the **100 tests** were performed. To determine the reproducibility, the computational experiment was performed 3 times a batches of 100 tests for the baseline and the "always pressure" AI. In both cases the standard deviation was approximately 0.8 %, thus confirming the study's reproducibility.

Table 1: Outcome of the tournament between the different heuristics used together with the iterative deepening (ID) search. Opponents were a random player (Random), minimax (MM), alpha-beta pruning (AB) with no (null), number of valid moves left (open), and distance of valid moves to the opponent (improved) heuristics.

wins / losses	Random	MM_null	MM_open	MM_imporved	AB_null	AB_open	AB_imporved
ID improved (baseline)	167/33 171/29 165/35	155/45 152/48 165/3	125/75 125/75 136/64	117/83 132/68 130/7	143/57 147/53 144/56	136/64 137/63 117/83	117/83 121/79 119/81
ID edge	172/28	152/48	130/70	119/81	145/55	119/81	114/86
ID mirror	162/38	146/54	126/74	120/80	136/64	120/80	106/94
ID pressure	168/32 173/27 173/27	170/30 176/24 173/27	128/72 133/67 138/62	122/78 129/71 131/69	163/37 152/48 165/35	130/70 138/62 131/69	120/80 122/78 107/93

Table 2: The overall winning rates of the different heuristics are summarized in the following table.

Heuristic	Winning rate	Number of tests
ID improved (baseline)	69.4 %	300
ID edge	67.9 %	100
ID mirror	65.4 %	100
ID pressure	72.4 %	300

The pressure scoring function significantly out competed the improved baseline scoring function across 300 tests.

Discussion

The investigated heuristics is based on simple, every day game strategies. Keeping close to the opponent, and always try to keep ahead. These strategies apply from the very beginning a lot of pressure on the opponent. The approximation for the phase of the game, allows for an adaptive strategy, the pressure will be increased, once the space on the board is even more cramped. Since, these heuristics does not rely on in depth knowledge of the game mechanics, or opening books, it can be easily transfered to other games, and put to trial there. It for sure will be interesting to test what counter strategies are successful against such an rather unpleasant opponent.