Halite, Not for the Faint of Minerals

Stanford Cline, Ember Fairbanks, David Packer

Introduction

Elements of competition arise frequently in multi-agent systems. Scarce resources and conflicting goals create rivalries between agents. An attempt to understand the workings of a multi-agent system becomes more complicated when each agent does not have equal capability to gain utility.

We see such competition and inequality in the U.S. economy. Estimates of 2013 infer that the top 1% of the citizens in America owned 36.7% of the wealth. The other 19% of the top 20% possesed 52.2% of the wealth. The remaining 80% of the population accounted for only 11.1% of the wealth [1]. Competing to obtain wealth can be analogous to an agent pushing to gain utility in a system with limited resources. Using creative problem solving and strategy could aid an agent in competition.

An online programming challenge, Halite, plays on this very notion. This game involves a map in space with various planets. Each team has a fleet of spaceships and is trying to gain control of the map by capturing planets and eliminating enemy planets and ships.

The programmer/competitor will define a Python script that uses the game's application programming interface (API) to direct the actions of each ship in their fleet. A ship may move, attack other ships, or mine resources after docking at a non-enemy controlled planet. Planets being "mined" will produce new ships to be added to the competitors fleet. A well thought out, predefined strategy for the the fleets actions is required to be successful in this competition.

In this simulation, there are two levels of a multi-agent system: the competition between programmers to best other programmers, and the interactions of this ships in the game. A serious competitor in this competition would use a centralized solution to direct ships. But, in real life, managing a fleet of ships in space would be difficult to do centrally due to the variability, and expansiveness of intergalactic space travel.

The team will explore the viability of a multi-agent based approach by developing and testing several strategies of that type against differing central strategies. Is there a multi-agent strategy that can effectively compete with a central solution?

Previous Work

Halite is considered an “open source artificial intelligence programming challenge.” [2] Players from around the world create their own bots to compete in the annual competitions. The most recent challenge, deemed Halite II, ended in January 2018. The challenge duration lasted three months. In this time, almost 6000 participants from 110 countries entered bots. These participants range in skill level from high school students to professional programmers. The recent competition will give the team ideas and a starting point to to try new strategies.

The most effective strategies in this past competition used complex forms of artificial intelligence. Artificial Intelligence aims to allow computer systems to act in a way that may be deemed intelligent by human standards [3]. This concept is an attempt to match the innate capability in humans to perceive their ever changing surroundings and react appropriately.

To better create artificial intelligence, a study must be performed on the way humans are capable of completing complicated tasks. This knowledge can then be transitioned and implemented into machines and computer programs [4]. The team sees Halite as an opportunity to research artificial intelligence as it pertains to multi-agent systems; how can a strategy created by a human be transitioned into code?

There are other games that have similar concepts as Halite. For example, there is a mobile game where there are multiple players attempting to overrun planets and mine their resources. The winner is the player that eliminates all others by taking over enemy resources and obtaining a monopoly of the playing field. In this game, a human controls the actions of the actors in the game. The human can assess the current situation of the playing field and react accordingly. They act as a central controller similar to the submitted bot in the Halite competition.

Trying to simulate the actions and decisions a human would make in Halite could prove difficult. The team is looking to find a quantifiable multi-agent strategy that could beat other programmed strategies.

Original Work

Thousands of submissions have already been given for this challenge. Amidst such submissions include bots created by professionals working under the companies Open AI and Ubisoft. An attempt to make a bot that includes a strategy that already hasn’t been thought of may prove futile. On the other hand, the team is looking to compare the current state of the art and meta game against a multi-agent solution.

The team is also looking to use this as a learning experience to push the members understanding of artificial intelligence and multi-agent systems further.

Several strategies have been identified that will be integrated into the bots used in the actual experiment:

- Central-agent: Program central agent approaches to see how they compare with the multi-agent approaches.
- Multi-agent, go for the strongest planet: By going after the strongest other planet, it is possible to weaken it, while still staying stronger than the other bots within the playing field.
- Multi-agent, go for the weakest planet: By going for the weakest planet first, it should be easier to eliminate the ships near that planet at which point you will be better prepared to take on the stronger enemies.
- Multi-agent, fortify owned planets, then colonize: By having multiple fleets on individual planets, a bot should be able to fortify it, and mine its resources faster in order to be making more fleets. (Go until closest planet is full, move to next closest)
- Multi-agent, take over the center of the map: By trying to take over the center of the map first, a bot could potentially have a better striking position to the other planets. (Go for closest but start in middle)
- Multi-agent, take over the closest planets: By taking over the closest planets, a bot's fleet can be built up more quickly, providing greater strength to attack.
- Multi-agent, take over further planets rather than the closest: By choosing to target further planets, a bot could take expand its reach into the board earlier into the game to give a better striking position.

