Advanced game playing: Heuristic Review

Octavio Navarro-Hinojosa

1 Evaluation functions

According to Russell and Norvig [1], an evaluation function returns an estimate of the expected utility of the game from a given position. The performance of a game-playing program depends on the quality of its evaluation function, which makes its crucial to define good evaluation functions:

- Evaluation functions should order the terminal states in the same way as the true utility function: states that are wins must evaluate better than draws, which in turn must evaluate better than losses.
- The evaluation function must not take too long.
- For the non-terminal states, the evaluation function should be strongly correlated with the actual chances of winning.

Most evaluation functions work by calculating various features of the state: pieces on the board, types of pieces, moves, spaces, etc. Most evaluation functions compute separate numerical contributions from each feature and then combine them to find the total value. This kind of evaluation function is called weighted linear function and can be expressed as:

$$E(s) = \sum_{i=1}^{n} w_i f_i(s) \tag{1}$$

where w_i is a weight and f_i is a feature of the position. this assumes that each feature is independent of the values of other features.

2 Selected Heuristics for Isolation

For the game of isolation, some of the features that can be used to create evaluation functions are: the the number of possible legal moves by either player, the number of filled spaces, the number of empty spaces, and the position of the players. These are all observable from the state of the board at any given time. Here, several heuristics that are linear combinations of the mentioned features are presented.

2.1 Heuristic 1: Using the number of moves for either player

As was seen in the course, one of the best heuristics was using the number of legal plays of the player minus the number of legal plays of the opponent. This can be expressed as:

$$E(s) = w_p * player\_moves + w_o * opponent\_moves \tag{2}$$

where w_p equals 1, and w_o equals -1. While these values may have yielded acceptable results, the weights can be improved in order to give more penalty to the opponent for having more moves (being less isolated), and more value to the player for having more moves. For this application, specific values were tested to try to find better results. For this heuristic, since the objective is for the opponent to fair worse than the player, only the w_o weight was varied; the w_p weight had a value of 1. Values for w_o ranged from -1 to -5, and it was found that -3 gave the best results. Further testing may be necessary to determine whether varying w_p has any impact on the score obtained.

2.2 Heuristic 2: Using the number of moves and the empty spaces

One key feature of isolation is the number of free spaces in the board. This metric gives an indication of how potentially free the players are, and also whether there is opportunity to move to an open area. By dividing the possible legal moves by the number of free spaces, we represent said opportunity. This heuristic values the players ability to move more than the opponent's, so a negative weight is used for the opponent. The heuristic can be expressed as:

$$E(s) = (player\_moves + w_o * opponent\_moves)/(num\_blank\_spaces + 1) \tag{3}$$

where w_o equals -3, and num_blank_spaces represent the available empty spaces in the board. To avoid divisions by zero, a 1 is added to the num_blank_spaces variable.

2.3 Heuristic 3: Using the number of moves and the filled spaces

Similarly to heuristic 2, another key feature is the number of filled spaces in the board. This metric helps to identify how isolated are the players. Ideally, the player will not be as isolated as the opponent, and it is intended to isolate the opponent as much as possible. To represent this, the heuristic is the number of possible legal moves for either player multiplied by the number of filled spaces. It should value the opponent worse, so a negative weight is also used. The heuristic can be expressed as:

$$E(s) = (player\_moves + w_o * opponent\_moves) * (num\_filled\_spaces + 1) \tag{4}$$

where w_o equals -3, and num_filled_spaces represent the filled spaces in the board. To avoid multiplications by zero, a 1 is added to the num_filled_spaces variable.

3 Results

To evaluate the heuristics, tournaments with the different heuristics were held with the parallel version of the *tournament.py* script. Each tournament consisted of 20 matches, with a time-out of 150 milliseconds. Since the alpha-beta pruning that was implemented was not ordered, and since the provided scripts returned moves at random, each time a tournament was run the resulting win-rate varied considerably; up to a 6% difference in the conducted tests. To get a more acceptable result, 5 tournaments were performed, and the win-rate of each heuristic was averaged. Table 1 shows the results of the tournaments.

Table 1: Average Wins and loses for the tournaments. Results are in the form of "Wins — Losses".

Opponents	AB_improved	Heuristic 1	Heuristic 2	Heuristic 3
1. Random	18.6 — 1.4	19.8 — 0.2	19 — 1	19.25 — 0.75
2. MM_Open	15.4 — 4.6	16.8 — 3.2	15.4 — 4.6	16.75 — 3.25
3. MM_Center	19 — 1	19.2 — 0.8	18.8 — 1.2	19.25 — 0.75
4. MM_Improved	15 — 5	14.2 — 5.8	13.2 — 6.8	16.25 — 3.75
5. AB_Open	11.6 — 8.4	11.4 — 8.6	13 — 7	11.25 — 8.75
6. AB_Center	16.8 — 3.2	17.2 — 2.8	17.6 — 2.4	17.5 — 2.5
7. AB_Improved	10.6 — 9.4	10 — 10	9.2 — 10.8	11.5 — 8.5
Win-Rate	76.43%	77.57%	75.86%	79.82%

The AB_improved column shows the average win-loss values for the baseline heuristic provided in the code ($player\_moves - opponent\_moves$), while the Heuristic columns show the win-loss values for each proposed heuristic.

From the win-rates alone, it is clear that the heuristics performance was really close, differing only by around 4%. From the results of the AB_improved and Heuristic 1 columns, it can be seen that modifying the values of the weights for the evaluation function has an impact on the calculated value (even though it was only 1.1% in this case), and it would be worth it to look for ways to select better weights.

Heuristic 2 and 3 included features of the board spaces themselves, not only of the available moves of the players. The results make it clear that adding features that add value are dependent on their meaning and relevance. In the case of Heuristic 2, since the number of empty spaces was considered for both players, it gave them both a better chance at selecting a better position, which led to more wins for the opponent. This was the worse performing heuristic overall because the goal is to isolate the opponent, not give them more spaces to move. Heuristic 3, in contrast, was the better performing heuristic overall since it looked for moves that gave the opponent less spaces to move to.

4 Conclusions

In the present report, three heuristics were presented and evaluated. From those heuristics, the one that exhibited the best performance was Heuristic 3, which used the number of moves and the filled spaces to evaluate a board position. Other features that could be studied are the impact of the moves, or the distance to a position that has a better chance to give better moves.

It was clear that adding certain features could improve the performance of the heuristics as long as their meaning had a significant relevance to the final objective of the game. Finally, the weights for the functions were selected rather arbitrarily, so better ways to determine them could greatly improve the performance of the heuristics.

References

[1] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Third Edition, 2010.