

Simulation interface for food harvesting by a colony of ants in an unknown environment with a dynamic Ant Colony Optimization (ACO) inspired behaviour

Bertille BEAUJEAN
(Dated: January 18, 2026)

This project is about developing a simulation of an ant colony in an unknown environment scattered with obstacles and food. The goal is for the ants to leave the nest, explore the unknown environment, collect the food and bring it all back to the nest. The exploration is based on a bio-inspired group dynamic which is the foundation of the Ant Colony Optimization algorithm (ACO). The model should try to optimize the time spent to collect all the food and the distance walked by the ants.

The simulation is written in Python and Tkinter library is used for the graphic interface.

The environment is a grid where each cell can be empty, have food, be an obstacle or be the nest. In the simulation, every ant is an independent agent and remembers the cells it has already visited. To help the colony work together, ants spread pheromones on the path they take to bring back food to the nest. Pheromones fade and spread over time so that ants can adapt to changes in the environment. When they leave the nest, they scan the presence of pheromones or obstacles around them and remember the intensity of the pheromones. Hereby, they create a map of potential of pheromone intensity, drawing them towards the paths where other ants carried food. The direction of the ants is then defined with a gradient descent algorithm on these pheromone maps of potential. Thereby, the ants are drawn towards the zones where a lot of other ants carried food. This makes the simulation realistic and shows how ants find the best paths to collect food efficiently. When bringing back the food to the nest, the ants should remember their current position and the position of the nest and proceed to go back there with a Bug 2 algorithm.

The simulation should aim at a best case scenario, where the positions of the food and obstacles are known *a priori* and the total distance walked by the ants to collect all the food is minimized.

The interface shows the environment in real time, with different colors for obstacles, food and pheromones.

The code should be organized into four main classes: Environment, PheromoneMap, Ant, SimulationGUI, and the main function. It is around 700 lines.

GitHub: <https://github.com/GitBB3/ComputerExercise>

I. INTRODUCTION & RELATED WORK

Bio-inspired behaviours are increasingly studied in swarm robotics to control several robots in an efficient way: flocks of birds, bee colonies or ant colonies. They have the benefit of favouring decentralized control and local interactions to enable tasks that would be more difficult for a single robot. [1]. Ant Colony Optimization (ACO) is a metaheuristic to solve optimization problems based on the ability of ants to find shorter paths between their nest and food. In nature, ants leave pheromone trails to communicate with each other and the others tend to follow the pheromone trails with stronger scents. As shorter paths are faster, the pheromones are accumulating faster on the shorter paths. Thereby, more ants are likely to follow those shorter paths and to spread even more pheromones. This process leads to the collective discovery of short paths, even if they are not necessarily optimal. [2]

ACO uses several agents (ants) to find the one best path to reach all food sources successively with one agent. The algorithm of ACO requires knowing the location of all targets at the beginning of the exploration. On the contrary, in this project, one aims at collecting food from unknown locations, so the position of food will be guessed and verified eventually along the collection. Hereby, rather than computing one path from food source to food

source for one agent, the chosen strategy is to collect the food step by step, every time that an agent finds some. It is considered that an agent can only visit one food source before going back to the nest. The environment will be discovered progressively and the agents will eventually be able to aim their exploration towards the most promising paths.

Hence, rather than using the pre-existing Ant Colony Optimization algorithms, this project aims at defining the key points that should be addressed when building new algorithms for multi-agent Ant Colony Optimization.

II. PROPOSAL

Let us recall that the final objective of the simulation is, for a given environment and a given number of ants, to find an efficient plan to harvest all the units of food present in the environment and to bring them back to the nest. The proposal of this project aims at optimizing this food harvest.

Given that the previous related worked studied tackled Ant Colony Optimization for a single robot, or, in a known environment, in order to create a useful basis for algorithm assessment, the aim of the project shall be to provide a representation of the environment of the simu-

lation along with some relevant tools to assess the accuracy of the optimization algorithms. A minimal tentative implementation of the optimization algorithms shall also be provided. Its accuracy may then be assessed by the implemented tools.

