A(I)nt Colony

Elisheva Seeman, Ethan Glick, Shaimaa Maranah

Problem Description:

Using a simulation of an ant colony, where the movement policy controls the how the ants explore their surroundings in search for food, we are trying to maximize the food collection for the winter. We have also explored queen policy a little – deciding how many ants to have each day.

This is a problem which Evolution has optimized throughout millennium, and we are curious to see how our results compare to the natural behavior of real ants.

Module Build:

In order to get as close to real life results as possible, we modeled our behavior after real ants (specifically Seed Harvesting ants). In order to achieve this, we divided the world into a grid (with the anthill in the center) and created a system for distributing food throughout it. We then allowed the ants to move freely throughout the grid, based on their movement policy.

Between each day additional food is dispersed in different quantities throughout the grid (based on a daily food distribution function) This reflects the natural randomness of food 'appearing' in the vicinity of an anthill.

In our module the year starts in the end of winter. Therefore, the initial distribution of food is less generous than the daily food distribution. We used three variants of the food distributions: a low, medium, and high amount of food dispersed each day. In the results of analysis they are referred to as LD, MD, HD for low, medium, high distributions respectfully.

The grid also holds scents: as ants explore the world, they leave different types of scents out there. In our simulation we used two: "food in this direction" and "I'm exploring this way"

The ants search for food one day at a time. Given that we are trying to optimize the ants' movement policy, we provide a reward based on the amount of food collected. However, since there is also a cost in maintaining the anthill (feeding ants, creating new ants to replace old/dying ones, etc.), the reward must consider how much food is consumed each day as a result of these costs:

$$reward = foodcollected - foodconsumed$$

We then have a game runner object, which receives as parameters all the policy functions (movement, initial food and food change, queen policy, etc.) and runs the simulation for a given number of years.

Queen Policy

In an attempt to achieve further realism, we decided to add an option for a 'Queen Policy'. This policy, like the queen in an anthill, will increase (or decrease) the number of ants in the colony each day, as it sees fit. This creates a second layer of optimization for our simulation: what is the optimal number of ants for a given ant movement policy. This is done using a q-learning algorithm which runs in tandem with an ant movement q-learning algorithm (more details below).

Ant Factors

The build of the grid, number of steps per day, and other parameters, were designed in order to mimic ant behavior:

The maximum distance an ant will travel from the ant hill in search of food is ~100m. In a day they can travel up to ~2000m. The average smelling distance is ~2m. Based on this information we have configured our anthill simulation as follows:
Each square in the grid represents the distance in which an ant can smell food from its center (meaning each square will be 4m). We assume that once an ant smells food they will go to it. We allow an ant to travel 1 square per step. Therefore, the maximum number of steps an ant can do is:

$$2000/4 = 500$$

Thus, each day in our simulation consists of 500 steps. Due to limitation in computer processing power, in our simulations we will have 10 days per year (as opposed to 365!). This ultimately has no effect on their behaviors, as we have set each policy to run for a different number of years based on their individual optimization time requirements.

An ant colony typically has 50,000-500,000 ants. With the computer power at our disposal, we cannot possible track that order of magnitude of ants. Therefore, each ant in our simulation will reflect ~500 ants in real life. We have therefore decided to have 300 ants in our ant colony. And when a 'Queen Policy' is involved, we permit the number of ants to range from 100 to 1,000 (due to the daily cost of living for each ant, the queen policy will reach an equilibrium somewhere in this range).
(Note that this will cause each unit of food in our module to reflect multiple units in reality).

In real life, when an ant finds food, they immediately return back to the anthill, leaving a scent trail in their wake directing other ants towards the food (if an ant should happen to be within smelling range of this scent trail). Ants also leave a general scent trail wherever they go, letting other ants know that they have gone searching for food in that direction. These scents decay over time and ultimately become unnoticeable. Therefore, we have included all these scent features in our simulation and use them to improve our movement policies.
Furthermore, since, for all intents and proposed, ants are essentially clones of one-another, the ants will all be using the same policy on any given day. This is extremely useful as, given there are 300 ants and 500 steps per day, this provides $300 \times 500 = 150{,}000$ units of data per day. You can see from this why having 10 days per year is more than sufficient.

Solution Methods And Process:

Non–Learning Movement Policies

Random Algorithm:

Randomly chooses a direction to move in. This provides poor results but is good for establishing a lower threshold for food collection.

