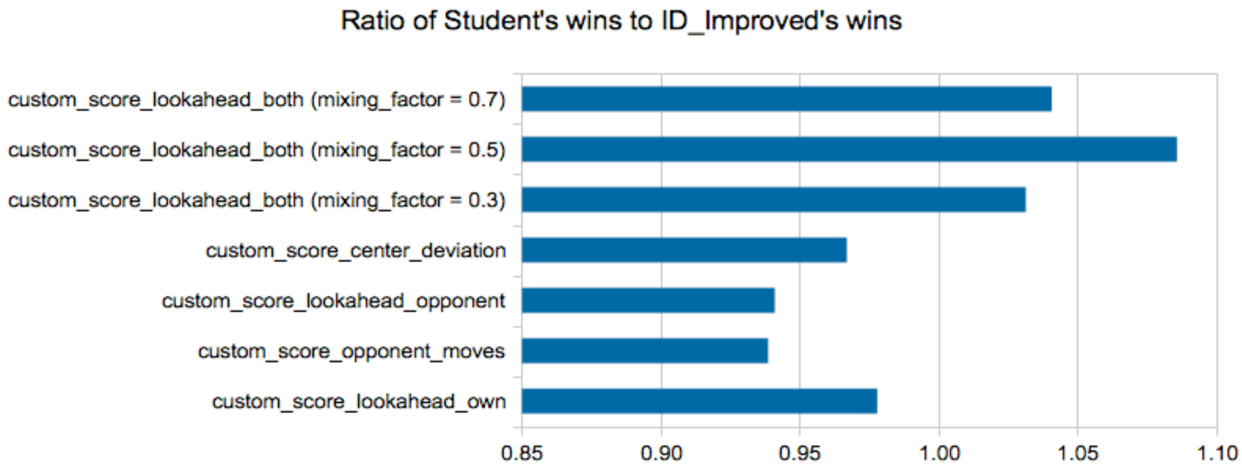
Heuristic Analysis

For this project the following heuristic score function algorithms are proposed. For each proposed algorithm, the results of the tournament are provided and compared with the improved iterative deepening:

```
def custom_score_improved(game, player):
    This score function returns the difference
    between the number of moves available for self and the opponent player.
    A mixing factor is used to compute weighted sum of the two instead of
    plain addition.

def custom_score_lookahead_own(game, player):
    This score function looks ahead one level deeper and
    returns the number of legal moves at the current step and the average of
    number of moves available in the next level due to each of the moves in
    the current level.


def custom_score_opponent_moves(game, player):
    This score function only returns the number of moves
    available for the opponent in the current state of the game.


def custom_score_lookahead_opponent(game, player):
    This score function looks at the number of available
    moves for its own player subtracted by the average number of moves available
    for the opponent in the next round.

def custom_score_center_deviation(game, player):
    This score function discourages the player to go to
    the boundaries of the board by discounting the distance to centre of the
    board from the returned score.

def custom_score_lookahead_both(game, player):
    This score function looks ahead one level deeper and
    returns the number of legal moves at the current step and the average of
    number of moves available due to each of the moves in previous step.
    This look ahead is performed for both players, and then return a weighted
    sum of the current and future number of available moves.
```

The following graph shows the ratio of the number of wins by the `Student` agent for each proposed score function to the number of wins by the `ID_Improved` agent in the same tournament. As can be seen, the score functions that looks ahead into the next level for both self and the opponent (`custom_score_lookahead_both`) tends to outperform the `ID_Improved` by about 9%. In this score function, there is a mixing_factor to distribute the emphasis on current or future states. The tournament is run for various mixing_factor's, and it seems an equal weight for the current and immediate future states is the best strategy. The degradation of the win ratio can be observed when mixing_factor is deviated from 0.5.



Recommendation for evaluation function:

In terms of win ratio, the best score function seems to be the one that looks at the current state of the game as well as the future state of the game for both players, i.e., custom_score_lookahead_both. This score function is the recommendation this report for the following three reasons:

1. Performance:
   The data shows that this score function can outperform the provided baseline by 9%.
2. Complexity:
   This score function can compute the score in linear time. That is, the complexity is $\theta(1)$.
3. Code Readability:
   The code for this score function is still small and readable, thus enabling easy editing and bug finding.