In practice, this project proposes a simulation interface with tunable parameters regarding the configuration of the environment, a simulation window with live evolution of the food harvest by the ants, a dynamic tunable parameter to modify the model during the simulation and live feedback on the efficiency of the method.

A. Optimization strategy

The highlight of this proposal is that the control of the swarm of ants is decentralized. Compared to the usual control strategies used for swarms of agents, every ant shall have its own representation of the environment and the research of food shall be issued independently by every ant. Nevertheless, different ants interact between themselves by the dispersion of pheromones in the areas where food has been found in the past, thus, fostering the collective knowledge about the environment.

On the one hand, this representation sticks to the reality of Nature, where ants only communicate by spreading pheromones. On the other hand, in the eventuality where this optimization strategy would be used to control a swarm of robots and not only to represent a colony of ants, this optimization strategy might enable designing a minimal control system based on pheromones detection on each robot, instead of integrating a complex system of communication to share all the knowledge acquired about the environment.

In practice, as described in the abstract, one may tackle the displacement of ants by describing one typical journey from the nest to a random unit of food. This journey shall be separated in two phases: the exploration phase - to find food, and the return phase - to bring the food back to the nest. The return phase is a usual problem of path planning in two dimensions because the ant remembers the position of the nest and knows the position of the food that it just found. Hereby, any path planning algorithm would be relevant for this part. Obviously, the path planning would better be optimal. The exploration phase is the focus of the research here. It shall be based on a reliable search algorithm to cover the environment and find the units of food, and enhanced by the ability to follow pheromone trails in the environment: that is to say that every ant would look independently for food until it would detect pheromones, after what the direction of its search would be pushed in the direction of the pheromone trail and away from the ant nest. This behaviour should lead the ants closer to the clusters of food which were previously discovered by other ants. Indeed, once an ant picks up a food unit, it should leave spread pheromones on its paths. Eventually, the most promising paths will have stronger pheromones intensity

and drive more ants to them in a self-sustained process.

B. Definition of the environment of the simulation

1. Boundaries

The chosen environment of the simulation is represented by a two dimensions rectangle grid of positions. The position of the ant nest in this environment can be chosen within the chosen boundaries.

2. Food clusters

Similarly to a real environment, food shall be scattered in the environment in clusters: that is to say that several units of food will be located in a close area. Indeed, with such representation of the food, if an ant picks a unit of food, it is then probable that there will be other units of food in the same area. As a reflect of a real environment, we could picture it with the idea that if an ant finds a fruit at the bottom of a tree, it is likely that there will be other fruits at the bottom of this same tree. This configuration gives credit to the pheromone-exploration strategy as the ants are likely to find several units of food in a cluster, and therefore, should go back to explore in the same area.

It is possible to tune the number of clusters, the diameter of a cluster and the minimum and maximum of food units present in a cluster. After tuning those parameters, the clusters and their food unit's position is randomly defined in the environment.

3. Obstacles clusters

It was chosen to scatter obstacles in clusters, in a similar way as food.

4. Pheromones characteristics

The pheromones are modeled in a realistic way with the following parameters: the amount of pheromones spread at each step, the evaporation rate and the diffusion rate. More than reproducing the natural behavior of chemicals, this enables to provide relevant pheromone trails for the ants which would be detectable, geographically precise enough and recent enough to be accurate.

5. Number of ants

The number of ants can also be tuned.

III. RELEVANT FEEDBACK INTERFACE

In order to assess the efficiency of the algorithms, three criteria are considered: the quantity of food harvested with regard to time, the proportion of food harvested with respect to time, the quantity of food harvested with respect to the quantity of pheromones in the environment, the duration of the harvest and the cumulative distance walked by the ants.