Smart Algorithm: Evolution Mimic

A basic mock for the way ants explore the world: if an ant smells a “food in this direction” scent - follow that scent. If it smells an 'explored in this direction' scent, explore in a different direction. This is very similar to real ant behavior, and ultimately yields very good results.
In our results and conclusions this algorithm is referred to as “scent”

Learning Movement Policies

Genetic Algorithm:

We create 'individuals', representing a different movement policy. Each movement policy contains a different weight to be given to different parameters. In each episode, each individual runs for a year. The top two performers are used as 'parents' for the next generation - with each child inheriting a mix of the parents weights. There is also a certain probability of mutation, in which a child individual will be created with random weights. While descent results are often achieved within just a few episodes (significantly better than the Random algorithm), it can takes a very long time to achieve results that can compare to the Smart algorithm (though this is to be expected given the nature of genetic algorithms).

As the rewards of each "individual" can only reliably be measured in years in each episode, each individual needs to run for at least a year. This makes the genetic algorithm relatively slow.

Q-Learning:

We used two different types of q-learning (one for the queen policy, one for the movement policy). Both were based off the code we had from ex4.

Feature–based q-learning:

Since the ants exist, together, in a large, ever-changing grid, using state-based q-learning would not be an efficient or effective approach. We therefore decided to use feature-based q-learning, as all the ants' knowledge of the world comes from what they can sense at any given time. After much trial and error, we found that the features relevant to the ants were the scent trails and whether we were moving away from the anthill or towards it. Thus, the features are built around these parameters, with the reward being the amount of food the ants found using this configuration. Due to the nature of feature-based q-learning, and the high number of data units per day (as mentioned in the 'Ant Factors' section), we are able to see impressive results after only a few years (episodes).

State–based q-learning:

For the queen's policy we had fewer variables, and therefore it was easier to use states as opposed to features. We set the legal actions to be increasing or decreasing the number of ants (by constant margin) before the start of each day. The states are determined based on the number of ants that were in the colony on the previous day. The reward is the amount of food collected that coming day, minus the food consumed. The queen policy runs together with the movement q-learning policy to reach an optimal combination of ant population size and ant food-finding ability. There are two ways the two policies can be run together: Simultaneously or Independently.

Simultaneous vs Independent Learning

As we are trying to maximize two different policies, we tested the efficiency of simultaneous learning (e.g. running both learning agents at once) vs independent learning - where the policies alternate which one is learning (one year on, one year off), so at each increment the policies learn independently and under slightly different conditions. The results and comparison of these two approaches are included in our analysis.

Results and Conclusions

In order to compare the different algorithms, we ran each version of the problem with different distributions of food on the grid: high, medium, and low. We also used a varying number of ants: 300, 400, 500. We compare the total reward of a single year for each policy type (after having given the q-learning and genetic algorithms several years to train/evolve), along with the compounding food. We also ran the q-learning movement policy together with the q-learning queen policy and compared the results between the two approach options (simultaneous vs independent).

Our conclusions are as follows:

Genetic algorithm:

The genetic algorithm carries a random component in it: the mutations, along with the initial choice of the population. What we noticed was in some cases a larger variance in the initial years, but by the third episode, most of the time, the genetic algorithm lands on a policy it likes, and from that point on there is very little change (as demonstrated in figure 1 - 300 ants).

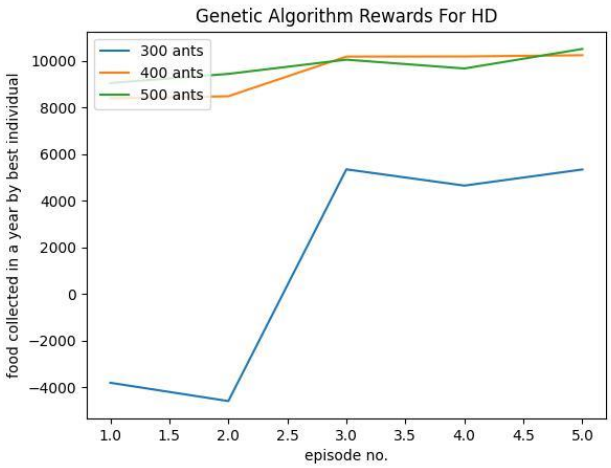


Figure 1: Genetic Algorithm over multiple episodes with a high food density

Q-learning – Movement:

The q-learning algorithm learned relatively quickly: unlike our other algorithms, it learnt from every interaction an ant had with food, as opposed to every day or year. Therefore, relatively quickly we were getting results that were consistent with the best results for later years. This made it hard to distinguish between the first year and the rest, within the margin of error created by the random nature of the grid, of course.

