Matthew Ragoza
mtr22@pitt.edu

Assignment 4

ISSP 2160
October 2, 2021

Problem 1. Genetic algorithm

To determine the effect of different parameters on the ability of the genetic algorithm to find an optimal tour on the traveling salesman problem, evaluations were performed on ranges of values for each of the following parameters: number of generations, population size, mutation probability, culling percentage, and elite percentage. With each parameter setting, the algorithm was run ten times with different random seeds, and the mean and minimum tour distances achieved using each setting are plotted in Figure 1.

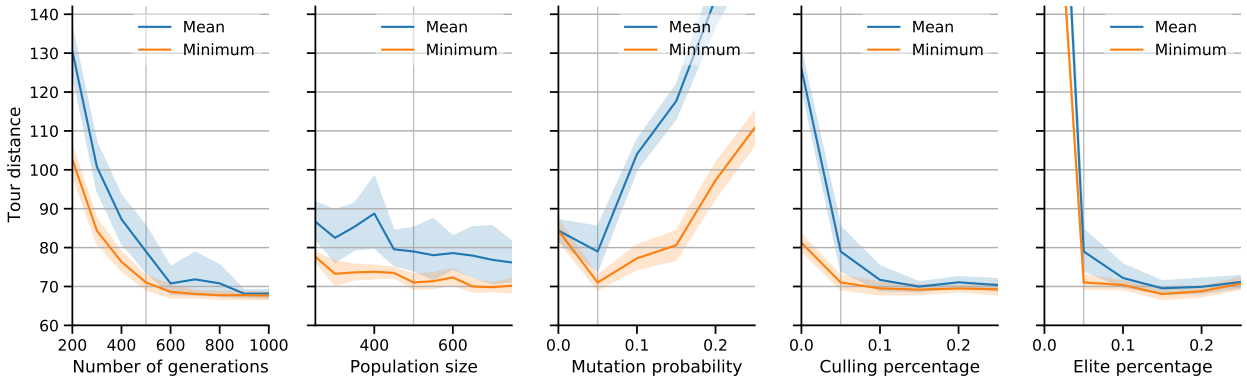

Figure 1: Evaluation of genetic algorithm parameters. The distributions of mean and minimum tour distance for ten different random seeds of each parameter setting are displayed here. The vertical gray lines in each plot mark the default values for the parameters.

a. **Number of generations.** The number of generations had a strong effect on the tour distance metrics. Decreasing it from the default value resulted in an exponential increase in the tour distance found by the algorithm, while increasing it further allowed improved tours to be recovered. At the highest number of generations, the mean population tour distance converged with the minimum tour distance, suggesting that the optimal tour was consistently recovered prior to generation 1000. For this reason, 700 or 800 generations are likely sufficient.

b. **Population size.** Population size had a more modest effect on the genetic algorithm's performance. While slightly worse mean and minimum tour distances were found by decreasing the population size, the effect was much smaller than that of the number of generations. In addition, increasing the population size improved the mean and minimum tour distances, but the fact that the mean tour distance did not converge with the minimum tour distance at any population size implied that increasing this parameter was less reliable at finding the optimal tour. The ideal population size appears to be 650 for this problem.

c. **Mutation probability.** The tour distance was very sensitive to the mutation probability, but the default value of 5% was the best value tested. When mutations were

