**Analysis of the heuristics used in the isolation problem:**

**First heuristic:**
**(own_moves - 3*opp_moves) * filled**
This heuristic is based on the heuristic presented in the lecture where the number of potential moves from the opponent was subtracted from the number of moves available to the player. The heuristic was improved by multiplying the number of moves of the opponent by a factor of 3 and multiplying the whole thing by the number of filled tiles on the board. This heuristic performed slightly better than the benchmark heuristic.

```
*************************
 Evaluating: ID_Improved
*************************

Playing Matches:
----------
  Match 1: ID_Improved vs   Random      Result: 17 to 3
  Match 2: ID_Improved vs   MM_Null     Result: 17 to 3
  Match 3: ID_Improved vs   MM_Open     Result: 15 to 5
  Match 4: ID_Improved vs MM_Improved   Result: 14 to 6
  Match 5: ID_Improved vs   AB_Null     Result: 15 to 5
  Match 6: ID_Improved vs   AB_Open     Result: 14 to 6
  Match 7: ID_Improved vs AB_Improved   Result: 13 to 7


Results:
----------
ID_Improved         75.00%

*************************
   Evaluating: Student
*************************

Playing Matches:
----------
  Match 1:   Student   vs   Random      Result: 18 to 2
  Match 2:   Student   vs   MM_Null     Result: 17 to 3
  Match 3:   Student   vs   MM_Open     Result: 16 to 4
  Match 4:   Student   vs MM_Improved   Result: 17 to 3
  Match 5:   Student   vs   AB_Null     Result: 14 to 6
  Match 6:   Student   vs   AB_Open     Result: 13 to 7
  Match 7:   Student   vs AB_Improved   Result: 16 to 4


Results:
----------
Student             79.29%
```

**Second heuristic:**
**(len(num_spaces)/5.0 * len(game.get_legal_moves(player)))**
This heuristic is based on the number of free tiles at a distance of 2 around the player. The idea behind this heuristic is that if the player is surrounded by occupied tile, than he is probably in an unfavourable position.
This heuristic performed rather poorly while matched against the reference heuristic ID Improved.

```
*************************
 Evaluating: ID_Improved
*************************

Playing Matches:
----------
```

```
  Match 1: ID_Improved vs   Random      Result: 19 to 1
  Match 2: ID_Improved vs   MM_Null     Result: 16 to 4
  Match 3: ID_Improved vs   MM_Open     Result: 18 to 2
  Match 4: ID_Improved vs MM_Improved   Result: 12 to 8
  Match 5: ID_Improved vs   AB_Null     Result: 18 to 2
  Match 6: ID_Improved vs   AB_Open     Result: 16 to 4
  Match 7: ID_Improved vs AB_Improved   Result: 14 to 6


Results:
----------
ID_Improved         80.71%

*************************
   Evaluating: Student
*************************

Playing Matches:
----------
  Match 1:   Student   vs   Random      Result: 15 to 5
  Match 2:   Student   vs   MM_Null     Result: 14 to 6
  Match 3:   Student   vs   MM_Open     Result: 6 to 14
  Match 4:   Student   vs MM_Improved   Result: 9 to 11
  Match 5:   Student   vs   AB_Null     Result: 8 to 12
  Match 6:   Student   vs   AB_Open     Result: 7 to 13
  Match 7:   Student   vs AB_Improved   Result: 8 to 12


Results:
----------
Student             47.86%
```

**Third Heuristic:**

**own_moves**

This heuristic simply returns the number of moves available to the current player. While this heuristic might seem to be too simple, it seems like it could be a good fit when the branching factor is relatively high because it is rather quick to compute, allowing the game tree to be search more deeply. In this case it performed slightly better than the ID Improved heuristic.

```
*************************
 Evaluating: ID_Improved
*************************

Playing Matches:
----------
  Match 1: ID_Improved vs   Random      Result: 16 to 4
  Match 2: ID_Improved vs   MM_Null     Result: 16 to 4
  Match 3: ID_Improved vs   MM_Open     Result: 12 to 8
  Match 4: ID_Improved vs MM_Improved   Result: 13 to 7
  Match 5: ID_Improved vs   AB_Null     Result: 16 to 4
  Match 6: ID_Improved vs   AB_Open     Result: 15 to 5
  Match 7: ID_Improved vs AB_Improved   Result: 14 to 6


Results:
----------
ID_Improved         72.86%

*************************
   Evaluating: Student
*************************

Playing Matches:
----------
  Match 1:   Student   vs   Random      Result: 18 to 2
  Match 2:   Student   vs   MM_Null     Result: 14 to 6
  Match 3:   Student   vs   MM_Open     Result: 16 to 4
  Match 4:   Student   vs MM_Improved   Result: 17 to 3
```

```
  Match 5:   Student   vs   AB_Null     Result: 16 to 4
  Match 6:   Student   vs   AB_Open     Result: 12 to 8
  Match 7:   Student   vs AB_Improved   Result: 11 to 9


Results:
----------
Student             74.29%
```

**Conclusion:**

Surprisingly, the simplest heuristic(*#3*) was one of the best ones evaluated. This may be due to the nature of the isolation rules used. Since the players move like a knight in chess, it is not possible to have a player locked in one part of the game board, because the player can jump over filled spaces. This is in contrast with regular isolation, where game board partitioning is usually a major factor in a victory. The more sophisticated heuristic(*#2*) that uses the number of available spaces around the player, combined with the number of legal moves performed very poorly against the benchmark heuristic. This probably because the heuristic was to expensive to calculate and my not have factor relevant information. Finally, from the 3 proposed heuristics, the heuristic that uses a subtraction of the player moves and opponent moves combined with the number of filled spaces(*#1*) was the best heuristic and the one used for the final implementation. It was chosen mainly because it is relatively simple to compute, uses 3 different sources of information and performed better than all other heuristics evaluated for the game.