Independent Vs Simultaneous learning:

We ran both the independent and the simultaneous learning for 7 learning episodes. As seen in Figure 2, the simultaneous did significantly better in a lower food density, and the two were similar enough in the medium and high food densities to be within the margin of error. Furthermore, the simultaneous learning was much faster, as a simultaneous learning episode is only one year.

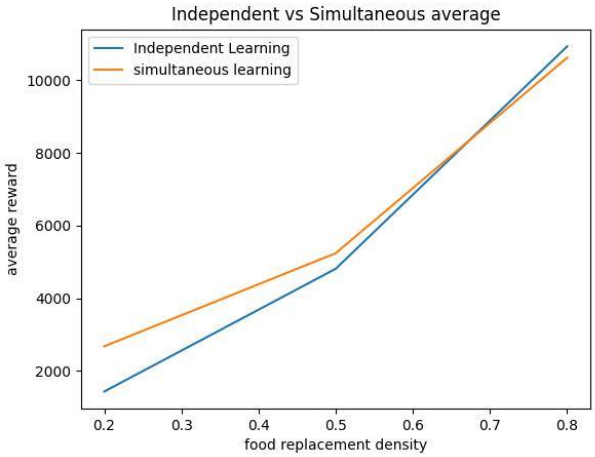


Figure 2 The average reward for the last three learning episodes

Comparing Combined Learning Strategies to Movement Only Strategies:

We Compared the results of our combined (independent, simultaneous) learning algorithms to the results we got from optimizing only a movement policy. For that we ran all the different movement policies on 300, 400, and 500 ants and chose the best result for each density and algorithm. We thought this would be a good predictor of the potential of the combined learning. The results are in figure 3. As you can see, the simultaneous learning did mostly better than the q-learning (in medium density it did slightly worse, but that is within the expected margin due to the grid's random behavior). It did mostly worse than the genetic algorithm. But, as it is based on our q-learning algorithm, this result is expected.

We didn't expect there to be any major change: in most of the data we collected, the ants got higher rewards with 400 ants than with 500. Therefore, we are already within the range of the maximum ant number for maximizing the reward.

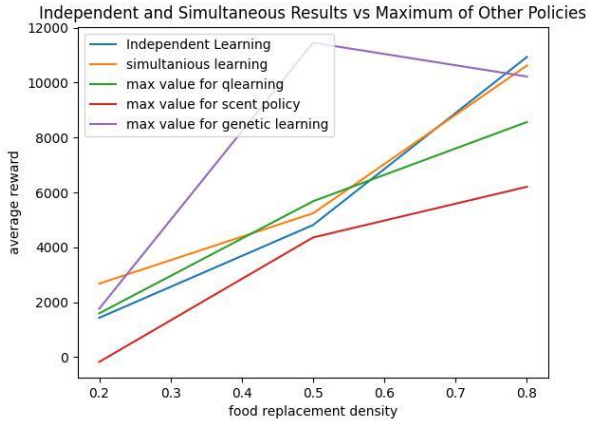


Figure 3: comparing the movement algorithms to the combined ones

General Comments:

The food does not replenish itself completely every day, and therefore different algorithms will be more efficient in different days of the season. Naturally, different food distribution and change functions will affect these inconsistencies, but regardless – any algorithm we choose will be limited by this.

Conclusions :

In our case, simultaneous learning was more efficient and more successful than independent learning. We do believe this will vary given different parameters, but it does answer a question that we were pondering throughout: what is the most effective way to learn multiple policies at once. Something else we noticed is how fast do different functions take to reach optimality: in our example, the q-learning learned much faster, but sometimes the genetic learning agent got significantly better results. However, the q-learning algorithm is faster by multiple orders of magnitude. This is a relevant conclusion for anyone attempting learning in a real time scenario: the attempt to balance successful results along with time limitations, and the knowledge of how to get faster results with a solution that is good enough.

Furthermore, our last conclusion is that even rudimentary AI, such as those we used, are limited by the ability of mankind: we had to change around the parameters and policies to better encourage learning behavior. We suspect this is the core reason that the q-learning didn't do as well as the genetic algorithm.

We have also found that our learning algorithms did better than our scent – based algorithm, which was made in order to mimic evolution and with our best guess as to how an ant can reach an optimal result.

