PHILLIP DUDERO
3/29/18

AIND Project: Build a Game-Playing Agent

Heuristic Analysis

The following heuristics were deployed in the Isolation game-playing agent realized for this project:

1) Custom_score: (own_moves – w*opponent_moves) where w = 10 / (move_count + 1). This weights positions that restrict the agent's opponent's moves much more when the move_count is low, at the beginning of the game; but after 9 moves in it begins to favor positions that maximize the number of moves available to the agent and ignores the opponent.
2) Custom_score_2: (own_moves – opponent_moves – w*nearWall), where "w" is the fraction of the board with blocked squares, and "nearWall" is Sum(nWall_own) – Sum(nWall_opp), where "Sum(nWall_...)" is the number of child nodes that are up against an edge of the board. Coefficient "w" is close to zero at the beginning of the game and close to one (1) at the end. This heuristic has the effect of weighting positions later in the game that are not on the edges for the agent, but are for the agent's opponent.
3) Custom_score_3: (own_moves – w*opponent_moves) where w=2.

Heuristics 1) and 2) were suggested by the Udacity reviewer.

One thing I noticed immediately is that the evaluation of these heuristics using "tournament.py" has one glaring problem: a lack of sufficient statistics needed to draw conclusions about the relative efficacy of these heuristics. With only 10 games played, the statistical uncertainty is roughly $1\sigma \approx \mathrm{sqrt}(10) = 3.2$ games, or 32%. We want to drive that down by increasing the number of games, under the assumption that the differences in wins between the heuristics might be within 30% of the total.

So I tested the three heuristics against all of the players provided in game_agents, namely "Random", "MM_Open", "MM_Center", "MM_Improved", "AB_Open", "AB_Center", and "AB_Improved", using a number of games per agent combination of 100 (Table 1). This gives a 10% uncertainty, which is still uncomfortably high. I also tested the three heuristcs against the "AB_Improved" agent (which implements "own_moves-opp_moves") only for 1000 matches (Table 2), which drops the relative statistical uncertainty down to 3.2%. The timeout thresholds were increased so that timeouts were kept to a negligible amount (running on a dual core Dell Windows 7 system with 4 GB RAM), but the total available time for searching was kept about the same.

Table 1

Match #	Opponent	AB_Improved Won	AB_Improved Lost	AB_Custom Won	AB_Custom Lost	AB_Custom_2 Won	AB_Custom_2 Lost	AB_Custom_3 Won	AB_Custom_3 Lost
1	Random	96	4	94	6	93	7	91	9
2	MM_Open	70	30	79	21	77	23	78	22
3	MM_Center	82	18	88	12	85	15	87	13
4	MM_Improved	72	28	74	26	78	22	70	30
5	AB_Open	56	44	51	49	52	48	56	44
6	AB_Center	49	51	58	42	58	42	59	41
7	AB_Improved	52	48	49	51	46	54	49	51
	Win Rate:	68.1%		70.4%		69.9%		70.0%	

Table 2

Match #	Opponent	AB_Improved	AB_Custom	AB_Custom_2	AB_Custom_3
		Won | Lost	Won | Lost	Won | Lost	Won | Lost
1	AB_Improved	513 | 487	509 | 491	496 | 504	501 | 499
	Win Rate:	51.3%	50.9%	49.6%	50.1%

At first, one may wonder why the performance is so much worse when running more games, but note that in Table 1 we are averaging over opponents that include 'Random' and "_Center', which perform poorly. When comparing "apples to apples", one sees that the performance is identical between the two tables against "AB_Improved." In fact, the two tables first demonstrate the huge difference in performance of the agents against "AB" opponents versus non-"AB" opponents. This is truly a statistically significant difference; it indicates that the gain in the depth of search from pruning is substantial.

However, when comparing the heuristics against the same opponent, one finds very little difference in performance; even with 1000 games, the win rates remain well within the statistical uncertainty. This may indicate the difficulty in defining a heuristic that successfully blocks an opponent who is capable of jumping over dead cells on the board.

As such, no recommendation on a particular heuristic can be given on the basis of performance alone. Given the choices, probably the simplest heuristic ought to be chosen, which for my money is the "AB_Improved" algorithm, which simply implements (own_moves-opp_moves). None of the other heuristics have shown that they can do any better for all their greater sophistication.