APPENDIX

In what follows the source code and tournament results for the above mentioned score functions are presented.

```python
def custom_score_improved(game, player):
    """Calculate the heuristic value of a game state from the point of view
    of the given player. This score function returns the difference
    between the number of moves available for self and the opponent player.
    A mixing factor is used to compute weighted sum of the two instead of
    plain addition.

    Parameters
    ----------
    game : `isolation.Board`
        An instance of `isolation.Board` encoding the current state of the
        game (e.g., player locations and blocked cells).

    player : object
        A player instance in the current game (i.e., an object corresponding to
        one of the player objects `game.__player_1__` or `game.__player_2__`.)

    Returns
    ----------
    float
        The heuristic value of the current game state to the specified player.
    """

    if game.is_loser(player):
        return float("-inf")

    if game.is_winner(player):
        return float("inf")
    mixing_factor = 0.7
    own_moves = len(game.get_legal_moves(player))
    opp_moves = len(game.get_legal_moves(game.get_opponent(player)))
    return float(mixing_factor * own_moves + (1 - mixing_factor) * (-opp_moves))
```

```python
def custom_score_lookahead_own(game, player):
    """Calculate the heuristic value of a game state from the point of view
    of the given player. This score function looks ahead one level deeper and
    returns the number of legal moves at the current step and the average of
    number of moves available due to each of the moves in previous step.

    Parameters
    ----------
    game : `isolation.Board`
        An instance of `isolation.Board` encoding the current state of the
        game (e.g., player locations and blocked cells).

    player : object
        A player instance in the current game (i.e., an object corresponding to
        one of the player objects `game.__player_1__` or `game.__player_2__`.)

    Returns
    ----------
    float
        The heuristic value of the current game state to the specified player.
    """

    # TODO: finish this function!
    #raise NotImplementedError
    if game.is_loser(player):
        return float("-inf")

    if game.is_winner(player):
        return float("inf")

    own_moves = game.get_legal_moves(player)
    if len(own_moves) == 0:
        return float("-inf")
    lookahead_moves = 0.0
    for move in own_moves:
        lookahead_game = game.forecast_move(move)
        lookahead_moves += len(lookahead_game.get_legal_moves(player))
    lookahead_moves /= len(own_moves)
    score = len(own_moves) + lookahead_moves
    return float(score)
```

This script evaluates the performance of the custom heuristic function by comparing the strength of an agent using iterative deepening (ID) search with alpha-beta pruning against the strength rating of agents using other heuristic functions. The `ID_Improved` agent provides a baseline by measuring the performance of a basic agent using Iterative Deepening and the "improved" heuristic (from lecture) on your hardware. The `Student` agent then measures the performance of Iterative Deepening and the custom heuristic against the same opponents.

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

Playing Matches:
----------
  Match 1: ID_Improved vs   Random       Result: 18 to 2
  Match 2: ID_Improved vs   MM_Null      Result: 15 to 5
  Match 3: ID_Improved vs   MM_Open      Result: 11 to 9
  Match 4: ID_Improved vs MM_Improved    Result: 10 to 10
  Match 5: ID_Improved vs   AB_Null      Result: 15 to 5
  Match 6: ID_Improved vs   AB_Open      Result: 11 to 9
  Match 7: ID_Improved vs AB_Improved    Result: 11 to 9


Results:
----------
ID_Improved         65.00%

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

Playing Matches:
----------
  Match 1:   Student   vs   Random       Result: 16 to 4
  Match 2:   Student   vs   MM_Null      Result: 13 to 7
  Match 3:   Student   vs   MM_Open      Result: 10 to 10
  Match 4:   Student   vs MM_Improved    Result: 14 to 6
  Match 5:   Student   vs   AB_Null      Result: 12 to 8
  Match 6:   Student   vs   AB_Open      Result: 12 to 8
  Match 7:   Student   vs AB_Improved    Result: 12 to 8


Results:
----------
Student             63.57%
```

```python
def custom_score_opponent_moves(game, player):
    """Calculate the heuristic value of a game state from the point of view
    of the given player. This score function only returns the number of
    moves available for opponent player.

    Parameters
    ----------
    game : `isolation.Board`
        An instance of `isolation.Board` encoding the current state of the
        game (e.g., player locations and blocked cells).

    player : object
        A player instance in the current game (i.e., an object corresponding to
        one of the player objects `game.__player_1__` or `game.__player_2__`.)