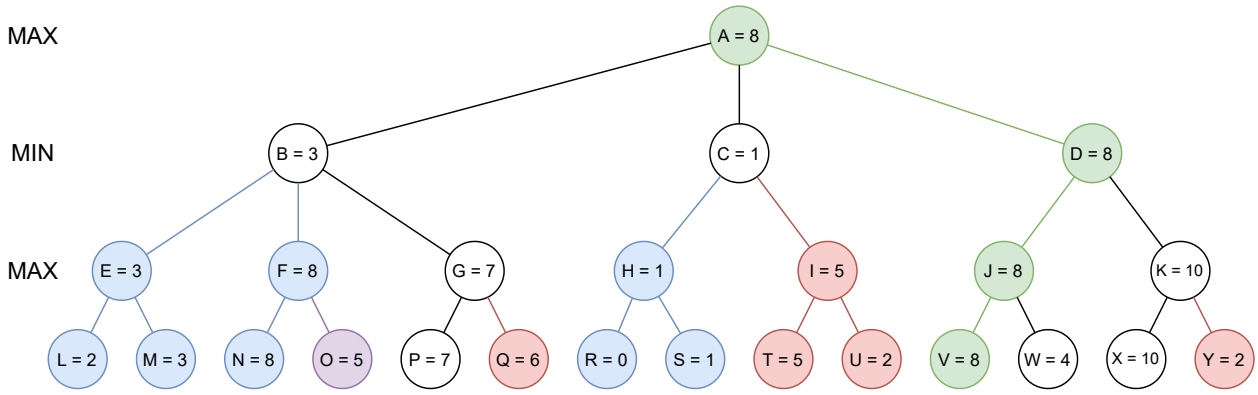

Figure 2: Minimax values for the provided adversarial search tree, using left-to-right traversal. The solution path of rational players is highlighted in green. Subtrees that would be pruned by the alpha-beta algorithm are shown in red for left-to-right traversal, blue for right-to-left traversal, and purple if they are pruned in either traversal order.

more frequent, the mean and minimum tour distance increased rapidly. Surprisingly, omitting mutations entirely also hurt performance–but much more modestly. Since no mutations occurred with this setting, it makes sense that the minimum tour distance was very close to the mean. With each successive generation, suboptimal tours were culled from the population, but no mutated tours were introduced. Therefore, the population became less diverse over successive generations.

d. **Culling percentage.** The algorithm's performance was also affected significantly by the amount of culling. When no culling was performed, the mean and minimum tour distances both increased, but the mean distance increased more sharply. This is expected since removing the worst tours from the population only affects the minimum tour distance indirectly, by reducing the likelihood of crossover between good and bad tours. Increasing the culling percentage to around 15% allowed slightly improved statistics, but there were no gains to be made beyond that value.

e. **Elite percentage.** Finally, the elite selection percentage had a similarly strong impact on the recovered tour quality. A drastic increase in both mean and minimum tour distance was seen when no elite tours were retained through successive generations. Without elite selection, the best tours were likely to change through crossover and mutation, which often made them worse. In contrast, increasing the amount of elite selection to 15% improved the recovered tour distances, and the minimum tour distance approached the hypothesized optimal tour distance. The performance started to decay as elite selection increased beyond 15%.

Assignment 4

Problem 2. Adversarial search

a. The minimax values of the nodes in the adversarial search tree are shown in Figure 2. The optimal move for the first player is $A \rightarrow D$, and the solution path of rational players would be $A \rightarrow D \rightarrow J \rightarrow V$.

b. The nodes that are pruned by the alpha-beta algorithm in left-to-right order are I, O, Q, T, U, and Y.

c. The nodes that are pruned by the alpha-beta algorithm in right-to-left order are E, F, H, L, M, N, O, R, and S.

Problem 3. Tic-tac-toe-10 player

a. The first aim was to evaluate the difference in game-playing performance when using the provided basic heuristics compared with naive heuristics. Ten matches were played between players, one using each of the heuristic functions. Each player made the first move on 5 matches and they both used 2-ply search. As seen in Figure 3, the player that used basic heuristics won 9 out of 10 matches, and there was one draw. This is because the naive heuristics only used three patterns to evaluate board positions, and each involved having at least three marks in a row. As a result, the player made completely random moves in the early game, and relied on getting three in a row by chance in order to start trying to get five in a row. In addition, none of its patterns involved mixes of each player's markers, so it had a very limited understanding of blocking moves. The basic heuristic function used more patterns to evaluate board positions, including patterns with between one and five markers, and one pattern specifically encouraging a blocking move. Therefore, it was more effective at getting to desirable positions–especially in the early game.