Annex 1 – Raw Data Collected:

Policy learned _ ant number _ change function density = [reward each year]

```
#  --------------------------- random, scent ---------------------
random_300_MD = [-1381, -940, -1133, 1044, -1684]
random_400_MD = [-2558, -2732, -2241]
random_500_MD = [-4177, -4002, -3818]

scent_300_MD = [3347, 4174, 3874, 4118, 4146]
scent_400_MD = [3537, 3772, 3726, 3249, 5470]
scent_500_MD = [1515, 2228, 2307, 2100, 2293]

random_300_HD = [-577, 106, 292]
random_400_HD = [-1294, -1212, -606]
random_500_HD = [-1706, -2309, -2654]

scent_300_HD = [6509, 6085, 6720, 5710, 5447]
scent_400_HD = [4800, 7229, 5898, 5265, 5714]
scent_500_HD = [5148, 4827, 5612, 6211, 6200]

random_300_LD = [-3081, 3043, -3454]
random_400_LD = [-4674, -4320, -4610]
random_500_LD = [-6347, -6390, -5586]

scent_300_LD = [-120, 258, 1006, -389, -628]
scent_400_LD = [-1183, -585, -579, -125, -224]
scent_500_LD = [-1862, -1724, -1602, -729, -2120]


#  --------------------------- qlearning ---------------------

qlearning_300_LD = [1707, 1101, 1985, 1480, 2022, 1282]
qlearning_400_LD = [1639, 1857, 1803, 1356, 151, 435]
qlearning_500_LD = [375, 672, -336, 567, 775, 25]


qlearning_300_MD = [5680, 4364, 5241, 5702, 4875, 4950]
qlearning_400_MD = [5733, 5709, 4437, 6010, 4863, 5440]
qlearning_500_MD = [5712, 5072, 4708, 5396, 5624, 6017]


qlearning_300_HD = [7897, 7785, 7594, 8605, 8056, 6909]
qlearning_400_HD = [7837, 7841, 8326, 8206, 9395, 8090]
qlearning_500_HD = [9624, 8486, 8378, 8718, 8425, 8515]


#  --------------------------- combined qlearning ---------------------

indipentent_LD = [1445, 2562, 2976, 772, -806, 2721, 3041]
simultanious_LD = [3535, 3685, 2980, 2198, 2146, 3702, 2661]

indipentent_MD = [6207, 6321, 6010, 4367, 3972, 5381, 5528]
simultanious_MD = [5848, 5964, 7082, 4605, 6032, 6341, 3984]

indipentent_HD = [9173, 10455, 9591, 11014, 10207, 11111, 11411]
simultanious_HD = [9450, 10419, 10331, 9481, 11059, 11242, 10715]
```

```
#  -------------------------- genetic learning ---------------------

genetic_300_LD = [1497, 2384, 1992, 1599, 1929]
genetic_400_LD = [-6687, -6687, 619, 1713, 1259]
genetic_500_LD = [-8540, -672, -14, -627, 2941]

genetic_300_MD = [5750, 5827, 5570, 5194, 5905]
genetic_400_MD = [2464, 3521, 2629, 2800, 3079]
genetic_500_MD = [-7548, 10800, 10800, 11111, 11802]

genetic_300_HD = [-3812, -4597, 5350, 4651, 5344]
genetic_400_HD = [8403, 8487, 10190, 10195, 10249]
genetic_500_HD = [9054, 9447, 10058, 9679, 10517]
```

Annex 2 - Sources For Ant Behavior:

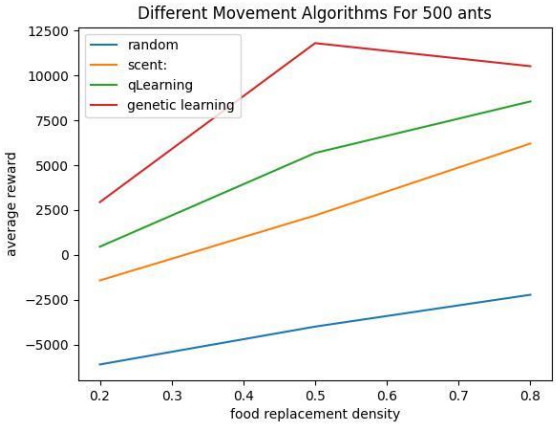
The Ant Realm – Rose E. Hutchins (1967)