The three first indexes of efficiency are dynamically drawn in respective graphs on the feedback window of the interface, to give a live feedback about how the simulation is going.

Last but not least, the simulation is displayed in a window where food and obstacles are shown in different colors, ants move along time and pheromones appear, fade and spread depending on the pheromone characteristics chosen.

IV. IMPLEMENTATION

A. Language and libraries

The simulation is implemented in Python. The following libraries have been used:

- **random**: for the generation of the environment and for the exploration algorithm of the ants
- **math**: for distance computation
- **tkinter**: for interface design (see [3])
- **matplotlib**: for graph display in the interface

The whole code is around 700 lines. Indeed, the point of this proposition is to provide a minimal architecture that would be very easy to integrate in an "ant-robot" if this exploration strategy was used on a swarm of robots. Therefore, the fact that the management of knowledge is decentralized and that all the "ant-robot" have analog roles enables to limit the actual quantity of code to 4 classes. A significant part of the code is focused on the design of the interface and the display of the simulation.

As a matter of fact, as this proposal is focused on the acquisition of a usable simulation interface to test exploration algorithms, the code is also focused on this part of the implementation. However, if we were to reach the development stage to refine the exploration method and improve the search algorithms, more code might then be written in the *Ant* class to add subtlety to the path planning algorithms.

B. Classes

The simulation is implemented with 4 major classes:

- **Environment**:
 - dimensions
 - a position grid (discrete positions with integer positions)
 - position of the nest
 - characteristics of food and obstacle dispersion
 - generator of random environments with food and obstacles scattered in clusters
- **Ant**:
 - a position
 - a local representation of the environment filled with the elements previously scanned
 - a scan function to detect obstacles and pheromones
 - an exploration function (**gradient descent**)
 - a function to go back to the nest (**Bug2**)
- **PheromoneMap**
 - dimensions of the environment
 - a map of pheromone values
 - a function to add pheromones in a cell
 - a function to simulate the evaporation of pheromones over time
 - a function to simulate the diffusion of pheromones over time and space
- **SimulationGUI**
 - run the simulation
 - update the feedback
 - sets the parameters

C. Architecture

```
dynACO/
|-- ant.py/
    # Ant class
|-- config.py/
    # Default parameters of the simulation
|-- environment.py/
    # Environment class
|-- gui.py/
    # Graphic interface
|-- main.py/
    # Main function
|-- pheromone.py/
    # Pheromone class
```

V. EVALUATION

A. Presentation of the interface

Let us present the design of the simulation interface implemented with the *tkinter* library.

Hereafter is the full page interface which appears on the screen of the user when running the main function (Fig. 1).

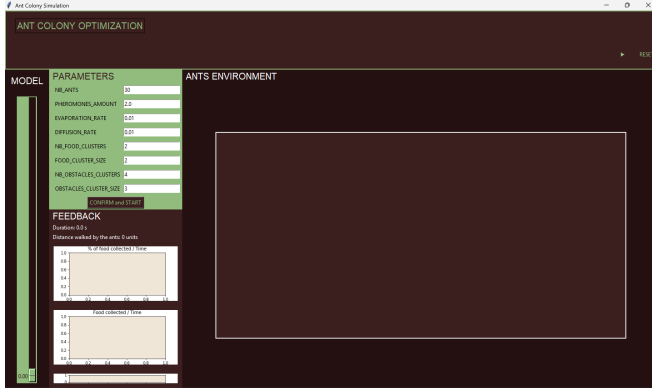


FIG. 1. Full page interface interface right after running the main function provided in the code.

1. Parameters insert

The entry sections on the green insert entitled "Parameters" enable the user to tune the parameters of the simulation that were previously mentioned.

Before launching the simulation, the user shall press the button "Confirm & Start" (Fig. 2) to confirm the new parameters and generate the random environment.