    Returns
    ----------
    float
        The heuristic value of the current game state to the specified player.
    """

    # TODO: finish this function!
    #raise NotImplementedError
    if game.is_loser(player):
        return float("-inf")

    if game.is_winner(player):
        return float("inf")

    own_moves = 0 # len(game.get_legal_moves(player))
    opp_moves = len(game.get_legal_moves(game.get_opponent(player)))
    return float(own_moves - opp_moves)
```

Tournament results for **custom_score_opponent_moves()** :

This script evaluates the performance of the custom heuristic function by comparing the strength of an agent using iterative deepening (ID) search with alpha-beta pruning against the strength rating of agents using other heuristic functions. The `ID_Improved` agent provides a baseline by measuring the performance of a basic agent using Iterative Deepening and the "improved" heuristic (from lecture) on your hardware. The `Student` agent then measures the performance of Iterative Deepening and the custom heuristic against the same opponents.

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

Playing Matches:
----------
  Match 1: ID_Improved vs   Random       Result: 19 to 1
  Match 2: ID_Improved vs   MM_Null      Result: 12 to 8
  Match 3: ID_Improved vs   MM_Open      Result: 14 to 6
  Match 4: ID_Improved vs MM_Improved    Result: 12 to 8
  Match 5: ID_Improved vs   AB_Null      Result: 17 to 3
  Match 6: ID_Improved vs   AB_Open      Result: 13 to 7
  Match 7: ID_Improved vs AB_Improved    Result: 11 to 9


Results:
----------
ID_Improved         70.00%

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

Playing Matches:
----------
  Match 1:   Student   vs   Random       Result: 17 to 3
  Match 2:   Student   vs   MM_Null      Result: 15 to 5
  Match 3:   Student   vs   MM_Open      Result: 11 to 9
  Match 4:   Student   vs MM_Improved    Result: 12 to 8
  Match 5:   Student   vs   AB_Null      Result: 15 to 5
  Match 6:   Student   vs   AB_Open      Result: 10 to 10
  Match 7:   Student   vs AB_Improved    Result: 12 to 8


Results:
----------
Student             65.71%
```

```python
def custom_score_lookahead_opponent(game, player):
    """Calculate the heuristic value of a game state from the point of view
    of the given player. This score function looks at the number of available
    moves for its own player subtracted by the average number of moves available
    for the opponent in the next round.

    Parameters
    ----------
    game : `isolation.Board`
        An instance of `isolation.Board` encoding the current state of the
        game (e.g., player locations and blocked cells).

    player : object
        A player instance in the current game (i.e., an object corresponding to
        one of the player objects `game.__player_1__` or `game.__player_2__`.)

    Returns
    ----------
    float
        The heuristic value of the current game state to the specified player.
    """

    # TODO: finish this function!
    #raise NotImplementedError
    if game.is_loser(player):
        return float("-inf")

    if game.is_winner(player):
        return float("inf")

    own_moves = game.get_legal_moves(player)
    if len(own_moves) == 0:
        return float("-inf")

    opp_moves = 0.0
    for move in own_moves:
        new_game = game.forecast_move(move)
        opp_moves += len(new_game.get_legal_moves(game.get_opponent(player)))
    # get the average moves that the opponent have
    avg_opp_moves = opp_moves / len(own_moves)
    return float(len(own_moves) - avg_opp_moves)
```

Tournament results for custom_score_lookahead_opponent():

This script evaluates the performance of the custom heuristic function by comparing the strength of an agent using iterative deepening (ID) search with alpha-beta pruning against the strength rating of agents using other heuristic functions. The `ID_Improved` agent provides a baseline by measuring the performance of a basic agent using Iterative Deepening and the "improved" heuristic (from lecture) on your hardware. The `Student` agent then measures the performance of Iterative Deepening and the custom heuristic against the same opponents.

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

Playing Matches:
----------
  Match 1: ID_Improved vs   Random        Result: 19 to 1
  Match 2: ID_Improved vs   MM_Null       Result: 17 to 3
  Match 3: ID_Improved vs   MM_Open       Result: 12 to 8
  Match 4: ID_Improved vs MM_Improved     Result: 11 to 9
  Match 5: ID_Improved vs   AB_Null       Result: 14 to 6
  Match 6: ID_Improved vs   AB_Open       Result: 13 to 7
  Match 7: ID_Improved vs AB_Improved     Result: 16 to 4


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

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

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


Results:
----------
Student             68.57%
```

```python
def custom_score_center_deviation(game, player):
    """Calculate the heuristic value of a game state from the point of view
    of the given player. This score function discourages the player to go to
    the boundaries of the board by discounting the distance to centre of the
    board from the returned score.

    Parameters
    ----------
    game : `isolation.Board`
        An instance of `isolation.Board` encoding the current state of the
        game (e.g., player locations and blocked cells).

    player : object
        A player instance in the current game (i.e., an object corresponding to
        one of the player objects `game.__player_1__` or `game.__player_2__`.)

