Isolation Heuristic Functions

This report describes the implementation and assessment of three heuristic functions for the Isolation board game on a 7x7 grid.

1 Heuristics

Three heuristic functions were implemented with the sample Isolation game-playing code given for this assignment, two of them being `null_score` and `open_move_score`, neither of which will be discussed further. The third supplied heuristic, called `improved_score`, is the difference in the number of moves available to each player from their current position.

Over and above these heuristics, three more were implemented, and are described next.

1.1 Weighted Moves

The first heuristic was a modification of the `improved_score` heuristic. It's the difference between the number of moves available to the current player, minus three times the moves available to the opponent. This difference is multiplied by the number of filled squares on the board.

In other words, a player has to have markedly more moves available than the opponent to get a positive score, and small differences in the players' available moves take on greater significance as the game progresses and there are fewer open spots left on the board.

1.2 Reachable Squares

Instead of simply counting how many *moves* are open to each player, this heuristic counts how many *positions on the board* a player can potentially reach. This heuristic is intended to compensate for the horizon effect.

Consider the situation where a player is caught in a cul-de-sac, and only has one move available (which would yield a score of 1 under the `improved_score` heuristic.) Making that move takes the player to a larger, less-occupied area, where it has more room to manoeuvre. The additional moves available to the player at this new position - and all positions reachable from them - are reflected in this heuristic score.

This heuristic rewards players who can potentially cover, or reach, a larger part of the game board. It doesn't say anything about if, or how likely it is that, the player's opponent will cut off that player.

1.3 Cut-off Reachable

This heuristic is a similar to the previous `reachable_score` heuristic. However, instead of calculating the simple difference between the amount of squares a player can potentially reach, it is the difference between the number of squares that each player, *and each player alone*, can potentially reach (in other words, the disjunction of the sets of positions available to each player.)

The score is bumped up (by an amount equivalent to the size of the board) if there are no common squares both players can potentially reach - in other words, if one player has managed to cut off the other.

The idea behind this heuristic is to steer players to positions where they isolate their opponents to a part of the board where they have less space in which to move around than the attacking players.

The attacker should be able to comfortably win.

2 Results

Three separate game-playing agents, each using one of the custom heuristics, were matched against an agent implementing the `improved_score` heuristic. In each match (consisting of 5 games) both players alternated in playing first, from a random starting position, giving a total of 10 games per match. 10 matches were played in each run, and 3 runs were played to get the results shown here.

The following tables show the scores obtained by aggregating the first run (200 games), then the first two runs (400 games), and finally, all three runs (600 games.)

200 games	versus ID_Improved		
	Won	**Lost**	**Advantage**
Weighted Moves	99	101	49.50%
Reachable	102	98	51.00%
Cut-off Reachable	105	95	52.50%

400 games	versus ID_Improved		
	Won	**Lost**	**Advantage**
Weighted Moves	194	206	48.50%
Reachable	202	198	50.50%
Cut-off Reachable	190	210	47.50%

600 games	versus ID_Improved		
	Won	**Lost**	**Advantage**
Weighted Moves	301	299	50.17%
Reachable	291	309	48.50%
Cut-off Reachable	310	290	51.67%

Results in the table are highlighted when an agent using custom heuristic wins more games than the agent using the `improved_score` heuristic.

3 Observations & Conclusion

From the results given, it doesn't seem as if any of the custom heuristic functions have an advantage over the `improved_score` heuristic, or vice versa. The biggest advantage is 2.5%, and is attained both by the `cut_off_reachable` heuristic in the 200 game aggregate (representing a 5-game lead), and also by the opposing `improved_score` heuristic in the 400 games result set (a 10-game lead.)

An advantage is not absolute - that is, one heuristic does not always dominate the other. Up to the 400 match aggregate, the `reachable_score` heuristic always beats the `improved_score` heuristic, but not when played over 600 games. The `cut_off_reachable_score` wins over 200 games, loses over 400, and wins again over 600 games.

It almosts seems as if the win/loss advantages are due to plain chance. If we'd have to chose among the heuristics, however, the results seem to suggest going with the `cut_off_reachable_score` one. When it does win, it has some of the larger advantages (1.67% and 2.50%) compared to the other heuristics... but also one of the largest losses (also 2.50%.) However, it would appear to be the best performer out of our custom heuristics.

4 Discussion & Further Work

To be convincingly and consistently successful, it would seem an AI would have to have a better heuristic, a better strategy, and take into consideration how a game is played.

For instance, the tournament script allowed a player only a fixed amount of time to evaluate moves. That sort of environment favours heuristics which are quick and simple to evaluate (like `improved_score`) which only considers available moves, as opposed to the others, which not only have to calculate available moves, but also all positions reachable from them... and all available moves from them. What if the tournament allowed each player a fixed number of *moves* it could evaluate? That would allow for more expensive heuristics, and shift the focus onto the search strategy, like using alpha-beta pruning instead of minimax to take full advantage of the allocated quota of moves.

Another facet to strategy is knowing when to apply which heuristic. For example, all the heuristics discussed here aren't likely to provide much guidance to a player, or differentiate between the two players, in the opening stages of a game – as they're largely equal. When a game is starting and the board is still relatively clear, each player will have roughly the same amount of moves available at each turn, and be able to reach the same number of squares. A better idea would be to use a quick, cheap heuristic (like improved_score) or even better, a pre-computed opening book, in the opening stages of the game, and then a heuristic like cut_off_reachable at later stages, when there is a greater chance that one player can cut off another, and those opportunities are likely to be only a few plies ahead.