This is an analysis of the custom heuristics in the Isolation project.

Heuristic 1:

Heuristic 1 is based on the premise of maximizing player 1's available moves in the late game.

Its performance is captured in the following table:

	Won	Lost	Win Rate
Random	10	0	100%
MM_Open	6	4	60%
MM_Center	8	2	80%
MM-Improved	7	3	70%
AB_Open	5	5	50%
AB_Center	5	5	50%
AB_Improved	6	4	60%
Total	47	23	67%

Heuristic 1 is implemented with an evaluation that measures the number of legal moves for player 1, weighted by the inverse of the number of unvisited spaces left.

It is implemented as follows:
player_moves_left = len(game.get_legal_moves(player))
unvisited = len(game.get_blank_spaces())
return float(player_moves_left)/float(unvisited)

Heuristic 2:

Heuristic 2 is based on the premise that a player has more move options nearer the center of the board than at the sides and corners.

Its performance is captured in the following table:

	Won	Lost	Win Rate
Random	8	2	80%
MM_Open	5	5	50%
MM_Center	8	2	80%
MM-Improved	6	4	60%
AB_Open	6	4	60%
AB_Center	7	3	70%
AB_Improved	4	6	40%
Total	44	26	62%

Heuristic 2 is implemented with an evaluation measuring the difference between player 2 and player 1's distances from the center of the board.

It is implemented as follows:
center_r = 3
center_c = 3
player_r = game.get_player_location(player)[0]
player_c = game.get_player_location(player)[1]
oppo_r = game.get_player_location(game.get_opponent(player))[0]
oppo_c = game.get_player_location(game.get_opponent(player))[1]
return float(abs(oppo_c - center_c) + abs(oppo_r - center_r) - abs(player_r - center_r) - abs(player_c - center_c))

Heuristic 3:

Heuristic 3 is based on the premise of minimizing the opponent's legal moves.

Its performance is captured in the following table:

	Won	**Lost**	**Win Rate**
Random	8	2	80%
MM_Open	7	3	70%
MM_Center	7	3	70%
MM-Improved	5	5	50%
AB_Open	5	5	50%
AB_Center	4	6	40%
AB_Improved	2	8	20%
Total	38	32	54%

Heuristic 3 is implemented with an evaluation measuring the difference between the maximum number of legal moves at any one point and the actual number of legal moves the opponent has.

It is implemented as follows:
opponent_moves_left = len(game.get_legal_moves(game.get_opponent(player)))
return float(8 - opponent_moves_left)

Conclusion:

The following table compares the performances of the three custom heuristics with the AB_Improved model:

	AB_Improved Win Rate	**Heuristic 1 Win Rate**	**Heuristic 2 Win Rate**	**Heuristic 3 Win Rate**
Random	90%	100%	80%	80%
MM_Open	60%	60%	50%	70%
MM_Center	70%	80%	80%	70%

MM-Improved	80%	70%	60%	50%
AB_Open	60%	50%	60%	50%
AB_Center	60%	50%	70%	40%
AB_Improved	50%	60%	40%	20%
Total	67%	67%	62%	54%

Heuristic 3 performed the worst. While it was the simplest function, it did not take into consideration the remaining legal moves for player 1. This likely gave rise to situations where player 1 would over-aggressively seek to minimize the remaining legal moves for player 2, while putting itself in jeopardy.

Heuristic 2 performed reasonably. It had the most computationally expensive heuristic and did not account for late game scenarios where there might be more blank spaces left at the sides instead of the center. Heuristic 2 likely performs better in early games than in late games. Potential to explore pairing this with a late game heuristic.

Heuristic 1 performed joint-best with AB_Improved. It took into consideration that most games in Isolation would likely be late-game finishes, and sought to maximize the available moves for player 1 in late games, instead of seeking an advantage in the early-game at the cost of a potential longer-term disadvantage.

Heuristic 1's strong performance and intuitive rationale makes it a good choice as the evaluation heuristic for our Isolation agent.