REPORT:

1) *A statement of the overall goal of the project*

Our primary goal for this project was to create an evolutionary algorithm which would evolve a AI for the game of Quoridor. Ideally, we hoped to make it human competitive, but at a minimum we want it to be better than a randomly generated AI. Smaller goals included exploring methods of making expensive fitness evaluation more computationally tractable through multithreading.

2) *A description of evolution algorithm include:*

   a. ***Representation and how individuals are initialized. Show example of an individual.***
   An individual is represented as an array of weights, and can be stored in plaintext in the following format:

   -4.93835806846 -1.59731149673 2.08922505378 -50.5754323005 18.437635898590088

   Each of these values corresponds to the weight of a node in the neural network that comprises each AI.

   b. ***Description of operators/parameters (pop size, mutation types/rates, crossover types/rates***

   Our parameters are, NUM_THREADS, POPULATION_SIZE, GENERATIONS, TOURNAMENT_SIZE and CROSSOVER.
   As of now, we have explored populations from 16 to 512. Unfortunately, because the actual fitness evaluation are expensive computationally, a population of 512 can take well over 30 seconds per generation – we will discuss the implications of this later in the report. Our mutation operator is a +/- Gaussian value (easily customizable – we were playing around with numbers between one and twenty) on every weight of a given AI. Crossover is uniform crossover on all of the weights between two individuals. Crossover can also be turned off completely. Tournament size is an adjustable variable. We think that increasing tSize strengthens selection, but the tSize also has a practical limit of PopSize/numThreads because the tournaments occur thread-locally so to make “very strong” selection, you would either need to decrease the thread count or increase the population size.

   c. ***Explanation of how we chose parameters (theoretical reasons, other things we tried that led to these etc…***
   We based our parameter choices based on some pieces of information that we found useful when playing test games of Quoridor as humans, and decided to give that information to the AI and see what it determined was most useful. We took concepts such as going towards the side of a wall which had an odd number of openings or distance from the end and made them our parameter nodes based on these initial hunches.

   One important note is that we realized halfway through the project that our random node caused evolution to slow drastically. Instead of removing it entirely, we now initialize it to zero so that it plays less of a role throughout the population.

d. Best individuals evolved and best individuals we could hand write.

We began testing arbitrary weights against a random AI. This means that we gave all nodes a weight of 0 except one which was set to an arbitrarily chosen weight of -50. We then had the AIs play each other a total of 5 times and recorded victories / draws / losses. When we tried this, only distanceFromEnd consistently beat the random AI. The highlight is that when distanceFromEnd is weighted, the AI is guaranteed to defeat the random 100% of the time – the other weights performed just as well as a random could be expected to against another random (two or three wins /losses).

We additionally tried arbitrary pairs of weights where all are 0 except for any two (there are 10 permutations of this with the 5 nodes we have construed). The results of that test are the following:

- distanceFromEnd will almost always be helpful – it will only lose when paired with distanceFromWall
- distanceFromWall is almost always detrimental – it will only win when paired with distanceFromWall and the opponent assists the player by crafting a 'tunnel' of walls to the end
- the other three – randomMove, odd, wallsLeft – are generally neutral and get average results against a random opponent

In combinations of three, the results were similar. We see again that distanceFromEnd is beneficial, and is only ever brought down to unusable levels when paired with distanceFromWall, which is consistently the most detrimental of all the nodes we have created thus far. The general behavior exhibited by an AI that weighs distanceFromWall is to hug the nearest wall – and then proceed to get stuck in a circular path where the 'optimal' move is to move away from the wall and then back.

This is the exciting part: we discovered through evolution and then tweaked through manual input, an AI which is indeed human competitive. It is capable of defeating human players given that the human player makes some suboptimal moves. The weights of this Ai are the following:

10 -10 0 -50 0

This AI can be tested by replacing the first line of evoOutput.txt and then selecting Player vs. AI on the menu. Our evolutions did not seem able to quite reach this solution, as it evolved most of the weights rather equally – had we more time, we could have tried tweaking with weight mutation a bit more in order to find a similar result.

3) Original Hypothesis/hypotheses we put forward in proposal and how experimental results either confirmed them or not (graphs).

Our original hypotheses did match up fairly well with the limited extent of our experimental results. distanceFromEnd was by far the most influential weight an AI could have – without it, random movement guaranteed suboptimal moves that often led to backtracking and moving backwards for no reason. Basing moves off odd/even lengths of unblocked wall in the end, like

hypothesized, has so little effect that it was basically just random movement as far as we could tell. The one hypothesis that we made that was wrong was that we expected walls to matter a lot when it was in fact the opposite – distanceFromWall was easily the most detrimental node an AI could evolve to heavily weigh, and the wall-hugging that ensued actually made an AI with a high weight for distanceFromWall to be less effective than a wholly random AI.