By programming a few different strategies and playing these strategies against each other, the team will be able to analyze the best strategies, and possibly develop better ones. Once several strategies have been implemented, the team will be able to compare their work to the strategies already submitted to the competition.

The team's original work includes how each of these strategies perform relative to all of the other bots in the playing field. A conclusion will be made on whether or not a multi-agent approach is indeed viable for this application and in turn which multi-agent approach is most efficient.

Experimental Design

In order for a user to participate in the Halite competition, the team first download all the necessary parts to create a bot. This includes a starter bot and an engine which allows for your bots to be run against each other. The starter bot, as the name implies, is only given very basic logic. The starter MyBot simply loops through all planets on the board and performs a check to see if a given planet is already being mined by another ship. If the planet is empty (not being mined), and a ship is close enough to dock, the ship will begin mining that planet. Otherwise, ships are instructed to navigate toward a new available planet.

The starter bot is not concerned with attacking other ships or performing efficient decisions and thus is unlikely to win any complex games, but it gives the user a place to begin to build his/her own bots. The team made a few changes to this starter bot and instead used the modified bot as a starting point for all the bots used in the experiment. The modified bot includes lists of the planets and ships ordered by distance from a given object and adds the functionality of attacking other ships if there are no available planets to mine.

In order for a fair tournament to be organized for all the bots involved in the experiment, the team decided hold a round robin competition where groups of four bots are pitted against each other at once. After completing the round-robin competition, the eight robots are ranked based on their performance. At this point, a single-elimination tournament is conducted to determine the winning strategy. The following is the description of all the individual bots.

Strategy 1 : Squirtle (Multi-Agent Control)
Named after the since faded popular pokemon, Squirtle, is a defensive strategy. Here, a ship or agent will look for a nearby planet that is not being fully mined. If one is available, a ship will work towards mining this planet. If no planet is available for mining, the ship will go towards the closest planet and wait ready to attack any other attacking ship.

Strategy 2 : Wartortle (Multi-Agent Control)
Just as the pokemon Wartortle is the evolution of Squirtle, so is this bot an evolution of the previous bot. Here, a ship will try to mine the closest available open planet similarly to Squirtle. If there is no open planet, a ship will go towards the second closest ship to it. This makes ships follow somewhat of a patrol pattern where they will zig-zag back and forth with in friendly territory.

Instead of waiting for enemy ships to come, a ship under Wartortle will attack an incoming ship if the enemy ships comes close enough. This potentially will create a stronger defense and prevent enemy ships from attacking the defenseless ships that are mining.

Strategy 3 : Blastoise (Central Control)
Blastoise is yet another evolution of the Wartortle/Squirtle. Blastoise follows a similar, defensive, conservative gameplay. A ship will try to mine close planets, to keep the territory of the team together. If there are no planets available to mine, this ship will stay at the closest friendly planet to defend. At turn 100 (approximately halfway through the match) all ships not mining a planet will be sent to attack the closest enemy ships. This will allow for the defensive advantages of Squirtle but then have an effective offense against the enemy team.

Strategy 4 : Strongest Planet (Multi-Agent Control)
The goal of the StrongestPlanet bot is to first build up a larger fleet and then locate the planet on the board with the greatest number of enemy ships surrounding it. If StrongestPlanet has eight or fewer ships in its fleet, each of those ships will find the closest empty planet and begin mining

that planet. Otherwise, each ship in the StrongestPlanet fleet will sort the enemy planets by the number of enemy ships either surrounding or mining them.

After finding the “strongest” planet, the ships in the fleet attack the ships surrounding the “strongest” planet with the intention of removing the enemy’s control of that planet. This process continues until all other enemy ships and planets have been defeated or the number of ships in the StrongestPlanet fleet drops to eight or below. The threshold of eight ships is a fairly arbitrary number, but one that allows for a strong enough fleet to be obtained before risking too much loss through attacking other ships. If this threshold is reached, the ships again try to find empty planets to mine.