Player	Wins	Losses	Draws
Basic heuristics	9	0	1
Naive heuristics	0	9	1

Figure 3: Basic heuristics vs. naive heuristics.

b. The next step was to determine the degree of first-mover advantage in this game. An experiment was run pitting two identical agents against one another in 10 matches, each using basic heuristics and 2-ply search, but always giving one player the first move. As shown in Figure 4, the outcome was one win and one loss for each player, with 8 draws. This implies that there was very little inherent advantage to making the first move, though it is difficult to say for sure that there is no advantage at all due to the small sample size.

Player	Wins	Losses	Draws
First move	1	1	8
Second move	1	1	8

Figure 4: One player always makes the first move.

c. The third analysis was on the effect of adversarial search depth on game-playing ability. To determine this, 10 matches were played between agents both using basic heuristics, and alternating the first move, but one used 2-ply search and the other used 1-ply search. The outcome of this was that the 2-ply player won all 10 matches, as seen in Figure 5. This highlights the importance of accounting for the opponent's moves in 2-player games: the 1-ply agent only considered their own next move followed by heuristic evaluation, while the 2-ply agent tried to pick the best move after taking into account how an optimal opponent would respond. Especially considering that the two agents used the same heuristic function, it is not surprising that the 2-ply search was superior, since it had an exact model of the opponent's behavior. The 2-ply search process was literally one step ahead at all times.

Player	Wins	Losses	Draws
2-ply search	10	0	0
1-ply search	0	10	0

Figure 5: 2-ply search vs. 1-ply search.

d. Finally, I wrote my own custom heuristic function to try and outperform the basic heuristics and compete against my classmates. To develop this, I first added a feature to `player.py` to output the heuristic values of each open board position prior to printing the board itself at each step. To speed up the process and better understand the heuristic patterns, I tested against the basic player using a smaller board size and 1-ply search, which made the games quicker and allowed direct observation of effects of different patterns on leaf node board evaluations. Then, starting with an empty pattern dictionary, I ran games one after another and decided where I would play on a given turn, trying to formalize the reasons why in terms of the linear marker patterns on the board. I set the heuristic values of the patterns to the minimum required in order for the player to make the move I wanted when the pattern appeared.

   By encoding my own moves into the pattern dictionary, the agent's behavior grew more complex over time and started to line up with my own estimated best-moves. When the agent's behavior was unexpected and I could not understand the advantage, I looked at the difference in heuristic values on the board between the move they made and the move I wanted to make, and then increased the score of the desired patterns accordingly, or added the patterns to the dictionary if they did not exist. As my agent started to beat the basic player consistently, I steadily increased the game difficulty by

increasing the board size, giving the opponent first-mover advantage, and eventually by playing against a 2-ply basic player using only 1-ply search with my own heuristics.

When I was satisfied with the quality of my heuristics, I ran a final evaluation by playing 10 matches between 2-ply agents on a full 10 by 10 board, my heuristics versus basic heuristics, and alternating the first move. The results are shown in Figure 6. My player was able to win 4 out of 10 games against the basic player, with the other 6 games ending in draws, validating that the changes I made to the heuristics improved the game-playing ability.

Player	Wins	Losses	Draws
Basic heuristics	0	4	6
My heuristics	4	0	6

Figure 6: Basic heuristics vs. my heuristics.

One of the simplest improvements my heuristics had over the basic patterns was symmetry: since the patterns were checked with simple sub-string matching, it was necessary to add reversed versions of asymmetric patterns to ensure that they were recognized in different directions. I also added far more patterns involving the opponent's markers to improve the blocking ability of the player, and many additional patterns that were unintuitive at first but resulted from me verbalizing why the moves that I would have made in a given scenario were advantageous. Many of these patterns had low score values, emphasizing that they did not contribute greatly to my decisions, but the sum of many small patterns can lead to a superior board position many turns down the line.