    Returns
    ----------
    float
        The heuristic value of the current game state to the specified player.
    """

    if game.is_loser(player):
        return float("-inf")

    if game.is_winner(player):
        return float("inf")

    mixing_factor = 0.8
    board_center = (game.width / 2.0, game.height / 2.0)
    current_location = game.get_player_location(player)
    distance_to_center = (current_location[0] - board_center[0]) ** 2 +
(current_location[1] - board_center[1]) ** 2
    distance_to_center_normilized = distance_to_center / ( (board_center[0]) ** 2 +
(board_center[1]) ** 2 )
    #print("distance to center = " + str(-distance_to_center))

    nrof_own_moves = len(game.get_legal_moves(player))
    nrof_own_moves_normilized = nrof_own_moves / 8.0

    score = mixing_factor * nrof_own_moves_normilized + (1 - mixing_factor) * (-
distance_to_center_normilized)
    #print('score = ' + str(score))

    return float(score)
```

Tournament results for custom_score_center_deviation():

This script evaluates the performance of the custom heuristic function by comparing the strength of an agent using iterative deepening (ID) search with alpha-beta pruning against the strength rating of agents using other heuristic functions. The `ID_Improved` agent provides a baseline by measuring the performance of a basic agent using Iterative Deepening and the "improved" heuristic (from lecture) on your hardware. The `Student` agent then measures the performance of Iterative Deepening and the custom heuristic against the same opponents.

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

Playing Matches:
----------
  Match 1: ID_Improved vs   Random      Result: 16 to 4
  Match 2: ID_Improved vs   MM_Null     Result: 19 to 1
  Match 3: ID_Improved vs   MM_Open     Result: 11 to 9
  Match 4: ID_Improved vs MM_Improved   Result: 10 to 10
  Match 5: ID_Improved vs   AB_Null     Result: 10 to 10
  Match 6: ID_Improved vs   AB_Open     Result: 12 to 8
  Match 7: ID_Improved vs AB_Improved   Result: 13 to 7


Results:
----------
ID_Improved         65.00%

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

Playing Matches:
----------
  Match 1:   Student   vs   Random      Result: 16 to 4
  Match 2:   Student   vs   MM_Null     Result: 17 to 3
  Match 3:   Student   vs   MM_Open     Result: 12 to 8
  Match 4:   Student   vs MM_Improved   Result: 9 to 11
  Match 5:   Student   vs   AB_Null     Result: 13 to 7
  Match 6:   Student   vs   AB_Open     Result: 10 to 10
  Match 7:   Student   vs AB_Improved   Result: 11 to 9


Results:
----------
Student             62.86%
```

```python
def custom_score_lookahead_both(game, player):
    """Calculate the heuristic value of a game state from the point of view
    of the given player. This score function looks ahead one level deeper and
    returns the number of legal moves at the current step and the average of
    number of moves available due to each of the moves in previous step.
    This look ahead is performed for both players.

    Parameters
    ----------
    game : `isolation.Board`
        An instance of `isolation.Board` encoding the current state of the
        game (e.g., player locations and blocked cells).

    player : object
        A player instance in the current game (i.e., an object corresponding to
        one of the player objects `game.__player_1__` or `game.__player_2__`.)

    Returns
    ----------
    float
        The heuristic value of the current game state to the specified player.
    """

    # TODO: finish this function!
    #raise NotImplementedError
    if game.is_loser(player):
        return float("-inf")

    if game.is_winner(player):
        return float("inf")

    own_moves = game.get_legal_moves(player)
    opp_moves = len(game.get_legal_moves(game.get_opponent(player)))

    mixing_factor = 0.3
    if len(own_moves) == 0:
        return float("-inf")
    lookahead_moves = 0.0
    opp_moves_next_level = 0.0
    for move in own_moves:
        lookahead_game = game.forecast_move(move)
        lookahead_moves += len(lookahead_game.get_legal_moves(player))
        # opponent moves
        opp_moves_next_level +=
len(lookahead_game.get_legal_moves(game.get_opponent(player)))
    lookahead_moves /= len(own_moves)
    # get the average moves that the opponent have
    opp_moves_next_level /= len(own_moves)

    if lookahead_moves == 0:
```

```
        # loosing game in the future
        #return float("-inf")
        pass

    score =  mixing_factor * (len(own_moves) - opp_moves) + (1 - mixing_factor) *
(lookahead_moves - opp_moves_next_level)
    return float(score)
```