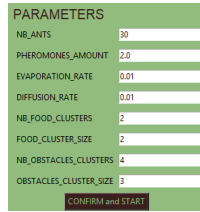


FIG. 2. Parameters tuning

After confirmation, the entries disappear from the green insert and the chosen parameters are displayed instead (Fig. 3).

2. Play/Pause & Reset

In the top right part of the interface, the buttons *Play/Pause* (Fig. 4 left) and *Reset* (Fig. 4 right) shall be pressed to respectively run the simulation, pause the

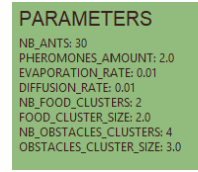


FIG. 3. Parameters information after confirmation

simulation, or stop the simulation and reset the parameters.



FIG. 4. Play/Pause & Reset buttons

3. Display of the simulation

On the biggest and central part of the interface, the environment of the ants is displayed (Fig. 5).



FIG. 5. Enter Caption

The legend is as such:

- orange cells: obstacles
- green cells: food units
- blue cell: the ant nest
- yellow cells with brown center: ants in exploration phase
- yellow cells with green center: ants carrying a food unit
- red cells: intensity of pheromones

4. Assessment of the method

In order to give live feedback along the simulation, the chosen criteria are displayed on the left part of the interface (Fig. 6).

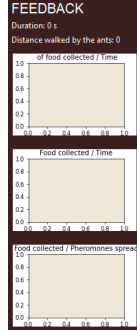


FIG. 6. Live feedback graphs

Above those graphs are also written the duration and the distance walked by the ants.

5. Choice of the model

Finally, the left sidebar enables to dynamically modify the model of exploration along the simulation.



FIG. 7. Tendency to follow pheromone trails from 0 to 1

Indeed, to the tendency of an ant to follow a pheromone trail or to explore randomly is defined by this sliderbar which is automatically set to 1: that is to say that every time that an ant detects a neighbouring cell where the pheromone intensity is higher, in the opposite direction of the nest, it will go in this direction. With a model-index of value m , the ant will follow this direction with a probability of m and otherwise it will stick to the random behaviour.

B. Assessment of the exploration algorithm

As a confirmation that the interface is relevant to assess the efficiency of an algorithm with regard to the chosen criteria, let us compare the efficiency of an exploration with pheromones to a random exploration.

PARAMETERS
 NB_ANTs: 20
 PHEROMONES_AMOUNT: 2.0
 EVAPORATION_RATE: 0.01
 DIFFUSION_RATE: 0.01
 NB_FOOD_CLUSTERS: 2
 FOOD_CLUSTER_SIZE: 2.0
 NB_OBSTACLES_CLUSTERS: 0
 OBSTACLES_CLUSTER_SIZE: 3.0

FIG. 8. Parameters of the simulation

Hereafter (9 & 10) are boxplots of the duration and distance walked by the ants indicated by the interface along the simulation, with both methods, on several experiments (10 with each method). The simulations were all executed in an environment grid of 30×30 cells, with the following parameters (8):

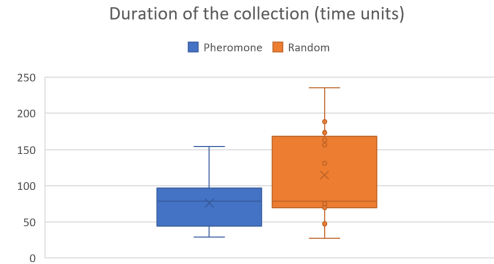


FIG. 9. Comparison pheromone exploration vs random exploration (duration)

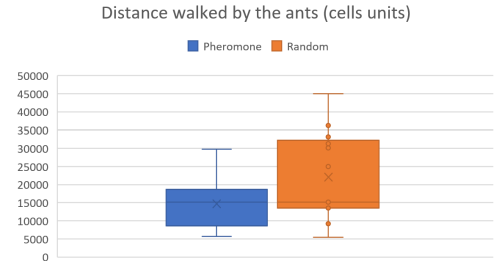


FIG. 10. Comparison pheromone exploration vs random exploration (distance)

Statistically, one may highlight that the method with pheromones is visibly more efficient than a random exploration, regarding the time of execution and the total distance walked by the ants.