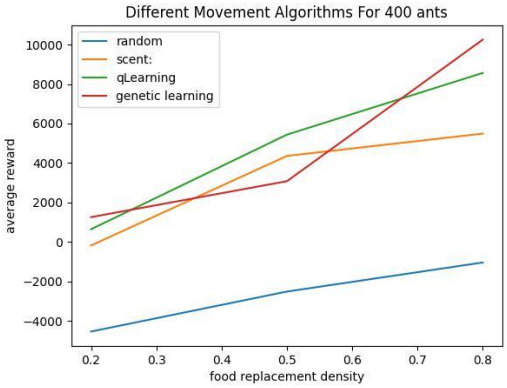
https://misfitanimals.com/ants/

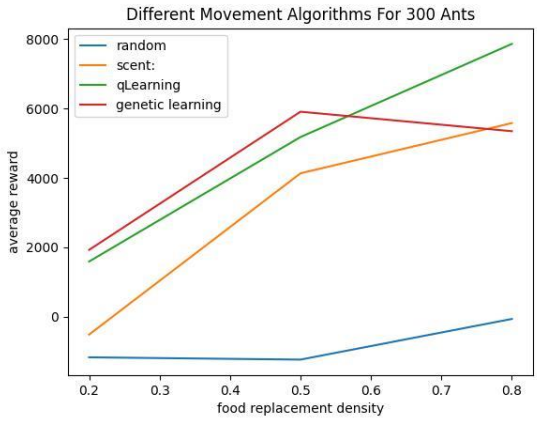
https://en.wikipedia.org/wiki/Ant

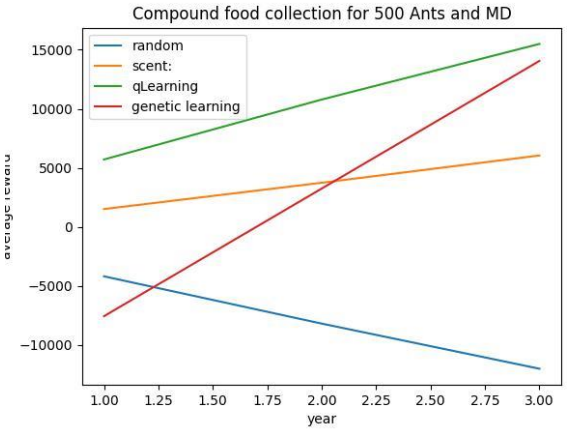
Annex 3 – All Graphs

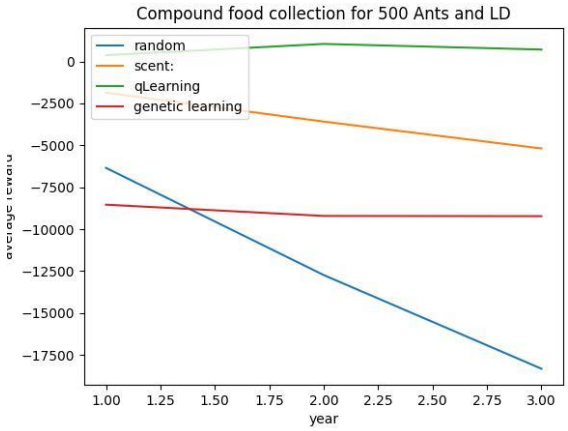
In the next pages you will find all the graphs we generated with our raw data:

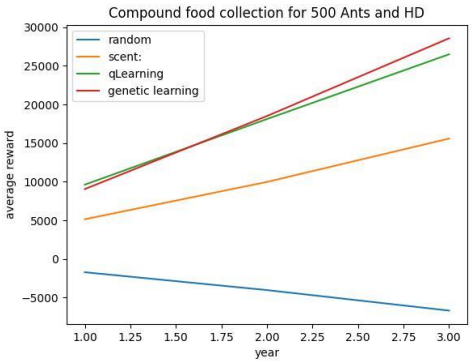

Different Movement Algorithms For 500 ants
random
scent:
qLearning
genetic learning
average reward
food replacement density

