Graph-Based Ant Colony Simulation

Shao-Hsuan Chu Meng-Ting Wu, Hung-I Lin, and Ju-Ting Chen

Abstract—Ant Colony Optimization(ACO) algorithm is a popular metaheuristic algorithm based on the behaviors of animals. The pheromone ants leave on their trails is especially useful to solve route-finding problems. In this paper, some rules of the algorithm are modified to make it closer to biological behaviors in nature. We build our simulation on a modified directed acyclic graph, which prevents an ant from following its own trail and enables us to observe in discrete configurations. In order to give a better view of the nature-inspired techniques and cross-platform compability, we implement the artificial colony in JavaScript/p5js.

Index Terms—Swarm Intelligence, Metaheuristics, ACO, Graph Theory.

1 Introduction

SWARM Intelligence based algorithm is a technique which simulates the behavior of creatures in nature. This concept is used widely in Artificial Intelligence. Boids, an early proposed concept developed by Craig Reynolds in 1986, is an Artificial Life program which mimics the behavior of flock of birds. Some metaheuristic algorithms are greatly inspired by biological behaviors. Ant Colony Optimization is one of the most popular metaheuristics, introduced by Marco Dorigo, which has several individual ants as agents to search for a better solution in the search space.

1.1 Introduction of Ant Colony Optimization

Bionics, which refers to the study of mechanical systems that function like living organisms. Marco Dorigo et al.[1] takes the idea of Bionics to the computer world. The paper points out the connection between the foraging behavior of ants and the combinatorial optimization problems, for instance, the famous Travelling Salesman Problem(TSP). The pheromone left by ants to find food is the key component of the optimization algorithm. It can be divided into two parts in the first ACO algorithm, Ant System(AS), pheromone update and route selection.

Live demo: https://ernestchu.github.io/ACO-Web/
Source code: https://github.com/ernestchu/ACO-Web

1.1.1 Pheromone Update

The paper states that pheromone are left by the ants who found food, and the other ants can follow their path to find the food. Like the smell in our world, pheromone evaporates over time. Therefore, an evaporation rate ρ is imposed on the existing pheromone τ.

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \sum_{k=1}^{m}\Delta\tau_{ij}^{k}$$

where τ_{ij} stands for the pheromone on the route from point i to point j
$\Delta\tau_{ij}^{k}$ is defined by

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k & \text{if ant k used edge (i,j)in its tour} \\ 0 & \text{otherwise,} \end{cases}$$

where Q is a constant, L_k is the length of route. It shows that the intensity of pheromone is inversely proportional to the length of route, which eventually makes ants to take the shortest route.

1.1.2 Route Selection

The paper claims that ants select the next points by a stochastic mechanism. When ant k is in point i and has the partial solution s^p, the probablility of going to point j is

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{c_{ij}\in\mathbf{N}(s^p)}\tau_{il}^{\alpha}\cdot\eta_{1l}^{\beta}} & \text{if } c_{ij}\in\mathbf{N}(s^p), \\ 0 & \text{otherwise,} \end{cases}$$

