**Building a Game-Playing Agent**

**Heuristic Analysis**

We present some of the heuristic evaluation functions that are used in conjunction with *Minimax* and *Alpha-Beta Pruning* algorithms to create a game-playing agent for a modified version of the game - *Isolation*. These functions are tested against the ‘*ID\_Improved’ baseline* heuristic*.* Additionally, one among them is identified and used in designing a superior game-playing agent.

**1. Introduction**

Isolation is a simple board game, the object of which is to "isolate" the opponent such they are left with no legal moves or alternately, be the last player with a legal move on the game-board. The modified version of the game, where the pieces move like knights on a chess board, makes for an array of interesting approaches.

**2. A Good Heuristic**

Russell and Norvig [1] set the following conditions for a good evaluation function.

1. Terminal states should be ordered just like they are in a utility function
2. The computation should be fast
3. For non-terminal states, the function should strongly correlate with the actual chances of winning

**3. The Strategy**

To achieve a sustained winning percentage in the region of 63-80% we devised the following two pronged strategy

1. Avoid the opponent’s path during the initial phase and reach a stage of the game where all scenarios are computable quickly

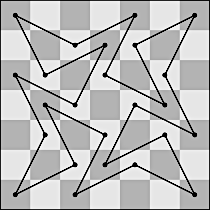
2. Look at all possible scenarios to compute the best next move. This should be deep enough to ensure quick computation however not so deep that the iterative search reaches the terminal stages and negate the advantage over the other agents.

**4. The Candidates**

The Student Agent powered by the heuristic that we create, is pitted against four different evaluation functions – Random, Null, Open Move Score, Improved Score.

**3.1. Diverge and Converge**

To begin with, we refined the Improved Score functions such that we pick those squares which the opponent does not move to. We named this function *Diverge* and as can be surmised, this ensured longer gameplay (more moves). The results were on par with the heuristics provided (~52-54%). To make things more interesting and arrive at a result before terminal nodes are reached in the iterative search, we introduced a *Converge* heuristic which kicked into action when the game was nearing it’s end (a function of the number of blank spaces remaining). While we found an uptick in winning percentage, the heuristic still struggled to (48-58%) to achieve a sustained advantage over the existing heuristics.



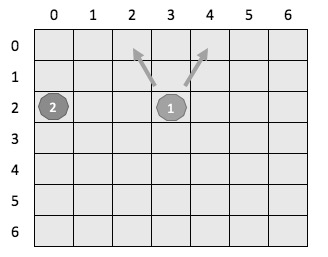
**3.2. Knight’s Tour (Warnsdorf's Rule)**

A Knight’s Tour is the path taken by the knight across an *n* x *n* chess board such that, it visits each square exactly once. When *n* is odd, as is the case in our tests, the knight traverses a closed path tour across the board as shown in Fig 1.

This ties in well with our strategy laid out earlier, since Warnsdorf’s rule ensures that the squares with the least possible moves are traversed first. This makes for extended game play with an average round containing l8-19 moves. The winning percentage hovered around 56 to 63% and was thus superior to the earlier heuristic. However, there were still improvements to be made to reach our targeted winning percentage.

Figure 1: Knight's Tour on 7x7 chess board

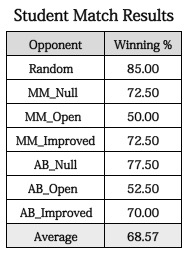
**3.3. Knight’s Tour Improved**



The Knight’s tour serves us well for the initial phase of the game, however we need a quick kill strategy for lesser agents and a more opportunistic one when dealing with the stronger agents.

Warnsdorf’s rule of choosing any square with the lowest moves works very well for boards with *n* < 50*.* However, for larger boards, we need devise a way to break a tie, without which the knight will fail to visit all squares. Similarly in our case, since we are dealing with an adversary we use a distance factor to break the tie. For example in figure 2, player 1 can move to either *(0, 2)* or *(0, 4).* We use distance formula to compute the distance between these points and the opponent’s position and go ahead with the point that is further away, that is *(0, 4)* in this case.

Figure 2: Decision making using distance factor.

Based on the match data, we observe that switching from Knight’s Tour to a max legal moves strategy after 8 moves gives the best results. Additionally, we are now able go traverse the entire game tree and factor in the branching and depth of each available scenario. These factors ensure that we are best placed to realize are initial goal.

**4. Results**

Based on our study of board configuration and game data we tweaked a well researched algorithm to create a game-playing agent which can outperform *ID\_Improved.* As can be seen from figure 3 and 4 our Student Agent has a winning percentage of 64% to 80%. This amounts to a performance of 20-41% better than *ID\_Improved*. Also the game data shows that the average moves per round is 16.32 and games involving Random and Improved Agents average less than 15 moves. However, there is further scope for improvement in terms of understanding the board configuration and using that to toggle between attack and defense.

Figure 3: Match results after 40 rounds with each agent. 68.57% vs 55.36% for ID\_Improved

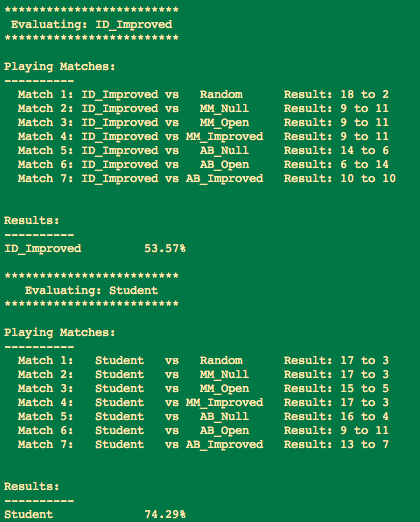


Figure 4: Match results after 20 rounds with each agent. Student 74.29% vs 53.57% for ID\_Improved

References

1. Stuart Russell, Peter Norvig, Artificial Intelligence – A Modern Approach (Pearson)

2. Judea Pearl, Heuristics Intelligent Search Strategies for Computer Problem Solving (Addison-Wesley Publishing Company)

3. Ronald L. Rivest, Game Tree Searching by Min/Max Approximation\*