Premise

I have experimented with multiple combinations of heuristics individually and then clubbing them in score values. Heuristics explored are based on following premises:

1. Centrality of the move - Moves leading to center should be better (less probable to get blocked)
2. Location of the player with respect to the center
3. Number of common moves available: number of moves which both player have in common (degree of freedom for blocking)
4. Number of games moves which have passed - to change strategy as the games goes older

custom_score

The evaluation logic augments the improved_score function by adding to it sum of centrality of all moves of the player. This should have favoured moves which would lead to center of the board.
It calculates the legal moves of both the players for calculating the difference of moves. However centrality of only the player was calculated.
In the runs, it did not improve the performance much as the average win rate was 98% of that of improved score version. However it won 58% times against improved score function.

```
opp = game.get_opponent(player)
opp_moves = game.get_legal_moves(opp)
p_moves = game.get_legal_moves()
if not opp_moves:
   return float("inf")
if not p_moves:
   return float("-inf")
return float(len(p_moves) - len(opp_moves) + sum(centrality(game, m) for
m in p_moves))
```

custom_score_2

The intuition of the function is to prefer moves which would not lead to a lot of common moves. This would lead to diverging ways for both the players. It computes legal values of both players and then computes the common legal moves.

The results were better than I would have expected, it won nearly 64% times as against 69% of improved score function.

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

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

return float(freedom_from_common_moves(game, player))
```

custom_score_3

The intuition of the function is to use the improved_score function if the player is in middle of the board or else use the non common moves as in custom_score_2.
It performed better than custom_score_2 however was way behind the improved score function which it used for half of the board.
It considers both the players and computes their legal moves.
I would consider it as a failed experiment as it could not improve the performance of improved score.

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

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

ploc_x, ploc_y = game.get_player_location(player)
oloc_x, oloc_y = game.get_player_location(game.get_opponent(player))
pmoves = game.get_legal_moves(player)
omoves = game.get_legal_moves(game.get_opponent(player))

if math.fabs(ploc_x - oloc_x) >= game.width / 2 and math.fabs(ploc_y -
oloc_y) >= game.height / 2:
    return float(len(pmoves) - len(omoves))
return float(freedom_from_common_moves(game, player))
```

custom_score_4

The intuition of the function is to calculate the euclidian distance between the two players. This would mean that the moves which are farther would be preferred (more defensive).
It is computationally inexpensive as it did not even compute legal moves of any of the players. However it considers both players as it needs their current location.
In my experiments, this was the best strategy as it beat the improved score results by around 105% (median of 4 runs).

```
ploc_x, ploc_y = game.get_player_location(player)
oloc_x, oloc_y = game.get_player_location(game.get_opponent(player))

return float((ploc_x-oloc_x)**2 + (ploc_y-oloc_y)**2)
```

custom_score_5

The intuition here is to augment the improved_score by adding centrality of the player location. The experiment however failed to improve the score. Its win rate was 61% as against 69% of improved score strategy.

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

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

pmoves = len(game.get_legal_moves())
omoves = len(game.get_legal_moves(game.get_opponent(player)))

return float(pmoves - omoves + centrality(game,
game.get_player_location(player)))
```

custom_score_6

Experiments with a combination of freedom of common moves and centrality of such moves. Achieved 60.0% in tournament as opposed to 61.4% of improved_score.

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

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

return        float(freedom_from_common_moves(game,        player)        +
best_common_move_to_center(game, player))
```

custom_score_7

Use freedom from common moves as a parameter in beginning and then use distance between players.
Achieved 64.3% in tournament as opposed to 61.4% of improved_score.

