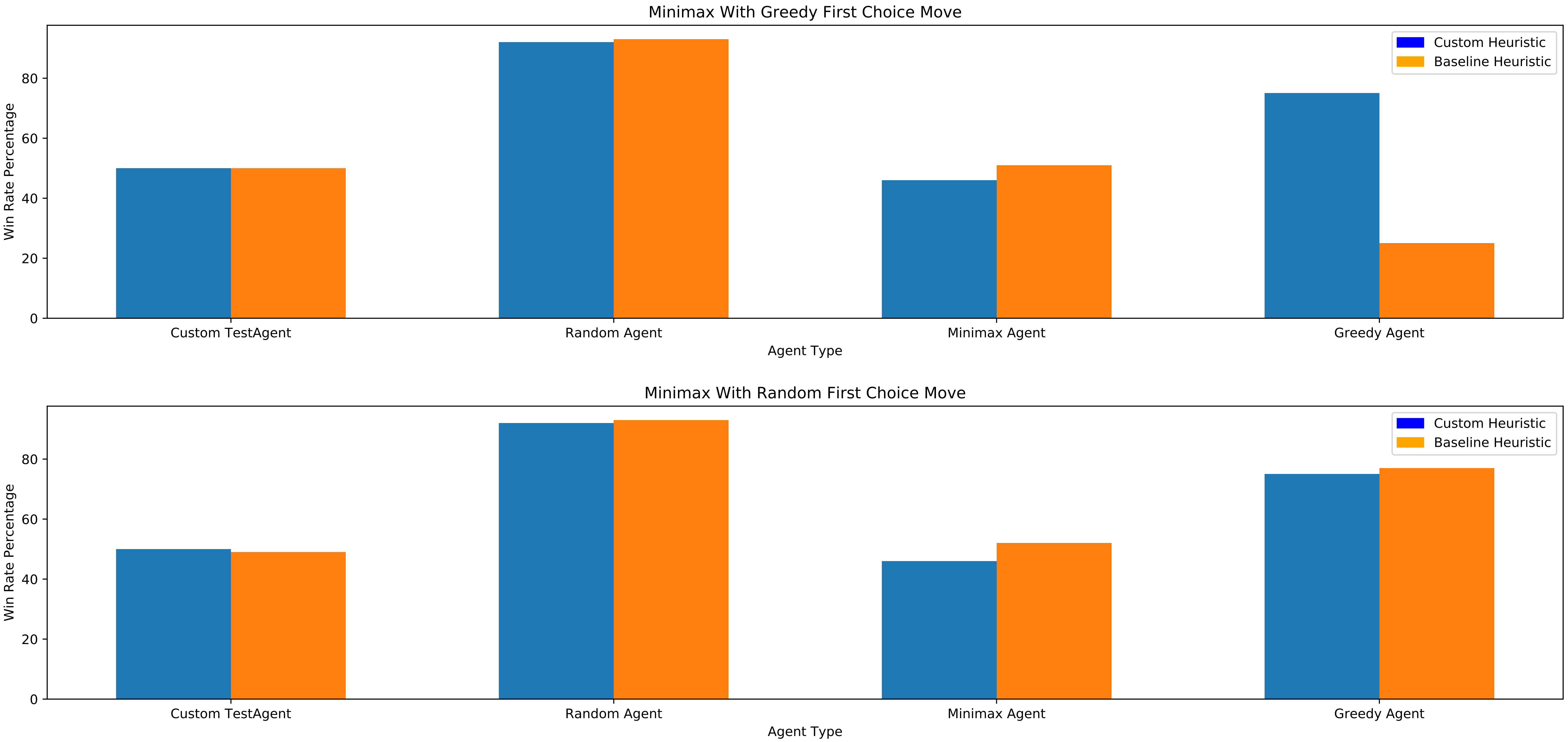
400 Games With Fair Matches



Above are two bar graphs showing my custom agents' win rate percentage using my custom heuristic (colored blue) and a baseline heuristic (colored orange) after 400 games with fair matches against Random Agent, Greedy Agent, Minimax Agent, and Custom TestAgent. These two heuristics can be described with two explanations:
+ My custom heuristic calculated the distance of the current player from the center square then returned this distance as a negative value. This simulated choosing the move that minimizes the distance from the player's current location to the center square as much as possible.
+ The baseline heuristic used Udacity's #my_moves - #opponent_moves calculation which simulated achieving a maximum amount of moves more than the opponent.

My custom agents involved in the top graph used a greedy heuristic to choose the first move while my custom agents in the bottom graph randomly chose the first move. Both agents then utilized the minimax search algorithm up through the fourth level of the search tree. If more search is required to find the optimal move after the fourth level of the search tree, the agent then used the heuristic calculation to choose a move from the current player's position.

I simulated two different ways for the agent to make a first move to see what kind of effect the first move might have on the agent's performance. Using a greedy first move resulted in nearly identical results to using a randomly chosen first move with one exception. When using the baseline heuristic against the Greedy Agent with a greedy first move, the agent loses all of the first 200 games then wins 50% of the last 200 games (i.e. the fair-match games), resulting with a 25% overall win rate. Without this exception, the above graphs show that the baseline heuristic performs the same as or slightly better than my custom heuristic.

Advanced Heuristic Analysis

What features of the game does your heuristic incorporate, and why do you think those features matter in evaluating states during search?

My custom heuristic incorporates choosing locations that are closest to the center square. Being close to the center square is important because a knight chess piece generally has more locations to move to when in the middle of the board and therefore indirectly incorporates the number of liberties feature at each state. The number of liberties at each state is important because at least one liberty is required to remain in the game and having more liberties than the opponent can increase the chance to win.

Analyze the search depth your agent achieves using your custom heuristic. Does search speed matter more or less than accuracy to the performance of your heuristic?

My agent achieves a search depth of four. Any search depth greater than four has a chance to take longer than the default time constraint placed on each move. Any search depth less than four causes the agent to guess an optimal move that could have been found otherwise. Because a search agent with infinite time to make a move can search more than 13 levels of nodes (which is more than three times more levels and exponentially more nodes) on an 11 by 9 Isolation board, heuristic accuracy is more important than search speed when under the default time constraints for making a move. As the search space becomes smaller, the agent is able to find a higher percentage of optimal moves relative to the total amount of optimal moves possible and thus search speed becomes more important than heuristic accuracy (because the relevance of the heuristic goes to zero as the search space approaches zero).