However, if this comparison underlines the potential of the exploration with pheromones, it concludes on the efficiency of the interface more than on the algorithm, and modifications should be made on the exploration algorithm to reach further expectations.

C. Discussion

One may wonder why the default exploration algorithm used to guide the pheromone driven model is a random brownian movement instead of a more efficient algorithm. Indeed, when the exploration is completely random, the collection of food is completely probabilistic and it is possibly one of the lowest levels of intelligence that the ants could get. As it is an early stage of the development of the pheromone algorithm, this is interesting for this simulation because it highlights the improvements made possible by the pheromones: if the pheromone algorithm is efficient, the difference from a "dummy" random exploration should be more striking. Furthermore, I would like to keep the algorithm minimal and basic, so that the control algorithms of each ant would have a low complexity. As a matter of fact, if the pheromone model had to be improved, I would try to focus on the best way to manage decentralized control of the ant-agents. To achieve this, I would try to control every ant with the same pheromone algorithm, which should only be influenced by the intensity of pheromones around it and not by direct communication with other ant-agents. Afterwards, once the pheromone-attraction part is convincing, it would be possible to substitute the "random exploration" function with another function. Indeed, the way that the exploration phase was implemented should enable modularity and it should be easy to extend.

To address specifically some questions asked during this project, let us underline that, for the same reasons, the "pheromone-spreading phase" (when the ants go back to the nest) was kept to a basic Bug2 algorithm instead of choosing a more powerful algorithm like A*. At the first stage of research, choosing algorithms which are easy to implement and easy to debug enables to spot errors in the invented code more easily. As a matter of fact, the obstacles have been designed in clusters in this simulation but this is an arbitrary choice and changing this design would probably mean changing the obstacle avoidance algorithm. Again, researching for bibliographic comparison of algorithms for the avoidance of different shapes of

obstacles is a matter for research. Rather than searching for it, thinking about this issue enabled to highlight that it is important to have an efficient and trustworthy obstacle avoidance algorithm which enables to spread pheromones on the shortest usable path between the food and the nest.

Last but not least, one might have liked to spawn several colonies of ants in the same environment to simulate direct confrontation between two different algorithms. However, at the moment, it is possible to test several algorithms in the same environment, so a competition between ants might be very visual and entertaining but figures would be more tangible to tune and improve the algorithms.

VI. CONCLUSION

All in all, the goal of this project was to obtain a relevant simulation interface for future research about Ant Colony Optimization and I was able to implement a Tkinter simulation interface with dynamic representation of the ant colony and tunable parameters to modify easily the size and disposition of the environment and the attraction of ants toward pheromones. The chosen criteria for a prior assessment of the efficiency of an exploration algorithm are relative to the duration of the food collection and to the correlation with the amount of pheromones spread in the environment.

What is more, a tentative architecture for an ant colony research model with decentralized control has shown encouraging results when assessed with this interface and it might be paving the way for future improvements, by integrating state-of-the-art exploration and obstacle avoidance algorithms instead of random movement and Bug2 algorithm.

The goal of this project has been reached by designing the simulation interface and being able to conduct an experimentation campaign with it. Even though it would be tempting to improve the exploration algorithm based on pheromone attraction, this is a matter for research. Instead, for this Computer Exercise class, I favoured the code implementation, which enabled me to learn how to use the *Tkinter* library of Python for interfaces.

-
- [1] O. Bezsonov, S. Rutska, O. Rudenko, and S. Piskunov, Using swarm algorithms to explore unknown areas in ros2 (2025).
 - [2] C. Blum, Ant colony optimization: A bibliometric review,

- Physics of Life Reviews **51**, 87 (2024).
- [3] TK commands, version 8.6.17 .