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

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

if game.move_count <= 5:
    return custom_score_2(game, player)
return custom_score_4(game, player)
```

Performance

I ran four tournaments with six different custom functions to test premises explained above. The results were than compared for

1. Winrates (average, median)
2. Winrates viz a viz improved score function
3. Winrates for group of other functions - for instance euclidian function won nearly 80% times against the center score function

Win rates of the custom functions per run

Logic	Algo	Run1	Run2	Run3	Run4
euclidian distance between players	AB_Custom_4	70.00%	67.10%	70.00%	78.60%
improved score	AB_Improved	68.60%	72.90%	64.30%	71.40%
sum of centrality + improved_score	AB_Custom	64.30%	72.90%	65.70%	68.60%
in center of board use improved score where as use freedom from common moves in sides	AB_Custom_3	62.90%	62.90%	67.10%	71.40%
freedom from common moves	AB_Custom_2	68.60%	64.30%	58.60%	64.30%
centrality of current loc + improved_score	AB_Custom_5	65.70%	57.10%	57.10%	65.70%
freedom from common moves + best common move towards center	AB_Custom_7	58.60%	62.90%	61.40%	55.70%
use freedom from common moves in beginning and then euclidian distance between players	AB_Custom_6	55.70%	61.40%	55.70%	61.40%

Win rates as a percent of win rates of improved score of the custom functions per run

	Run1 as Improved%	Run2 as Improved%	Run3 as Improved%	Run3 as Improved%	Avg	**Median**
AB_Custom_4	102%	92%	109%	110%	103%	105%
AB_Improved	100%	100%	100%	100%	100%	100%
AB_Custom	94%	100%	102%	96%	98%	98%
AB_Custom_3	92%	86%	104%	100%	96%	96%
AB_Custom_2	100%	88%	91%	90%	92%	91%
AB_Custom_5	96%	78%	89%	92%	89%	90%
AB_Custom_7	85%	86%	95%	78%	86%	86%
AB_Custom_6	81%	84%	87%	86%	85%	85%

Win rates against specific standard functions

	AVERAGE of AB_Custom	AVERAGE of AB_Custom_2	AVERAGE of AB_Custom_3	AVERAGE of AB_Custom_4	AVERAGE of AB_Custom_5	AVERAGE of AB_Custom_6	AVERAGE of AB_Custom_7
Center	70%	64%	66%	79%	64%	63%	69%
Improved	58%	56%	63%	58%	45%	45%	46%
Open	64%	58%	55%	66%	56%	51%	49%
Random	93%	93%	95%	95%	100%	93%	90%

Raw Data of runs

```
********* Run1 ********
Opponent  !  AB_Improved !AB_Custom  !AB_Custom_2 !AB_Custom_3 !AB_Custom_4  !AB_Custom_5  !AB_Custom_6  !AB_Custom_7
           ! Won | Lost !Won | Lost ! Won | Lost ! Won | Lost ! Won | Lost  ! Won | Lost  ! Won | Lost  ! Won | Lost
  Random    !  9  |  1  ! 8  |  2  ! 9  |  1  ! 9  |  1  ! 10 |  0   ! 10 |  0  ! 10 |  0  ! 10 |  0
  MM_Open   !  8  |  2  ! 6  |  4  ! 7  |  3  ! 5  |  5  ! 8  |  2   ! 8  |  2  ! 5  |  5  ! 6  |  4
  AB_Open   !  3  |  7  ! 6  |  4  ! 5  |  5  ! 4  |  6  ! 4  |  6   ! 5  |  5  ! 3  |  7  ! 2  |  8
 MM_Center  !  8  |  2  !10  |  0  ! 9  |  1  !10  |  0  ! 8  |  2   !10  |  0  ! 8  |  2  ! 6  |  4
 AB_Center  !  7  |  3  ! 5  |  5  ! 5  |  5  ! 5  |  5  ! 7  |  3   ! 4  |  6  ! 5  |  5  ! 7  |  3
MM_Improved !  8  |  2  ! 5  |  5  ! 9  |  1  ! 6  |  4  ! 7  |  3   ! 8  |  2  ! 5  |  5  ! 6  |  4
AB_Improved !  5  |  5  ! 5  |  5  ! 4  |  6  ! 5  |  5  ! 5  |  5   ! 1  |  9  ! 3  |  7  ! 4  |  6
-------------------------------------------------------------------------------------------------------
  Win Rate: !   68.6%    !  64.3%    !   68.6%    !   62.9%    !   70.0%     !   65.7%     !   55.7%     !   58.6%

********* Run2 ********
Opponent  !  AB_Improved !AB_Custom  !AB_Custom_2 !AB_Custom_3 !AB_Custom_4  !AB_Custom_5  !AB_Custom_6  !AB_Custom_7
           ! Won | Lost !Won | Lost ! Won | Lost ! Won | Lost ! Won | Lost  ! Won | Lost  ! Won | Lost  ! Won | Lost
  Random    ! 10  |  0  !10  |  0  ! 9  |  1  ! 10 |  0  ! 9  |  1   ! 10 |  0  ! 8  |  2  ! 8  |  2
  MM_Open   !  7  |  3  ! 8  |  2  ! 8  |  2  ! 8  |  2  ! 7  |  3   ! 5  |  5  ! 8  |  2  ! 7  |  3
  AB_Open   !  6  |  4  ! 8  |  2  ! 4  |  6  ! 5  |  5  ! 4  |  6   ! 4  |  6  ! 4  |  6  ! 6  |  4
 MM_Center  !  9  |  1  ! 8  |  2  ! 7  |  3  ! 6  |  4  ! 10 |  0   ! 10 |  0  ! 8  |  2  ! 8  |  2
 AB_Center  !  6  |  4  ! 6  |  4  ! 6  |  4  ! 4  |  6  ! 7  |  3   ! 2  |  8  ! 6  |  4  ! 6  |  4
MM_Improved !  7  |  3  ! 7  |  3  ! 6  |  4  ! 8  |  2  ! 6  |  4   ! 5  |  5  ! 4  |  6  ! 6  |  4
AB_Improved !  6  |  4  ! 4  |  6  ! 5  |  5  ! 3  |  7  ! 4  |  6   ! 4  |  6  ! 5  |  5  ! 3  |  7
-------------------------------------------------------------------------------------------------------
 Win Rate:  !   72.9%    !  72.9%    !   64.3%    !   62.9%    !   67.1%     !   57.1%     !   61.4%        62.9%

