AI Overview

The Ai was designed but not implemented. The AI design consisted of two separate decision making methods. These methods are tile placement and tiger placement. Both methods were developed using an online version of Carcassonne and comparing results against the AI. This does not directly translate to TigerZone due to the changes in specifications, but was to be a guide for developing a winning strategy.

Tile Placement

The first method was to decide tile placement. The AI was to get a list of available moves, the current tile, and the number of tigers the AI still maintains.

The AI was then to add the moves' scores for each placement by finding the highest score from those available. If at any point placing a tile resulted in the completion of a lake that was owned by our AI or uncontested, this move was given priority over any other move. The second highest priority move was a move that resulted in the completion of a game trail that was owned by our AI or uncontested. Both moves result in the return of tigers and are extremely valuable.

While tigers were still available to the AI, the next most valuable move was determined by the following equations: for placing a tile to accomplish the task of "extending a lake" the value was set to be (4-x)+ 3y where x was that amount of uncompleted open edges for a lake entity and y was the count of tigers in the lake entity (for this equation and all other including tiger count, enemy tiger count was to be included as well). The more open edges on a lake, the less valuable extending the lake was. For placing a tile to accomplish the task of "extending a trail" the value was set to be (3-x) +2y where x was the amount of uncompleted trail edges and y was the count of tigers on that stretch of game trail. For placing a tile to accomplish the task of "extending a jungle", the value was set to be (2-x) +4y where x was the amount of uncompleted jungle edges and y was the count of tigers on that portion of jungle. Lastly, for placing a tile with a den on it, the AI simply looked to place it in an available move where the tile was as close to the starting tile as possible. This gives it a high likely hood that it will continue to be surrounded by more tiles. These scores were then compared for all available moves and the highest score was to be chosen, and when ties occurred precedence was given to lake, then trails, and lastly jungles.

If the AI had no remaining tigers, it would only look to complete lakes with our tigers inside, complete trails with our tigers in control, or add tiles in the Moore neighborhood of den tiles with our tiger inside of it. If given a tile with a den on it and no tigers remained, the AI would look to place the den as far away from the starting tile as possible, the opposite of before.

