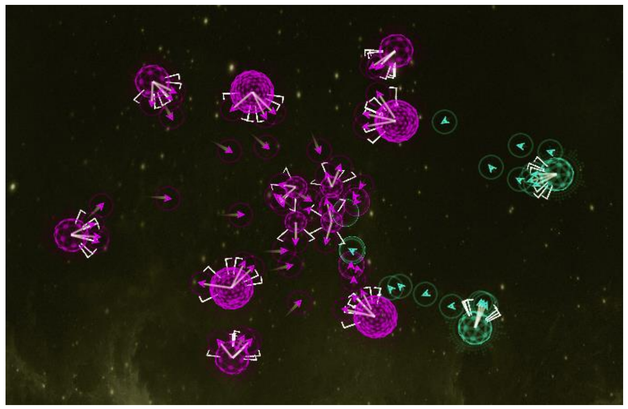
Halite II Game Challenge

Muxi Xu
muxi@Stanford.edu

Introduction

- Halite II is a competition held by Two Sigma in 2017~2018, which attracted around 6k players/programmers. This game can be viewed as a **simultaneous turn-based strategy game**. Each player's bot starts with 3 space ships and try to defeat all other players in the map.
- The game map is set as continuous Cartesian plane. Below chart shows a typical frame in a game between two bots.



- While the details rules of this game are not short, followings are some key factors we need to consider:
  - **Ship**: this is the only object controlled by players. Ships have a health of 255 and can thrust or dock/undock an planet.
  - **Planet**: This is the main resource on the map. Planet can produce new ships if being docked. The planets have different size (radius) which determine its health and docking spots.
  - **Attack**: When ships owned by different players are within the attacking range (5 units), they will attack each other with 64 damages per turn.
  - **Collision:** if a ship runs into an other friendly ship or planet, it will be destroyed simultaneously.
  - **Turn**: At beginning of each turn, all bots will receive the current game update, and have 2000 milliseconds to issue the commands.
- **Project Goal**: This competition has already concluded in early 2018. A few top players have open sourced their bots after this competition. While the online competition is no longer available any more, my goal is to set up a local competition environment and build bots to achieve as higher ranking as possible.

MDP Based Bot

For this game, the bot need generates for each ship based on available information while the reward is depending on the actions taken. This is very suitable for MDP.

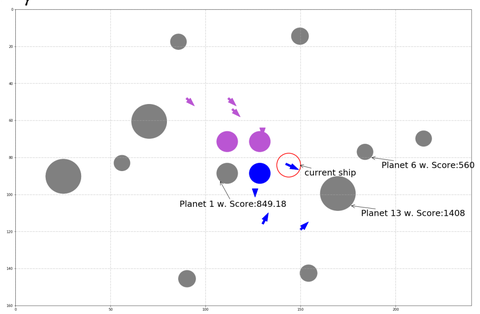
- **Two Phase Strategy**: Before design MDP models, I first split the action generating to two phases:
  - **Target Assignment**: this is to select best target for each ship based on ship's location, planets' status and enemy ship/planets' status. I use a MDP type of design for this phase.
  - **Navigation**: Once a target is determined, we need to find the best route to it. Currently, my bot uses greedy search based method with anti-collision logic.
- **MDP Model Design**:
  - **State**: $[Map\ Status\ (planet, ship\ info, etc.)\ for\ current\ turn]$
  - **Action**: $[assigment_{ship\ i}\ for\ all\ ships]$
  - $\boldsymbol{IsEnd(S)}$: health =0 or all enemy health =0 or round =255 (Game Over)
  - $\boldsymbol{Reward(s, a, s')}$: the potential gain from different actions like $\Delta expected\ health$
  - $\boldsymbol{t(s, a, s')}$: This is defined as the probability of future states based on current actions.

- **Bot 1 - Occupy First Bot**: The first example Bot presented here is 'occupying first Bot' or a "settler" which prioritize docking planes. The core logic of this bot is following formula:

$$Reward_{ship\ i, planet\ j} = \frac{21.25}{1-\gamma} * \gamma^{\frac{distance}{7}} * radius$$

This formula takes three key factors into account:
a): expected gain ($\frac{21.25}{1-\gamma}$, future gain sum)
b): distance discounting ($\gamma^{\frac{distance}{7}}$)
c): planet size factor (radius)

A sample score result is as the left plot, for the target ship (red circle), planet 13 get highest score.