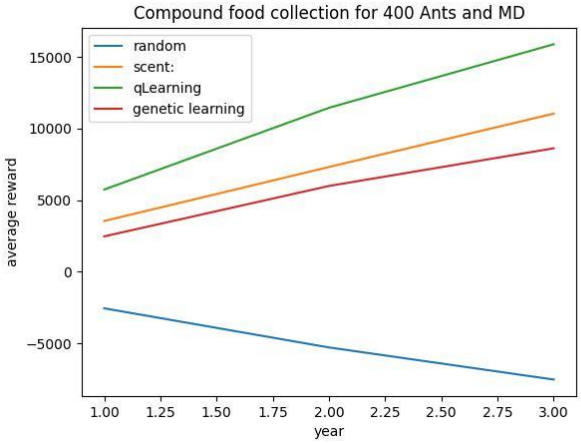

Different Movement Algorithms For 400 ants
random
scent:
qLearning
genetic learning
average reward
food replacement density

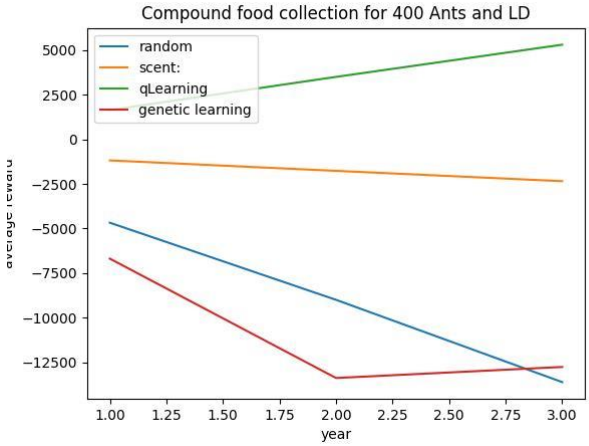

Different Movement Algorithms For 300 Ants
random
scent:
qLearning
genetic learning
average reward
food replacement density

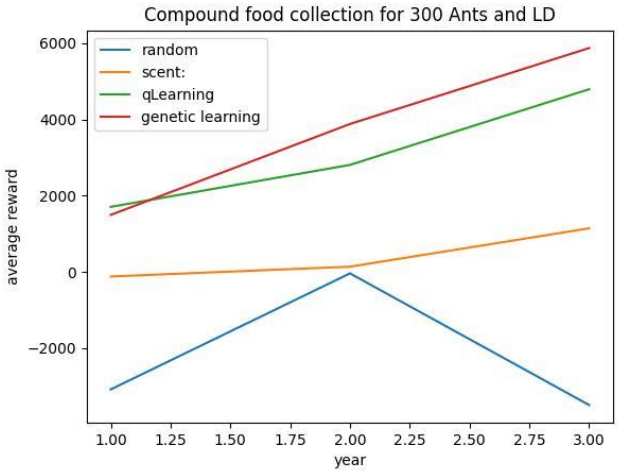

Compound food collection for 500 Ants and MD
random
scent:
qLearning
genetic learning
year

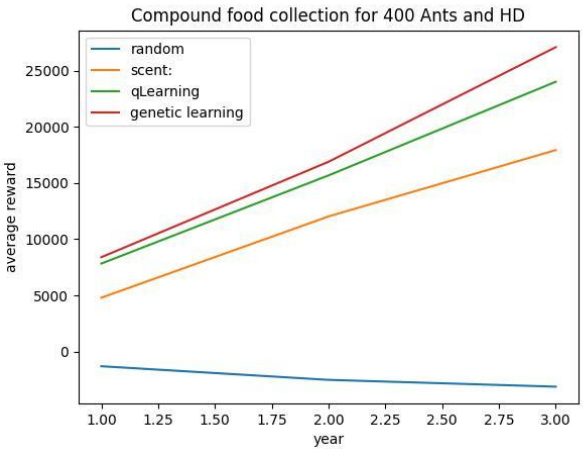

Compound food collection for 500 Ants and LD
random
scent:
qLearning
genetic learning
year

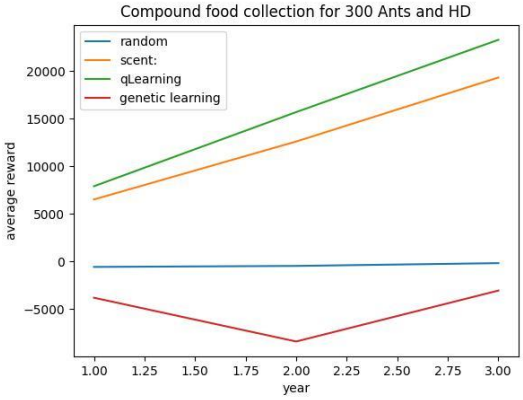

Compound food collection for 500 Ants and HD
random
scent:
qLearning
genetic learning
average reward
year