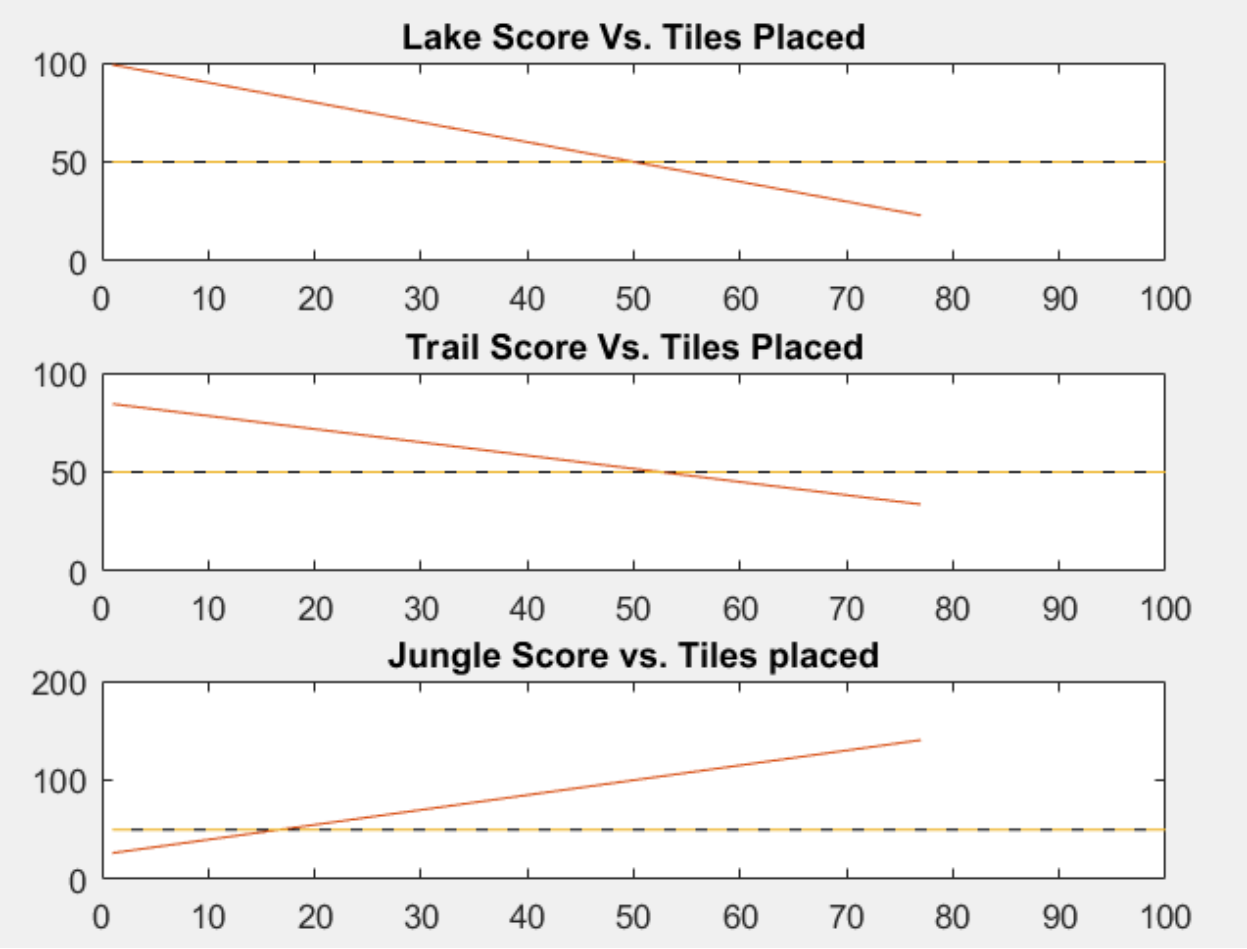
Tiger Placement

The second method was to decide tiger placement. The strategy for this portion involved placing tigers early and often while the tigers remaining was greater than 3, and to always place a tiger in a completed entity (lake or game-trail) or in any den tile when possible. The placement of tiger precedence changes throughout the course of the game and dependent upon the remaining number of tigers and the remaining number of tiles. If the number of tigers remaining was greater than 3, we were extremely likely to place a tiger. For this situation, the value associated with various tiger placements was based on the following equations: (Note: These decisions are based on available tiger placements that were given to the AI.) for placing a tiger in an uncompleted lake entity, the value was L= 100-p, where p is the number of tiles currently placed on the board (placed by both teams, not just our own.). For placing a tiger on an uncompleted game-trail entity, the value was T=85-(2/3)p. For placing a tiger on an uncompleted jungle entity, the value was J=25+(3/2)p. These values were then compared to find the highest scoring value possible. Once the highest scoring tiger placement was found, this value was compared to 50, if the value was greater than 50 then a tiger would be placed in that corresponding location. The purpose behind this strategy was to emphasize tiger placement into lake entities early on, then eventually shift to placing tigers on uncompleted trails as priority, and to have an emphasis on tiger placement in uncompleted jungles towards the end of the game.