Strategy 5 : Weakest Planet (Multi-Agent Control)
The WeakestPlanet bot is similar to the StrongestPlanet bot, but instead finds the “weakest” planet and attacks the ships surrounding that planet. The “weakest” planet is defined as the planet with the fewest enemy ships surrounding or mining it. WeakestPlanet, like StrongestPlanet, first builds up a fleet of at least eight ships before venturing into the rest of the playing field to attack. The intention of the WeakestPlanet bot is to quickly remove enemy control of the “weaker” planets so they are not being mined to create more enemy ships.

Strategy 6 : Left To Right (Central Control)
The LeftToRight bot is programmed in a central-control fashion. Rather than each ship calculating what it should do on its own, the lists of possible target planets and target ships are calculated once for each turn of a round and the ships are assigned a planet to mine or ship to attack from those lists.

The bot is named LeftToRight because the empty planets on the board are sorted by their x-coordinate. As a result, the LeftToRight bot inhabits those planets from left to right and attempts to create a monopoly starting from one side of the board. As with the StrongestPlanet and WeakestPlanet, LeftToRight builds up its fleet until it has at least eight ships. After that is accomplished, LeftToRight assigns an enemy ship to each of its own ships and sends it to attack.

Strategy 7 : Shortest Path Bot (Multi-Agent Control)
The shortest path bot had the strategy of going for the closest planet to it. It starts by assigning each bot to a different planet that is within the range of the bot, and from there it will proceed for the first couple of rounds towards that planet. Once there it will dock and begin harvesting the planet for Halite from which it can build new ships.

The strategy for the Shortest Path Bot is that it will go for the closest planet. If the planet is owned, it will go toward the ship that is docked at the planet and then dock. If the planet isn’t owned it will simply dock and begin making ships to continue on to the next closest planets. It has no regard for other ships in the path as it continues towards its destination.

This is a multi-agent approach because each ship will continue towards the closest entity to it. The decisions are not based on what can be seen on the large scale where all the information in the map is available, rather each individual would be making them, making this a multi-agent approach to the problem.

Strategy 8 : Longest Path Bot (Multi-Agent Control)
The idea behind the Longest path bot is very similar to the shortest path but with one major difference. At the beginning of the round, instead of going directly towards the closest planet to it, It will proceed to a planet that is a little further away with the idea that the chosen planet would be harder to take over in the future than it is at the beginning of the round. After reaching the planet, it will dock and begin mining for Halite there to be producing more ships. The ships will then proceed to the closest planets.

The team thought this would be a good strategy because if you were able to have a good foothold in the map early on in the game, it would increase the number of planets being mined throughout and should therefore be able to produce more of a fleet.

This is a multi-agent approach because each individual ship is making decisions based on the information available to him. He will begin by seeing planets in the distance and taking the closer planets for granted that they are going to win them. Once the first few planets have been taken over, they will proceed to simply the closest planet.

Results

Table 1 below shows the results of the pool play. The lower the number, the better the member did in the pool. The background of each cell shows which four bots played each other so, for example, in Round 1, ShortBot, LongBot, Squirtle, and Wartortle all played each other. The winner of that match was ShortBot because he had the lowest score.

Table 1. The results of the round-robin play.

	Round 1	Round 2	Round 3	Total		Rank:
ShortBot	20	23	21	64		WeakestPlanet
LongBot	27	21	18	66		Blastoise
Squirtle	24	23	30	77		ShortBot
Wartortle	29	31	31	91		LongBot
Blastoise	24	20	18	62		Squirtle
StrongestPlanet	20	36	21	77		StrongestPlanet
WeakestPlanet	21	10	27	58		Wartortle
LeftToRight	35	36	34	105		LeftToRight

The score is calculated in the following way: in each match there are ten games. In each game, each of the four competing bots can place 1-4, and that number is added to the bot's overall

score. The total score at the end of the match is the sum of its ten placements. This means the more first and second place finishes a bot earns, the lower the overall score will be at the end. The “Total” column in Table 1 shows the sum of all the scores for the pool play. From this table, the bots are ranked from best to worst and those rankings are given in the “Rank” column.

After obtaining the overall ranking, a tournament bracket is created from the ranking and given in Figure 1 below. The highest-seeded bots from the round-robin competition are placed against the lower-seeded bots. The first place seed would be grouped with the fourth, the fifth and the eighth, while the second place seed will be grouped with the third, the sixth, and the seventh. The top two winners of each side proceed to the final four. The top two bots from the final-four match square off in a one-on-one match to determine the overall winner.