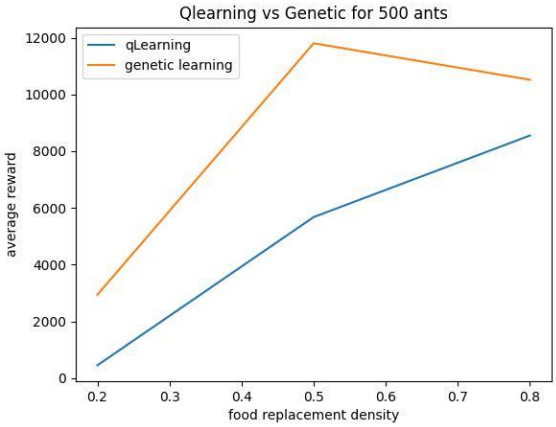

Compound food collection for 400 Ants and MD
random
scent:
qLearning
genetic learning
average reward
year

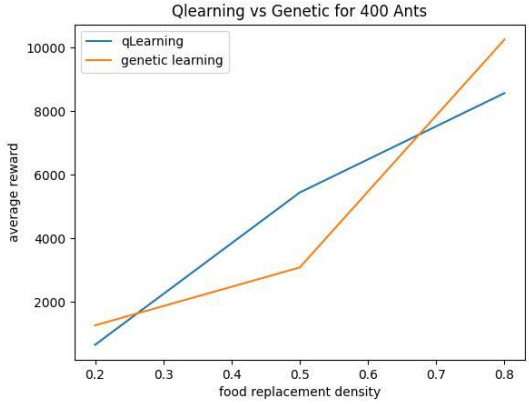

Compound food collection for 400 Ants and LD
random
scent:
qLearning
genetic learning
year

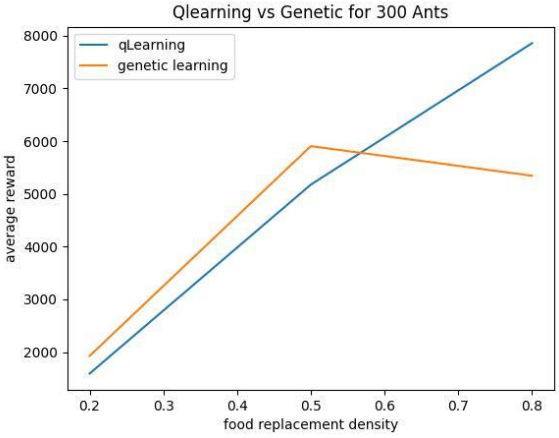

Compound food collection for 300 Ants and LD
random
scent:
qLearning
genetic learning
average reward
year

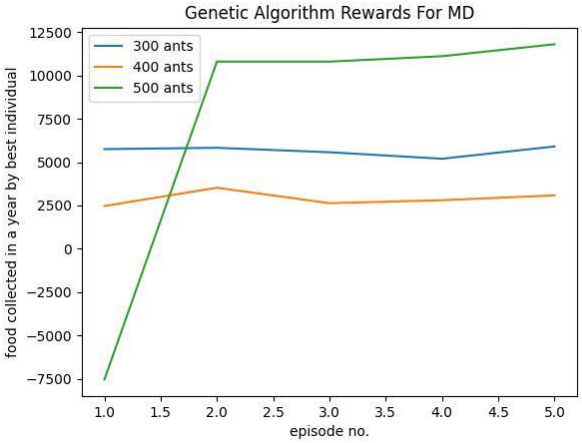

Compound food collection for 400 Ants and HD
random
scent:
qLearning
genetic learning
average reward
year

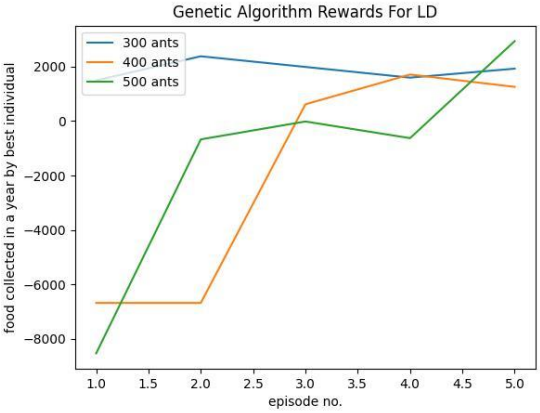

Compound food collection for 300 Ants and HD
random
scent:
qLearning
genetic learning
average reward
year