********* Run3 ********
Opponent  !  AB_Improved !AB_Custom  !AB_Custom_2 !AB_Custom_3 !AB_Custom_4  !AB_Custom_5  !AB_Custom_6  !AB_Custom_7
           ! Won | Lost !Won | Lost ! Won | Lost ! Won | Lost ! Won | Lost  ! Won | Lost  ! Won | Lost  ! Won | Lost
  Random    ! 10  |  0  !10  |  0  ! 9  |  1  ! 10 |  0  ! 9  |  1   ! 10 |  0  ! 9  |  1  ! 9  |  1
  MM_Open   !  6  |  4  ! 6  |  4  ! 5  |  5  ! 7  |  3  ! 7  |  3   ! 8  |  2  ! 6  |  4  ! 5  |  5
  AB_Open   !  6  |  4  ! 5  |  5  ! 5  |  5  ! 2  |  8  ! 6  |  4   ! 5  |  5  ! 3  |  7  ! 3  |  7
 MM_Center  !  8  |  2  ! 8  |  2  ! 7  |  3  ! 8  |  2  ! 8  |  2   ! 7  |  3  ! 8  |  2  !10  |  0
 AB_Center  !  3  |  7  ! 4  |  6  ! 6  |  4  ! 6  |  4  ! 7  |  3   ! 2  |  8  ! 3  |  7  ! 5  |  5
MM_Improved !  7  |  3  ! 9  |  1  ! 4  |  6  ! 9  |  1  ! 7  |  3   ! 6  |  4  ! 5  |  5  ! 6  |  4
AB_Improved !  5  |  5  ! 4  |  6  ! 5  |  5  ! 5  |  5  ! 5  |  5   ! 2  |  8  ! 5  |  5  ! 5  |  5
-------------------------------------------------------------------------------------------------------
 Win Rate:  !   64.3%    !  65.7%    !   58.6%    !   67.1%    !   70.0%     !   57.1%     !   55.7%     !   61.4%

********* Run4 ********
Opponent  !  AB_Improved !AB_Custom  !AB_Custom_2 !AB_Custom_3 !AB_Custom_4  !AB_Custom_5  !AB_Custom_6  !AB_Custom_7
           ! Won | Lost !Won | Lost ! Won | Lost ! Won | Lost ! Won | Lost  ! Won | Lost  ! Won | Lost  ! Won | Lost
  Random    !  9  |  1  ! 9  |  1  !10  |  0  ! 9  |  1  !10  |  0   ! 10 |  0  !10  |  0  ! 9  |  1
  MM_Open   !  6  |  4  ! 6  |  4  ! 7  |  3  ! 7  |  3  !10  |  0   ! 6  |  4  ! 5  |  5  ! 6  |  4
  AB_Open   !  6  |  4  ! 6  |  4  ! 5  |  5  ! 6  |  4  ! 7  |  3   ! 4  |  6  ! 7  |  3  ! 4  |  6
 MM_Center  ! 10  |  0  ! 9  |  1  ! 7  |  3  ! 9  |  1  !10  |  0   !10  |  0  ! 8  |  2  ! 9  |  1
 AB_Center  !  5  |  5  ! 6  |  4  ! 4  |  6  ! 5  |  5  ! 6  |  4   ! 6  |  4  ! 4  |  6  ! 4  |  6
MM_Improved !  9  |  1  ! 7  |  3  ! 8  |  2  !10  |  0  ! 8  |  2   ! 8  |  2  ! 6  |  4  ! 4  |  6
AB_Improved !  5  |  5  ! 5  |  5  ! 4  |  6  ! 4  |  6  ! 4  |  6   ! 2  |  8  ! 3  |  7  ! 3  |  7
-------------------------------------------------------------------------------------------------------
 Win Rate:  !   71.4%    !  68.5%    !   64.3%    !   71.4%    !   78.6%     !   65.7%     !   61.4%     !   55.7%
```

Recommendations

As per the analysis and intuition I would recommend the custom_score_4 function which uses the euclidian distance between players current location.

1. It would favour positions which are taking the two players far away (a defence stance)
2. It is computationally inexpensive as it does not even need to calculate the legal moves
3. It considers both the players
4. It resulted in nearly 105% winr ates using improved score winrates as baseline
5. It won against improved score function ~58% times.