The completed tournament bracket is given in Figure 1. The structure of this tournament differs slightly from most common tournaments due to four competitors facing each other in one round. It can be seen that from the leftmost group, WeakestPlanet and LongBot performed the best and therefore move on to the next round. In the second round, WeakestPlanet, LongBot, Blastoise, and Wartortle face each other. The two best bots from that round are Blastoise and Wartortle, which then face off one-on-one. Blastoise is the overall winner of the tournament.

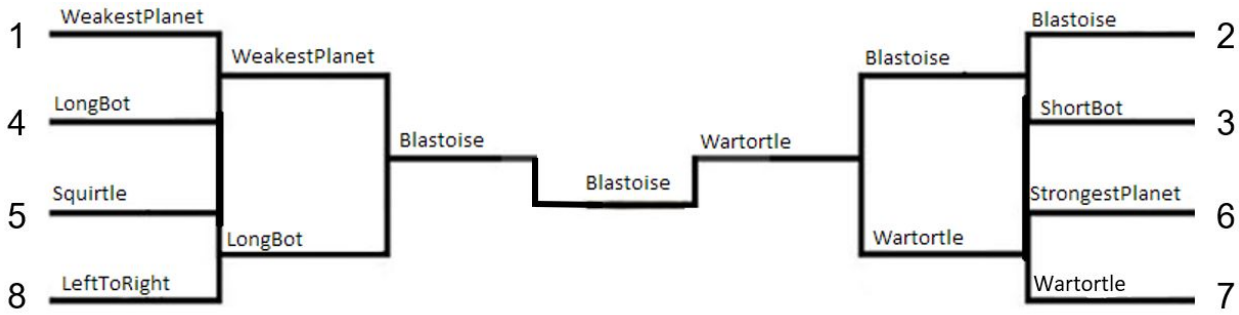

Figure 1. The tournament bracket after testing is executed.

The following paragraphs give an analysis of how each strategy performed in the tournament and why those results were obtained.

Strategy 1 : Squirtle (Multi-Agent Control)
In many cases Squirtle performed comparatively well against the other bots. But, this bot had two major downfalls. One, if squirtle didn’t claim enough planets early in the match, its lack of design to attack enemy planets would prevent it from expanding enough to overtake the other bots. Second, in several cases, Squirtle was performing well, but would time out because there were too many ships. If a bot times out, it is disqualified. Squirtle would be doing well, have a lot of ships from not using any of them for attacking, then time out.

Strategy 2 : Wartortle (Multi-Agent Control)
Wartortle performed similarly to Squirtle. Its major downfall was also timing out. In a future study, further optimizations would need to be made to prevent a disqualification caused by timing out. But, Wartortle's balance between defense and offense helped it go to the final round.

Strategy 3 : Blastoise (Central Control)
This was the winning strategy out of the eight strategies the team looked at. The videos of the competitions show the ships building up a fleet and then using the built up ships to attack the enemy ships in force. Because the ships are attacking together in force, the other strategies were unable to effectively handle the larger attack.

Strategy 4 : Strongest Planet (Multi-Agent Control)
The StrongestPlanet bot, when not building its fleet, attacks ships based on the planet with the greatest number of ships surrounding it. Ships in the StrongestPlanet fleet attack enemy ships around that planet. This may not be an efficient strategy as the target planet could be on the opposite side of the board from where the StrongestPlanet ships start. As they travel across the board, they are more vulnerable to attack from enemy ships. The ships can spend a lot of time traveling, in which time those ships are not helping to increase fleet size or decrease the fleet size of enemy bots.

After obtaining a fleet of at least the threshold size, StrongestPlanet does not attempt to further colonize any new planets. Instead, its sole focus is attacking enemy ships until the fleet size falls below the threshold. If the fleet size does fall below the threshold, StrongestPlanet may not have enough time or empty planets to again build up the fleet. All the reasons listed above would have contributed to StrongestPlanet not doing well.

Strategy 5 : Weakest Planet (Multi-Agent Control)
Because WeakestPlanet is based on the same concepts as StrongestPlanet, WeakestPlanet also has the same performance issues. As with StrongestPlanet, the target planet could be on the opposite side of the board so travel time between the planets can be detrimental. Though WeakestPlanet had the same issues as StrongestPlanet based on the structure of the code, WeakestPlanet was the winner of the round-robin competition. This could be attributed to the strategy of eliminating control of weaker planets being more effective because the WeakestPlanet bot can then move through enemy planets more quickly.