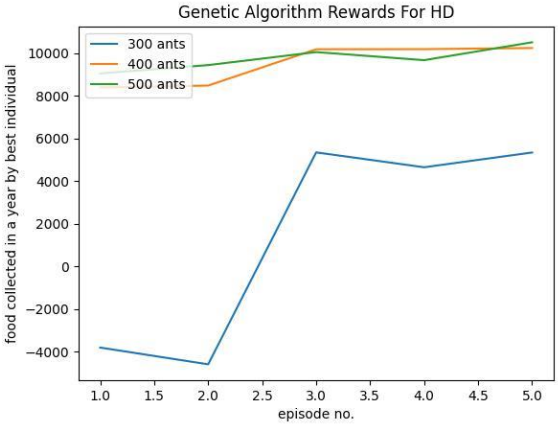

Qlearning vs Genetic for 500 ants
qLearning
genetic learning
average reward
food replacement density

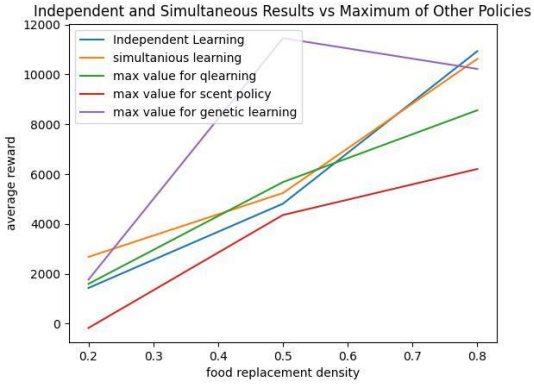

Qlearning vs Genetic for 400 Ants
qLearning
genetic learning
average reward
food replacement density

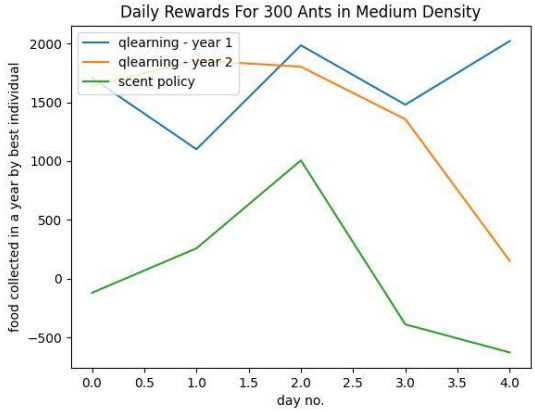

Qlearning vs Genetic for 300 Ants
qLearning
genetic learning
average reward
food replacement density

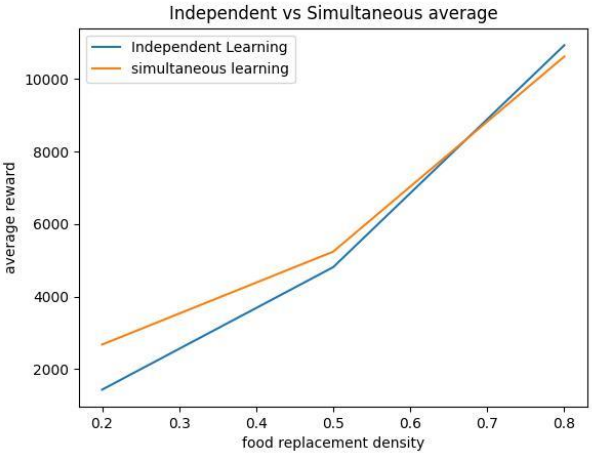

Genetic Algorithm Rewards For MD
300 ants
400 ants
500 ants
food collected in a year by best individual
episode no.


Genetic Algorithm Rewards For LD
300 ants
400 ants
500 ants
food collected in a year by best individual
episode no.


Genetic Algorithm Rewards For HD
300 ants
400 ants
500 ants
food collected in a year by best individual
episode no.


Independent and Simultaneous Results vs Maximum of Other Policies
Independent Learning
simultanious learning
max value for qlearning
max value for scent policy
max value for genetic learning
average reward
food replacement density


Daily Rewards For 300 Ants in Medium Density
qlearning - year 1
qlearning - year 2
scent policy
food collected in a year by best individual
day no.


Independent vs Simultaneous average
Independent Learning
simultaneous learning
average reward
food replacement density