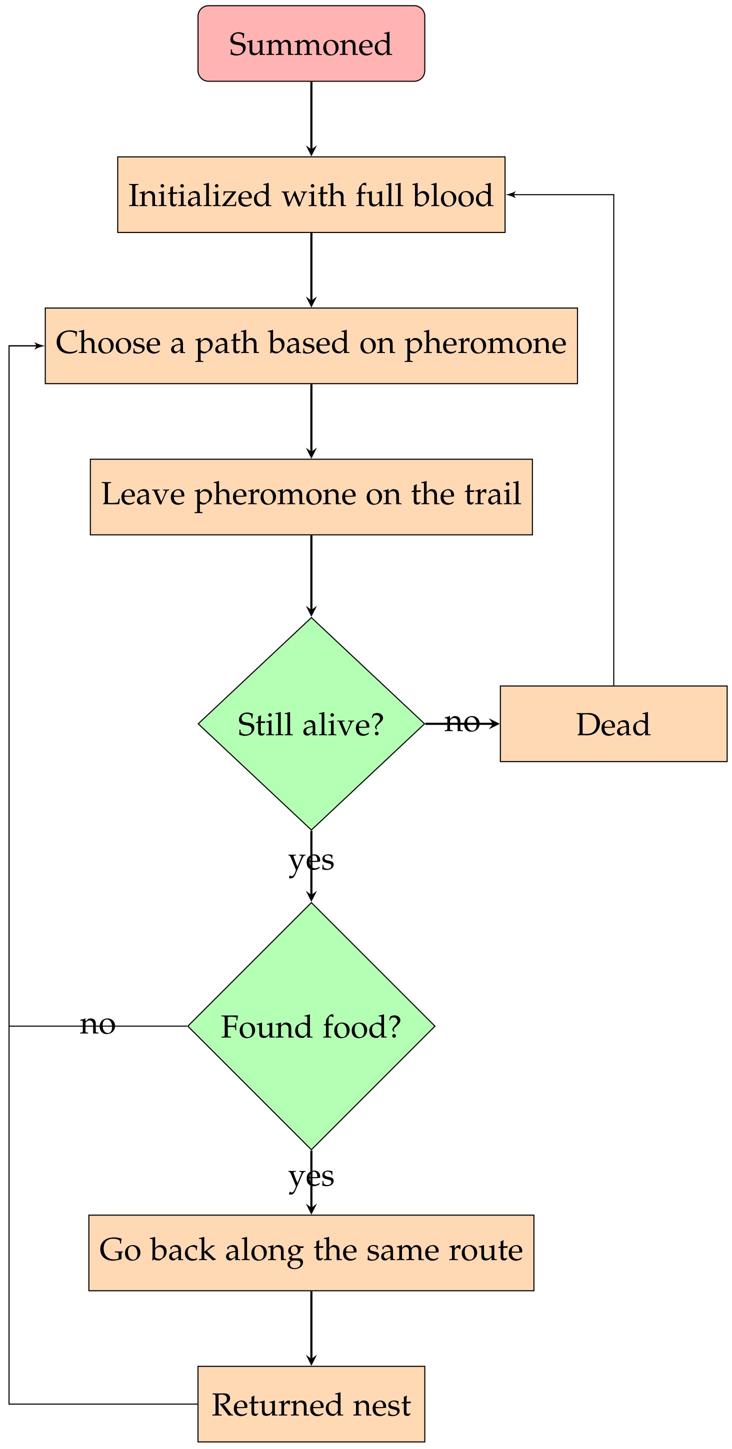
where α and β controll the importance of pheromone versus η_{ij}, which is given by.

$$\eta_{ij} = \frac{1}{d_{ij}},$$

Where $\frac{1}{d_{ij}}$ is the distance between points i, j. Intuitively, the shorter route means the ants comes back quicker, which attract the other ants since there's no other ants come back from another route. As a result, those ants attracted reinforce the intensity of the pheromone on the route they take.

2 The Ant Colony Simulation

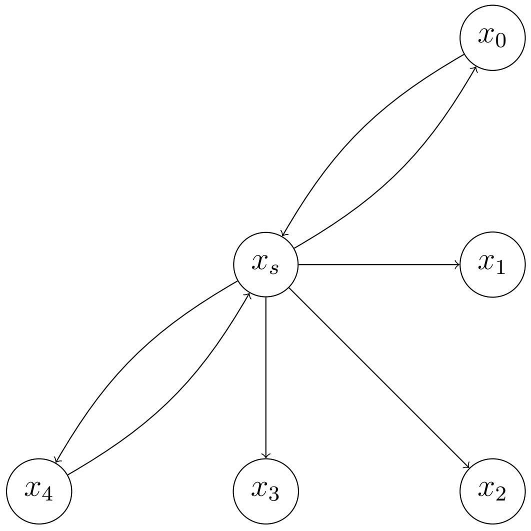
At the start of the program, a fixed number of ants is summoned to the world. The world is essentially a directed acyclic graph (DAG) with some exceptions. The ants can visit reachable vertices and leave the pheromone on the edges of the graph. While leaving the pheromone, the ants also record the edges they've taken, so they can go back to the nest as long as they found food or there's no reachable vertex. The basic workflow of every ants in the program is as follow:


Summoned
Initialized with full blood
Choose a path based on pheromone
Leave pheromone on the trail
Still alive?
no
Dead
yes
Found food?
no
yes
Go back along the same route
Returned nest

The ants are initialized with full blood, which decrements in a constant rate. When an ant has no blood, it's dead and it a new ant will be summoned immediately. So in the program implementation, we can just reinitialize the very ant. Also, the world contains not only ants but food and obstacles. We will elaborate below.

