Halite

Tom Rochette <tom.rochette@coreteks.org>

July 24, 2025 — daae079c

0.1 Context

0.2 Learned in this study

0.3 Things to explore

- Based on the different heuristics defined, how can we design a program that will attempt to exploit these heuristics and find the best balance between each of them?
  - Is it possible to design a program such that we write a bunch of heuristics as procedures and let it optimize itself?

1 Overview

2 Game format

2.1 Input

- Player identifier
- Map width and height

3 Metrics

- Territory: Sum of pieces owned by the player
- Production: Territory * Production of owned territory pieces
- Strength: Sum of the strength of individual pieces

4 Approaches

4.1 Naive

- Read the rules and understand them
  - Discover limits (e.g., strength is capped to 255)
- Think of various strategies
- Implement said strategies
- Iterate
- Easy to get something working
- Most likely will find an acceptable solution, but not an extremely good one
- Cannot improve with “training”

- Most likely going to end up writing conditional spaghetti that is difficult to improve/optimize

4.1.1 Analysis

- Try to get high production pieces as soon as possible
- Try to increase territory as soon as possible
  - Attack the pieces with smallest strength first
- Avoid hitting the strength cap of a piece (otherwise that space is wasted)
  - Always attempt to move the strongest pieces to the borders of the territory
- Prefer fighting against unowned pieces than owned pieces
  - Win territory as far away as opponents as possible in order to increase the amount of effort required on their part to win the territory
  - This prevents the "waste" of cycles between you owning the piece and your opponent owning the piece
- A piece with strength = 0 should stay STILL at least 1 frame
- Do not attempt to attack unowned pieces until we are at least as big as them
  - This is probably not true (since it makes no difference)

4.2 Machine learning

- Determine available inputs
- Determine requested outputs
- Game state can be computed based on the input
- Requires the creation of metrics and their programming
  - Chain length
  - Proximity to high-producing pieces
- Still a lot of manual/intellectual work must be done in order to determine the best course of action
- Is more likely to find a better optimal (if the programmer is experienced)
- Can potentially improve with training (playing more games)

4.3 Neural network

- Feature input vector
  - Strength
  - Each 4 adjacent pieces strength
  - Each 4 adjacent pieces owner =?= self

5 Learning from examples

- How can the agent learn the rules and constraints of the system?
  - If the agent does nothing, but is able to collect data (input), then it is possible to determine the range of valid values for certain metrics (for example the map width and height, the minimum/maximum strength value of a piece, the minimum/maximum production of a piece)
  - Through randomization of its action, the agent should soon realize that it is worthwhile to do something vs not doing anything at all

6 Ideas

- Build a 2x2 matrix that represents where the agents should flow toward. In a metaphorical sense, imagine the agents being water, and the map being a terrain. Based on the formula used to compute the terrain, the agents simply "trickle" down the path of least resistance.

7 Testing

- Test against various variants of the agent
  - Run against the same agents multiple time in order to establish stable metrics
- Train and tweak parameters

8 Understanding how to improve

- Relax the constraints
- Change the constraints
  - With the opponents removed, how would you collect the most territory as fast as possible?
- Optimize for strength, production, territory (pick one)
- With only one piece moving per turn, how would you optimize strength, production, territory (pick one)?

9 Heuristics

- Move the biggest producers as little as possible (as moving incurs a frame of production penalty)
- Attack the highest production sites as soon as possible
- At equal strength, prefer attacking a site with more production
- At equal production, prefer attacking a site with less strength
- At equal production/strength ratio, TBD?
- It is worth attacking a site if attacking such site makes it possible to reach a given total strength faster than by not attacking
  - For instance, considering you have 2 sites which you could attack and which have different strength/production values, there's an order in which it makes sense to attack each site in order to minimize the amount of frames required to acquire both
- When approaching the end of the game, trade strength for territory (as the winner is based on final territory size)
- Sometimes it's best to wait for an opponent to "waste"/use his strength on a valuable piece of territory to then acquire it at a lower strength cost

10 See also

11 References

- https://halite.io/index.php