Tournament results for custom_score_lookahead_both(), mixing_factor = 0.3:

```
This script evaluates the performance of the custom heuristic function by
comparing the strength of an agent using iterative deepening (ID) search with
alpha-beta pruning against the strength rating of agents using other heuristic
functions.  The `ID_Improved` agent provides a baseline by measuring the
performance of a basic agent using Iterative Deepening and the "improved"
heuristic (from lecture) on your hardware.  The `Student` agent then measures
the performance of Iterative Deepening and the custom heuristic against the
same opponents.


*************************
 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: 11 to 9
  Match 4: ID_Improved vs MM_Improved     Result: 12 to 8
  Match 5: ID_Improved vs   AB_Null       Result: 12 to 8
  Match 6: ID_Improved vs   AB_Open       Result: 15 to 5
  Match 7: ID_Improved vs AB_Improved     Result: 10 to 10


Results:
----------
ID_Improved         67.86%

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

Playing Matches:
----------
  Match 1:   Student   vs   Random        Result: 18 to 2
  Match 2:   Student   vs   MM_Null       Result: 16 to 4
  Match 3:   Student   vs   MM_Open       Result: 12 to 8
  Match 4:   Student   vs MM_Improved     Result: 12 to 8
  Match 5:   Student   vs   AB_Null       Result: 17 to 3
  Match 6:   Student   vs   AB_Open       Result: 11 to 9
  Match 7:   Student   vs AB_Improved     Result: 12 to 8


Results:
----------
Student             70.00%
```

Tournament results for custom_score_lookahead_both(), mixing_factor = 0.7:

This script evaluates the performance of the custom heuristic function by comparing the strength of an agent using iterative deepening (ID) search with alpha-beta pruning against the strength rating of agents using other heuristic functions. The `ID_Improved` agent provides a baseline by measuring the performance of a basic agent using Iterative Deepening and the "improved" heuristic (from lecture) on your hardware. The `Student` agent then measures the performance of Iterative Deepening and the custom heuristic against the same opponents.

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

Playing Matches:
----------
  Match 1: ID_Improved vs   Random      Result: 20 to 0
  Match 2: ID_Improved vs   MM_Null     Result: 15 to 5
  Match 3: ID_Improved vs   MM_Open     Result: 16 to 4
  Match 4: ID_Improved vs MM_Improved   Result: 8 to 12
  Match 5: ID_Improved vs   AB_Null     Result: 12 to 8
  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         70.00%

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

Playing Matches:
----------
  Match 1:   Student   vs   Random      Result: 18 to 2
  Match 2:   Student   vs   MM_Null     Result: 16 to 4
  Match 3:   Student   vs   MM_Open     Result: 15 to 5
  Match 4:   Student   vs MM_Improved   Result: 11 to 9
  Match 5:   Student   vs   AB_Null     Result: 18 to 2
  Match 6:   Student   vs   AB_Open     Result: 12 to 8
  Match 7:   Student   vs AB_Improved   Result: 12 to 8


Results:
----------
Student             72.86%
```

Tournament results for custom_score_lookahead_both(), mixing_factor = 0.5:

```
This script evaluates the performance of the custom heuristic function by
comparing the strength of an agent using iterative deepening (ID) search with
alpha-beta pruning against the strength rating of agents using other heuristic
functions.  The `ID_Improved` agent provides a baseline by measuring the
performance of a basic agent using Iterative Deepening and the "improved"
heuristic (from lecture) on your hardware.  The `Student` agent then measures
the performance of Iterative Deepening and the custom heuristic against the
same opponents.


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

Playing Matches:
----------
  Match 1: ID_Improved vs   Random      Result: 17 to 3
  Match 2: ID_Improved vs   MM_Null     Result: 16 to 4
  Match 3: ID_Improved vs   MM_Open     Result: 14 to 6
  Match 4: ID_Improved vs MM_Improved   Result: 10 to 10
  Match 5: ID_Improved vs   AB_Null     Result: 12 to 8
  Match 6: ID_Improved vs   AB_Open     Result: 14 to 6
  Match 7: ID_Improved vs AB_Improved   Result: 10 to 10


Results:
----------
ID_Improved         66.43%

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

Playing Matches:
----------
  Match 1:   Student   vs   Random      Result: 19 to 1
  Match 2:   Student   vs   MM_Null     Result: 18 to 2
  Match 3:   Student   vs   MM_Open     Result: 12 to 8
  Match 4:   Student   vs MM_Improved   Result: 13 to 7
  Match 5:   Student   vs   AB_Null     Result: 16 to 4
  Match 6:   Student   vs   AB_Open     Result: 11 to 9
  Match 7:   Student   vs AB_Improved   Result: 12 to 8


Results:
----------
Student             72.14%
```