2.1 The Modified DAG

In our modified DAG, the nest, i.e. the statring vertex, is at the upper left most and each vertex has directed edges to the lower-right, right, lower verticess, which follows the rules of DAG. However, for the purpose of exploring upper right and lower left of the world, we added the connections to the directions respectively as the following figure illustrated.



Where x_s represents the starting vertex and the bidirectional edge is one edge instead of two seperated edges. This auxiliary connections violet the rules of DAG and lead to the infinity cycles in which the ants would be trapped. Supposed an ant visit x_0 from x_s. Because of the pheromone affinity, the ant would have high possibility taking the edge which it just took and goes back to the vertex where it came from, and so on and so forth. To prevent this, we added a constraint in our program, which forbids the ants from taking the bidirectional edge again.

2.2 The pheromone affinity

We know that an ant makes its decision on the edges based on the pheromone they last. The probablility p_{ij} of each vertex j from vertex i is given by

$$p_{ij} = \begin{cases} p_{pher}\frac{\tau_{ij}}{\sum_{k\in\mathbf{Adj}(i)}\tau_{ik}} & \\ +(1-p_{pher})\frac{1}{\|\mathbf{Adj(i)}\|} & \text{if } j \in \mathbf{Adj}(i), \\ 0 & \text{otherwise,} \end{cases}$$

where $\mathbf{Adj(i)}$ is a set of reachable vertices from vertex i, τ_{ij} is the pheromone and p_{pher} is the probablility of an ant affected by the pheromone. The ants do not follow the pheromone every time so it's less likely to fall into local optima. In addition, we also remove some reachable vertices in the boundary condition as the ants cannot go out of the world.

2.3 The Food

There are there cases that an ant doesn't continue to explore the world and go back to the nest. First, there's no reachable vertices due to either the obstacles or the boundary of the world. Second, the ant died of the hunger, i.e. the blood became zero. Third, they found food. We can consider the constraint of the blood as the longest path an ant is allowed to explore. It also helps the ant colony to find the shortest path toward food easily. When an ant found the food, it's blood gets refilled so it can bring the food back to the nest. In the program, food can be placed dynamically and we can place multiple units of food in the world. Sometimes, the ants would stuck at a local optimal path from the nest to boundary. That's is because the ants leave pheromone as soon as they leaved the nest and when they hit the boundary, they go back without producing pheromone. Even with such configuration, the ants still have chances to stuck at a nest-to-boundary path. It's most likely when they finished a food unit near the boundary. In our program, we provide a button to eliminate all the pheromone in the world, so the ants can search for food randomly again.

2.4 The Obstacles

As mentioned above, the reachable vertices can be removed by the obstacles. In our program, obstacles can be placed dynamically on a certain vertex, which disable the very vertex so no ant can visit it. We add the setting of obstacles in order to observe how ant colony acts in an irregular landform. However, we allow the ants to cross the obstacles when they're going back to the nest, which is necessary otherwise they don't know how to get home. In the real world, ants have antenna to communicate with each other, and we believe that this is how ants overcome the dynamic changes of the landform. But we didn't include such setting in our program.

3 Conclusion And Observation

Several decades of development in Swarm Intelligence has seen plenty exciting works. The purpose of this research is to visualize the artificial biological behaviors in such algorithms while there are still some space for improvement. We noticed that there're several hyper parameters that matter a lot w.r.t. the operation of the colony, including the decay rate of pheromone, the probablility of being affect by pheromone and the blood decrement rate. We tested several combinations and finally got an presentable outcome. Still, the ants stuck at the local optima sometimes as the world is finite. As state-of-the-art nature-inspired algorithms springing up, we dedicate to implement them in future works.

References

[1] M. Dorigo, M. Birattari and T. Stutzle, *"Ant colony optimization"* in IEEE Computational Intelligence Magazine, vol. 1, no. 4, pp. 28-39, Nov. 2006, doi: 10.1109/MCI.2006.329691.