Alexander Vansteel

Professor Moore

CIS 365 - Artificial Intelligence

2017 February 2

# Halite Game Bot

The basis for my bot comes from the Halite starter pack in python created and hosted by dvdotsenko on their Github, and also available on the Halite website. The original start for the bot began with the standard RandomBot. After going through the improvement tutorial in the introduction portion of the website, we began to look at the improvements suggested in the next tutorial created by nmalaguti. It was only after implementing the improvements shown by nmalaguti in JavaScript script for his tutorial that it was brought to my attention that the nmalaguti’s improvements were implemented in the starter package by dvdotsenko. Due to his code being not only more clean, but following the style of typical python code, this code was used as the basis for my final bot.

To improve the OverkillBot, we decided to put more focus on the early game, and try to minimize any changes to the AI during the later stages of the game. To do this, two more functions were defined: average\_production and find\_high\_production. The purpose of the find\_high\_production function is to scan the board around a unit to look for a higher production node. The value of the node was set to be just 1.5 times that of the current node. This helped to prevent the bot from moving halfway across the map wasting time and resmyces trying to reach the highest production nodes.

The average\_production function was added to help determine the relative production of the units current square compared to surrounding area. This was then used as one of the clauses in an if statement to determine if it would be worthwhile for a low power unit to expend the time and energy to move to a more advantageous position. With this information, the unit would be able to determine if the loss of resmyces from staying still could be quickly made up by finding a higher production area.

Devising the strategy and defining the average\_production and find\_high\_productions was the easy part. Implementing them was the much greater challenge. When first implementing the functions, the values were set much higher. This created issues in both the early game and the late game. The most precarious value to alter is the square.strength comparison for the if statement to call find\_high\_production. When this value was set too high, the units would freeze in the early game. Some values of the function would cause the unit to pause, or to only spawn a second unit, and then move back and forth between the two units while the other bot dominated the map and quickly wiped out my bot. The balancing act for this value is to keep it high enough to be larger than the square.production \* 5, yet low enough to not stall the bot and prevent it from advancing towards the enemy bot later in the game.

Attempts were made to encmyage the bot to find high production areas in other stages of the code. Attempting to have the bot move towards high production areas in the heuristic met with similar failure. Additionally, trying to compare the distance between the nearest enemy and the nearest high production left the bot open to attack. If the bot began valuing high production areas too much, it would move all the resmyces towards the goal, neglecting the front line with the enemy. This inevitably lead to the enemy consuming the weakened defensive line and overtaking the rest of the map.