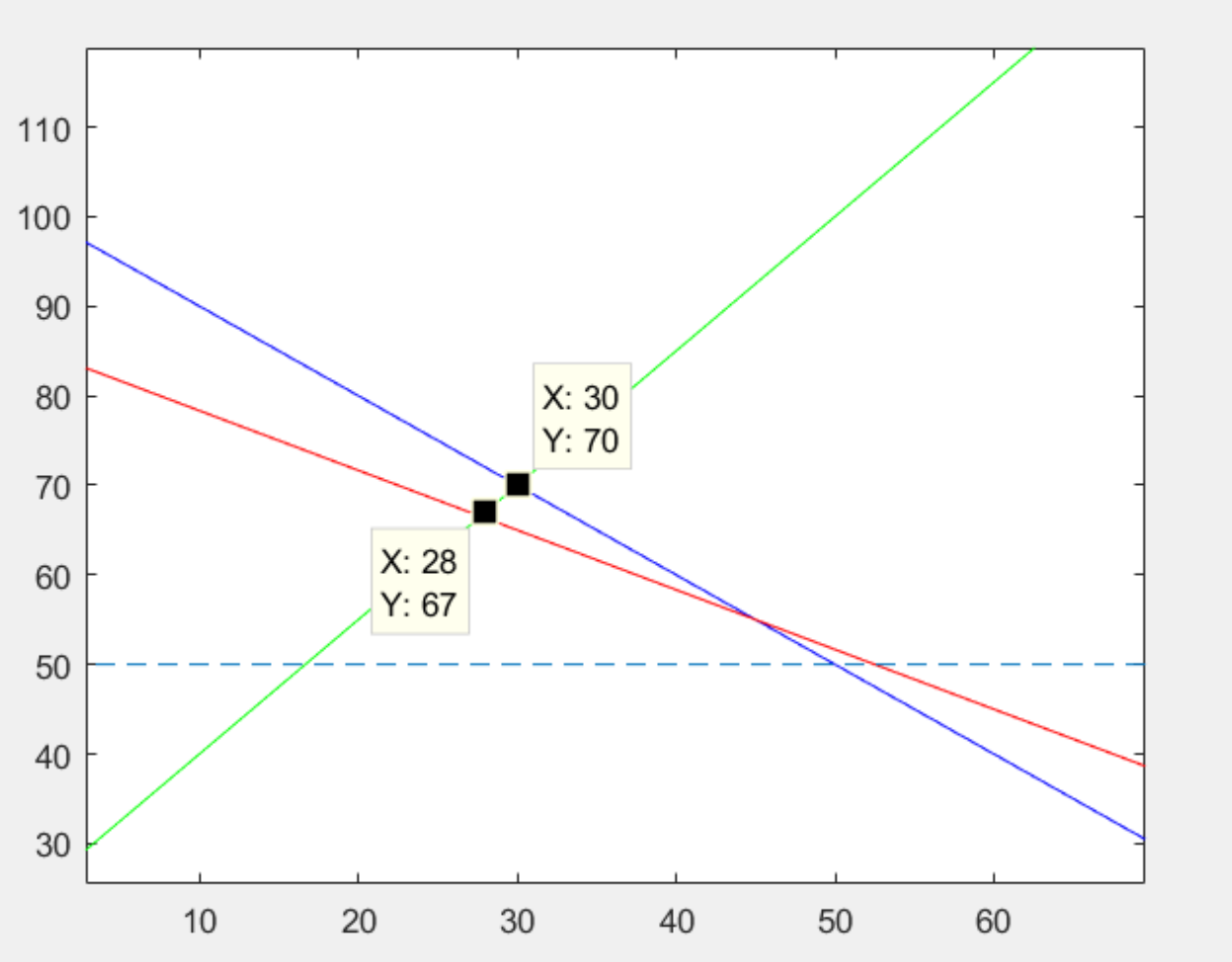
Once the tiger count got to less than or equal to 3, the value for L and T are multiplied by (t/3) and the value for J stays unchanged, where t is the remaining count of tigers for our AI. These values are then compared in the same way and scored against 50 to find the highest valued move and whether we want to place a tiger.

Once the number of tiles placed surpassed 60, the values for L, T, and J were no longer multiplied by (t/3), but instead took on their original values from the start of the program execution. The difference between scoring these values is that the highest score for L, T and J would be acted upon without comparing the value to 50. By doing this, we hope ensure that we do not end the game with tigers remaining and are still sticking with the strategy of prioritizing lakes early on, and jungle placement at the end of the game. Rarely does placing a tiger on a trail take precedence over jungle or lake placement, however we will still hope to place tigers on trails because it will not always be possible to place a tiger on a jungle or lake through a current list of available tiger placements.

The following plots show the value of each decision as the number of tiles placed changes. The first graph corresponds with placing tigers into an uncompleted lake entity. The second graph corresponds with placing tigers onto an uncompleted trail entity. The third graph corresponds with placing tigers onto an uncompleted jungle entity.



This graph is merely a merge of the three previous plots, with lake placement represented by the corresponding blue line, trail placement with the corresponding red line, and jungle placement with the corresponding green line.



This AI strategy was developed through guess and check, with the original scoring for T, J and L being calculated by hand. Once the final values were set, using this strategy for 10 games against an online Carcassonne computer, 5 games were won strongly(difference of more than 20 points), 2 games were barely won (difference of less than 20 points) and 3 games were lost.