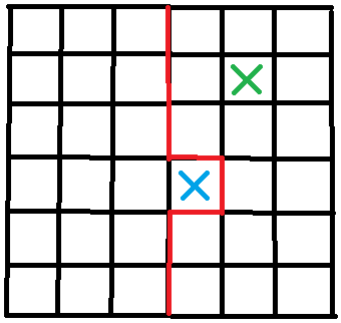
Comp 424: Final Project Report

- An explanation of how your program works, and a motivation for your approach.

  Our approach consist of getting the most efficient win possible in the average case. The strategy is slightly different depending on whether the board size is even or odd.

  Even case:

  The way we achieve the most efficient win on the even boards is by drawing a line in the middle of the board in order to separate the board into two even halves. When the agent reaches the final barrier in the line, it doesn't build it and instead it places itself on the same side as the opponent and build barriers in order to connect the wall and to control one square of the opponent's territory. On a board of size 6 this would look like this where the blue X represents our agent and the green X represents the enemy agent:

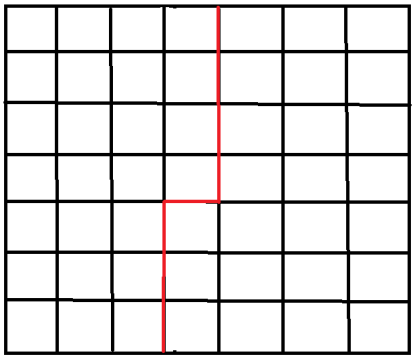


  This enable us to secure $\frac{M^2}{2}+1$ squares which means half of the squares in the board $+1$, so that we can control the minimal amount of squares required for a win.

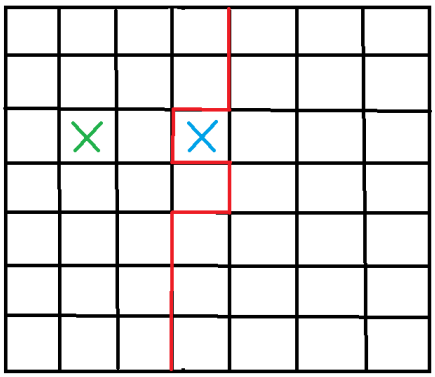
  Odd case:

  We need to find another strategy on odd boards since we cannot draw a line in the middle that evenly splits the board in half. So, for odd boards, we draw a pattern on the middle column such that there is a barrier on the right for the first K rows and a barrier on the left for

the last $K-1$ rows with a barrier on the bottom of the middle square to connect these two walls. This would result in such a pattern on a board of size 7:



This would separate the board in such a way that there are $\lceil \frac{M^2}{2} \rceil$ squares on the left and $\lceil \frac{M^2}{2} \rceil - 1$ squares on the right. The goal of our agent would then be to control $\lceil \frac{M^2}{2} \rceil$ squares. When our agent reaches the last required barrier to build, it needs to act depending on where it is and where the enemy agent is. If our agent is on the left and the enemy agent is on the right or both agents are on the right, then it goes to the left side (or stays there) and just build the last barrier of the wall normally in order to control the left side. If our agent is on the right and the enemy agent is on the left or both agents are on the left, then it places itself on the left side (or stays there) and build barriers in order to control the right side as well as one square of the left side like in the even case. This will allow our agent to control $\lceil \frac{M^2}{2} \rceil$ squares. This is an example of what this would look like with again the blue X representing our agent and the green X representing the enemy agent:



The motivation for our approach comes from the fact that enemy agents usually do inefficient moves in the early stage of the game, which would give us the opportunity to build our wall fast and win the game. We are basically trying to win the game as fast as possible without giving the time to the enemy agent to counter us effectively. We think our strategy will succeed well against other student agents because it is hard to counter without any knowledge of the strategy.

- A brief description of the theoretical basis of the approach (about a half-page in most cases); references to the text of other documents, such as the textbook, are appropriate but not absolutely necessary. If you use algorithms from other sources, briefly describe the algorithm and be sure to cite your source.

  Our approach was not based off of any pre-existing strategy or theorem, but rather off of an observation of the game itself. The goal of the game is to be located in an enclosed section of the board containing more squares than the opponent's section. We identified two main strategies for doing so: separate the board into two sections and make sure you are on the side with the most squares, or search and destroy the opponent. Identifying the former strategy as the most straightforward, upon observation of the board, it is evident that the simplest way to accomplish this was to create a barrier down the middle of the board, and positioning our final barrier so that we control $\frac{M^2}{2} + 1$ squares in the even case and $\lceil \frac{M^2}{2} \rceil$ squares in the odd case. The theory behind this strategy is that if the opposing agent opts into separating the board, that agent will be put on a timer to stop us, since if not stopped, this algorithm is the fastest at guaranteeing a win. Furthermore, if the opposing agent is following the search and destroy strategy, our algorithm should be much more difficult to entrap, as it is constantly on the move towards the next wall, seldom staying or moving around the same area.

- A summary of the advantages and disadvantages of your approach, expected failure modes, or weaknesses of your program.

  The biggest advantage of our approach is that in the majority of cases, it is able to win with the minimal amount of moves, so it is a very effective and pragmatic approach. It is also a simple algorithm that is very fast and does not use a lot of memory. The biggest disadvantage of our approach is that if the barriers we want to build for our wall are physically unreachable for our agent (meaning other barriers completely block off the access to our target barrier(s)), then our strategy does not work anymore. In this case, our agent is programmed to

make random moves, so its win rate cannot be expected to be very good. Another disadvantage is that our agent does not adapt its strategy with respect to the different environments. Its objective is always the same regardless of the position of the current barriers. This could lead to defeat in situations where our strategy would not be optimal. Another disadvantage is that we are not using classical AI game algorithms such as minimax and MCTS, so we have no way to make moves that we know would be optimal in a specific state.

- If you tried other approaches during the course of the project, summarize them briefly and discuss how they compared to your final approach.

  At first, we considered using a minimax approach. Then, we tried to compute the branching factor and the solution depth with a board size of 12 and we quickly realized that it was unfeasible since we had limited time and memory. We also considered using an MCTS approach, but using the random agent as a default policy did not seem to be a good idea since it is was playing very poorly. Thus, we needed to find a decently good default policy to be able to accurately characterize if a given move was good or not. So, we initially developed our strategy with the idea of using it as a default policy for a future MCTS, but we ended up just using it as our agent since it was already doing fairly well against the random agent.

- A brief description (max. half page) of how you would go about improving your player (e.g. by introducing other AI techniques, changing internal representation etc.)

  There are a couple of edge cases where our agent does poorly, so there is still room for improvement. The biggest improvement we could make is to find an alternative strategy for when the barriers we want to build are physically unreachable for the agent. We could maybe try to build a new wall in the zone where our agent is constrained. Another improvement could be to use minimax in board sizes of 6 and 7 since it could be computationally feasible for these boards and it's with these boards that our agent is struggling the most. We could also try to implement an MCTS with our current agent as a default policy as it was our initial plan. Another improvement we could make is to count the number of controllable squares on each side of the line because there are some cases where there are some squares that are closed off, meaning that they are unable to be controlled by our agent. This leads to situations where our agent would place himself on some side of the line expecting to be controlling the higher number

of squares, when in fact there are some squares on his side that he cannot control and this leads to defeat.