Strategy 6 : Left To Right (Central Control)
The LeftToRight bot performed the worst in the team's testing. LeftToRight doesn't have a sophisticated way of deciding which ships to attack, it simply assigns enemy ships to team ships from a list of ships on the board. The list of enemy ships is not sorted by which are closest to team ships as the lists for StrongestPlanet and WeakestPlanet are. The team ships could be

assigned to any enemy ship on the board and are then again presented with the same issue as StrongestPlanet and WeakestPlanet above regarding across-board travel.

Strategy 7 : Shortest Path (Multi-Agent Control)
Shortest Path Bot was one of the first bots the team built. Because of this, it is fairly similar to the MyBot file that was downloaded with the original game engine. The team predicted that this bot was going to perform better than it did, because of the straightforward nature of the bot.

Although it seemed logical for each ship to go after the closest object, the bot had a flaw. The strategy did not allow for ships to amass at all. The strategy sends all of its fleets out in individual bunches which are easily overwhelmed by the other teams. One improvement that could have been made was to let some of the other computers fight things out while it fortified its portion of the map, and gradually move onto the remaining planets.

Strategy 8 : Longest Path (Multi-Agent Control)
Initially, while the team was determining the different strategies of bots that were going to be tested, the team predicted that the long path would beat some of the other strategies. One flaw is the travel time. If ships didn't start by going to the closest planets, it would take crucial time to travel to the distant planets and then establish a base. By the time the LongBot made it to the planet, other strategies had already mined enough mineral to make a few new fleets. While it did put the bot at a more offensive position, the first few turns are too critical to the success of the bot that the advantages didn't outweigh the disadvantages.

MultiAgent vs. Central
In the end, it was a central approach the won the competition. But, strategies like Wartortle, where each ship had limited vision, performed well against the central control agents. This goes to show that a well thought out multi-agent strategy can perform well up against central control systems.

A dynamic strategy, like Blastoise, where tactics change throughout the match has shown its usefulness. Having an agent in a multi-agent system that can adapt to its environment and the situation can be more effective than an agent that only follows a list of predefined rules.

Conclusion
While playing the bots against each the team found that there were a few crucial factors that led to success. The first ten turns of the game are essential, and one can often predict the winner this early in the game. Fleets in the game have a near exponential growth because created ships can move on to create more. Watching these games shows a colored virus taking over the screen. At the beginning of the round, frequently, two ships would crash into each other going for the same planet. When this happened, the team with the crashed ships was immediately at a disadvantage compared to the other bots.

Another important factor in the success of the individual bots was the placement of the planets. The multi-agent bots tended to do better when the planets were closer packed especially through the center, while the central bots would have an advantage when the planets were more spread out and split into distinct quadrants.

Future Work

To obtain more credibility, the team would need to produce a state of the art strategy that competes effectively against a world class submission. An analysis would need to be conducted on that strategy to identify the strategies effectiveness under a multi-agent paradigm, where each spaceship acted “independently” without omnipotent knowledge.

The team would need to further research how the lessons learned from this competition could be applied to real world applications. Could this knowledge be applied to networking algorithms where each “planet” is considered a node in a network? Could the interaction between the space ships be applied to creating autonomous vehicles, mining, or interactions between machines in a factory?

Competition between programmers in Halite has pushed creativity and ingenuity to greater heights. Can other problems in computer science be solved by turning the problem into a competitive game? Taking a research topic in multi agent systems and making a game of this problem may be a cheaper way in academia to push the state of the art further. The team could make such a game and investigate this strategy as an alternative form of research.

References

[1] G. William Domhoff (April, 2017) "Wealth, Income, and Power" retreived from: https://www2.ucsc.edu/whorulesamerica/power/wealth.html
[2] Two Sigma. (2017). Halite AI Programming Challenge. Retrieved March 02, 2018, from https://halite.io/
[3] Berkeley, I. S., Ph.D. (1997). What is Artificial Intelligence? Retrieved March 02, 2018, from http://www.ucs.louisiana.edu/~isb9112/dept/phil341/wisai/WhatisAI.html
[4] Artificial Intelligence. (2018). Retrieved March 02, 2018, from https://cpsc.yale.edu/research/artificial-intelligence