4) ***Thoughts on the project:***

   a. ***How well did we succeed in original goals***

      We did occasionally generate an AI that made sophisticated moves in certain situations and seemed to genuinely intelligently react to the move of the opponent. Even playing against a human player, while they moved sub-optimally, their wall placements were noted to be effective as a response. However, we were limited in our discovery of these AI's by computational power and constant tweaks of our code. Thus, these AI's were few enough that we cannot claim to have succeeded in creating a human competitive AI. These AI's however did give us significant promise that given more computational power/time we could definitely evolve relatively sophisticated specialists. We have yet to observe good generalists – however, we can determine that of the weights we tested, distance from the goal was definitely an important positional factor to consider when making a move in Quoridor.

   b. ***Where did we run into problems and what solutions did we create?***

      The first problem we ran into was the overall computational complexity of this problem. We were proactive in solving this by designing the EA to be parallel from the beginning, however despite our best efforts, we still did not have enough power and time to evolve a population as far as we would like. Another problem we ran into is that when very poor AI is pitted against other bad AI, the game often will go on for hundreds of thousands of moves. We put in a condition to declare the game a tie after a few thousand moves, but this case is common enough that it slows the algorithm down. Later on we realized that changing this parameter could also help weed out AI's that aren't "motivated" to win. We adjusted the legal moves before calling a tie to 100 per individual. Neither of these individuals progress in the tournament, and subsequently, they will eventually die. This was one modification we made to increase the "aggressiveness" of the AI's. This also drastically improved runtime.

      Another problem we encountered, and one that we didn't have time to weed out by tweaking (though it still sometimes disappears in evolution) is a tendency for an AI to drop all of its walls as soon as possible. This is a byproduct of the "minimize the opponents next best move" attitude we took to programming the AI, and in many cases, it makes the AI very passive. Additionally, the sheer number of such wall placements simply made it highly probable that one of the dozens of wall moves would be chosen over the three to four square-movements an AI could make. Again, what gives us hope that this was a good approach is the few AI that evolved well enough to create sophisticated blocks on their opponents in some situations. Tweaking could have probably gotten the AI's to balance offense and defense more in general though. As is, most AIs that did not move directly toward the end (ie, had high distanceFromEnd weights) ended up doing a lot of backtracing.

Conversely, the AIs that did place such high weight on moving directly for the end effectively did not utilize walls.

Arguably, the biggest problem we encountered is that proper identification of useful nodes is nearly impossible. From the beginning we tried to come up with as many nodes that could be feasibly useful as possible, but our testing shows that only one of them consistently benefits the individual which possesses a strong weight on that node. All in all, the two biggest problems were time and lack of imagination for nodes.

c. What did we learn about (not all necessarily):

i. Technique we used

Our use of neural network was, at the very least, extremely elegant and efficient. Our implementation, in general, seemed to be very efficient – dual-threading and breadth-first-search. However, we were definitely limited by the nodes we decided to choose and spent a large amount of time trying to find the ideal balance between input variables that struck a good balance between effectiveness and time – we wasted several hours at first doing high-generation evolutions that did not output particularly effective AI's.

ii. Problem we tried to solve

While we obviously knew that a turn-based strategy game such as Quoridor could not easily be boiled down to factors as simple as the ones we chose, it was still surprising to see how little most of our chosen nodes mattered outside of distanceFromEnd – in fact, while odd and even were marginally or unnoticeably more effective than random, distanceFromWall was actually detrimental and worse than random movement.

We also discovered that the addition of a random node to eliminate infinite loops, was a poor solution which allowed poor AI's to gain fitness by chance when defeating other poor AI's. When we decreased the number of allowed moves (to increase performance) we also realized that this was an effective manner of eliminating slow-to-win AI. Unfortunately, this tactic resulted in early, unevolved populations NEVER having an AI beat another AI, so the fitness of entire populations stagnated at zero.

iii. Question we asked

We did confirm our hypothesis that the distance from the goal was a critical factor to consider – the AIs developed with a high weight in this factor tended to outperform their colleagues by a large margin. It was the only weight that by itself, was capable of outperforming random movement.

d. Data graphs showing experiments we did or measuring success of the EA how we define it

Due to the nature of our project, there doesn't seem to be an especially meaningful way of reporting our results in a graph format. While we could nebulously define some qualifier to

describe the AIs that were evolved, we believe it makes more sense to just report the types of experiments we did and our general conclusions.

Fixed input: We conducted a gauntlet of 5-game tests with predetermined weights of either -50 or 0. Every possible combination of weights in this fashion was run.

Findings: randomMove, oddDistance, wallsLeft seemed to have little effect and were effectively random. distanceFromWall was found to be a suboptimal node to be minimized, performing much worse than random in almost all cases except in the edge case where a wall 'tunnel' is formed with the assistance of the opponent. distanceFromEnd performed extremely well – in almost every test in which its weight was -50, the AI won 5 / 5 games against the randomly generated AI. Conversely, distanceFromWall lost or drew every game that it was not paired with distanceFromEnd (in which case it caused the AI to perhaps win once in a series of five).

Evolution: We conducted several tests of the best two AIs of our evolved populations, using varied input variables. Factors we tested:

Generations: Numbers from 15 to 1000 were used.
Population Size: Multiples of 16 from 64 to 512 were used.
Tournament Size: Numbers from 2 to 16 were used.
Crossover: This was a toggle-able option that we implemented.

Due to limited time and computer power, a large portion of this project was spent evolving populations that were not optimized. Because of this, we were unable to evolve, for example, a population size of 512 with 1000 generations and tournament size of 16 due to time constraints.

What we discovered was that despite most of efforts, our evolutions trended towards generalist and low-weight AIs that had a high proclivity to place walls. However, they did tend to place a higher weight on distanceFromEnd, which we then used to craft the aforementioned human-competetive AI player. At a few points during testing, we were able to evolve competent players, but generally, these were rather few and were tested with volatile code.