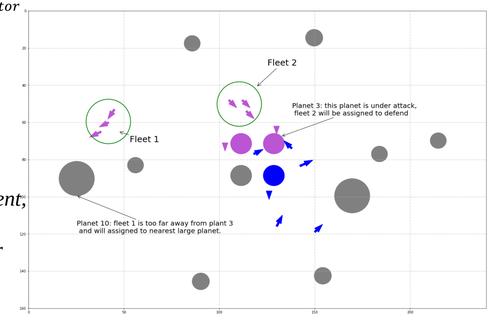


- **Bot 2 - Balanced Strategy Bot** : I then enhance the above bot, new reward functions are:

Enemy Planet: $\frac{1.08^{docking\ space}}{3distance} * \frac{central\ factor}{enemy\ strength\ factor}$

Friendly Planet: $\frac{1.08^{docking\ space}}{distance} * \frac{central\ factor}{defense\ factor}$

This formula has couple key enhancements:
*a): **Fleet Based Score**: calculate one score based on fleets (a group of nearby ships) to increase efficiency*
*b):**Defense logic**: including a defense factor to reflect the urgency of defending. The more urgent, the smaller the factor, hence higher reward.*
*c):**Integrated Attack logic**: assign a reward for attacking as well. The Bot will attack early if there is a valuable target nearby.*



A sample strategy outcome is the plot above. As we could see, the purple player's ship could be divided to two fleets. While fleet 2 is assigned to defend planet 3, fleet 1 is assign to occupy planet 10.

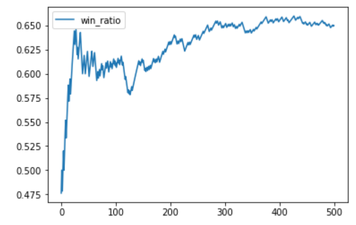
Machine Learning Based Bot

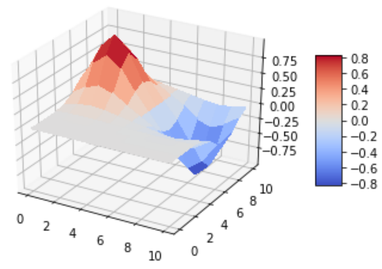
- **Neural Network Based Bot**: This type of bots is aiming to mimic top players' strategy by learning from a large amount of game replays.
  - **Data Processing:** the replay file doesn't include ship movement orders. A key step is to "guess" each ship's target based on the object locations. Post data processing, our goal is to get the ship to planet assignment, which is our target, as well as other features.
  - **Features**: features includes player number, current ship's location, planet health, etc.
  - **Model Structure**: my current model uses two layers with nodes varies from 8~32.
  - **Execution**: the model outcome includes a probability for each (ship, planet) assignment. Our bot will use the one with highest probability.
- **Reinforcement Learning Based Bot**: this is policy gradient type algorithm.
  - **Parameterized Policy**: to use policy gradient method (REINFORCE), we need to have a parameterized policy. From a high level, the design is:
    - **Assign Score to all Locations**: assign positive score if near friendly objects, negative other wise. Then move towards negative points.

$$\begin{cases} Score((x,y)) = \sum_{all\ ships, planets} Friendly\ Weight * RBF(distance, theta) \\ RBF(distance) = \exp(-(theta * Distance)^2) \\ Friendly\ flag = -1\ for\ enemey\ 1\ otherwise \end{cases}$$

A sample score is the lower right chart. This chart contains one friendly (2,8) and enemy object (8,8).

    - **Choose the directions that minimizing the score**: this is done by calculating the gradient function regarding x and y.
  - **Model Training**: uses REFINROCE logic $\theta \leftarrow \theta + \alpha\gamma^t \nabla \Sigma \ln(\pi)$ with wining ratio as target. Left plot shows one round of training.





Competition Result

- **Bot Evaluation:** The strength of bots is evaluated by the TrueSkill system which is being used by Microsoft on Xbox Live.
- **Competitors:** In addition to my bots, I have included 8 bots from those top players. All of these 8 are ranking in the top 100.
- **Result:** Unfortunately, my bot is not able to defeat those top bots. Currently, the MDP Bot 2 is the best one. It ranks #9 with a skill score 15.4 compared with 17.6 of #8.
  - Among my other Bots, the neural network bot performs only slightly better than the baseline. This may due to that my computer is not able to training model fast enough.
  - The reinforcement learning bot learns pretty fast which achieves a skills score 14.3 of within couple hours' training.

Introduction Video Link: https://youtu.be/fDdK8KKIlxQ

Thanks for reading!