Ant Optimization Algorithm

Maxine Amin[1], Brian Godwin Lim[1]
[1]Department of Mathematics, Ateneo de Manila University

Abstract

The Ant Optimization Algorithm is inspired by the communication system between colonies of ants using trail pheromones. The algorithm makes use of the previous positions of particles (ants) to determine the direction of the next position for each particle. The algorithm assumes that pheromones diffuse after 1 iteration; thus, only the previous positions affect the current positions of the particles. Our experiment shows that the proposed algorithm outperforms the Particle Swarm Optimization in some test functions; however, the algorithm is still in need of further analysis.

Keywords: nature-inspired algorithm, ant optimization, pheromones

1 Introduction

Inspiration Communication in any form of society is crucial to ensure its survival. No exception to this are *ant colonies*, consisting of as many as 306 million worker ants (*Understanding chemical communication in ant societies*, n.d.). Ants are social insects. While humans communicate through the five senses, ants, on the other hand, use chemical signals called *pheromones* to communicate with their nestmates. Ants use pheromones in many situations. For instance, ants can produce *trail pheromones* in warning other ants about nearby predators, defending their colony, or in foraging for food, leaving pheromones from the nest to food sources to lead other ants to the location, and additional ants may also deposit trail pheromones to reinforce the original trail (Chalissery et al., 2019). Thus, pheromone concentration can then vary in different trails. This whole setup can be commonly seen in how ants travel together: they place pheromones as they walk, while other ants behind would follow this trail, leading them from one place to another in a line. Through this usage of pheromones, they are able to establish an efficient system for food gathering.

Information Exchange Ants use their antennas to detect pheromones and respond to situations accordingly (*Understanding chemical communication in ant societies*, n.d.). Being highly sensitive and containing both touch and smell organs, ants' antennae are even able to detect their fellow nestmates from intruders. In the case of trail pheromones, information regarding food quality is communicated to other ants who have no first-hand contact with the food source. Higher trail pheromone concentration (i.e., the amount of pheromone deposited on a trail) increases the likelihood of other ants exploring this trail. "After deposition on the substrate, a trail pheromone diffuses" (Carde & Resh, 2009).

2 Proposed Algorithm

Ant Optimization Algorithm We propose an optimization algorithm based on the behavior of ants and their use of trail pheromones. Central to the algorithm is the positions of particles (ants) at time t are dependent on the previous positions. In our version of the algorithm, we assumed that the pheromones diffuse after 1 iteration. Thus, the position of the i^{th} particle at time $t+1$, $P_{t+1}(i)$, is dependent on the set of positions at the last iteration, P_t. The objective of the algorithm during each iteration is for each particle to find one trail pheromones represented by p_{max} (chosen randomly from P_t) and move towards its direction by a factor of $c \times r_{[0,1]}$. After several iterations, all particles will converge to the optimal point.

Update Equation The update equation is given by

$$P_{t+1}(i) = P_t(i) + c \times r_{[0,1]} \times (p_{max} - P_t(i)) \tag{1}$$

where:

c is the scale factor,
$r_{[0,1]}$ is a random number uniformly chosen from $[0,1]$,
p_{max} is the point of maximum value among the randomly chosen points, and
$P_{t+1}(i)$ is the position of the i^{th} particle at time $t+1$.

Pseudo code The pseudo code of the proposed algorithm is shown below. The implementation of the Ant Optimization in Matlab/Octave can be downloaded here.

Algorithm 1 Ant Optimization

```
1: while termination not met do
2:     for each particle $i$ do
3:         Generate $n$ points from $P_t$
4:         Evaluate the function at each point
5:         Assign the maximum of these points to $p_{max}$
6:         $P_{t+1}(i) \leftarrow P_t(i) + c \times r_{[0,1]} \times (p_{max} - P_t(i))$
7:         Update the global best $g_{best}$
```

Comparison with Other Algorithms The Ant Optimization (AO) Algorithm has a similar update equation as the Particle Swarm Optimization and even behaves asymptotically like it as the iterations increase. However, the Ant Optimization does not have a velocity variable, and there is no contribution from the local best position (i.e. $w = 0,\ c_1 = 0$). Instead, the perturbation is a scaled (by $c \times r_{[0,1]}$) vector from the current position to the p_{max}. The $r_{[0,1]}$ ensures non-linearity in the update equation. The positions are then incremented by this vector.

Moreover, the selection criteria of p_{max} also follows the selection criteria of the Genetic Algorithm. The probability that the position of particle i at time t will be chosen as a p_{max} at time $t + 1$ is directly proportional to its function evaluation $f(P_t(i))$. This further ensures non-linearity in the update equation since every particle position is eligible to be chosen as a p_{max}, although with varying probability. The implementation is adapted from the an implementation of the Genetic Algorithm.

Like the proposed algorithm, the Ant Colony Optimization (ACO) is an existing algorithm that also utilizes the concept of trail pheromones, but its application in solving computational problems is limited to the problem of finding the best path on a weighted graph (Dorigo, 2007). Furthermore, while the former assumes that pheromones diffuse immediately in a way that only the previous iteration affects the current positions of the particles, the latter assumes the ants' 'memory' of their paths by considering how pheromones diffuse after a period of time.

3 Performance Analysis

Comparison with Particle Swarm Optimization We compared the performance of Ant Optimization to Particle Swarm Optimization. We chose the parameters $c = 0.3$ for the former and $w = 0.5$, $c_1 = 2, c_2 = 2$ for the latter. In both algorithms, we used 500 particles. Setting the same termination criterion of $\sigma(P) < 0.005$, we obtained the following results:

3.1 Sphere Functions

Using the function $f(x) = -(x - 5)^2$, the Ant Optimization terminates after 46 iterations while the Particle Swarm Optimization terminates after 88 iterations.

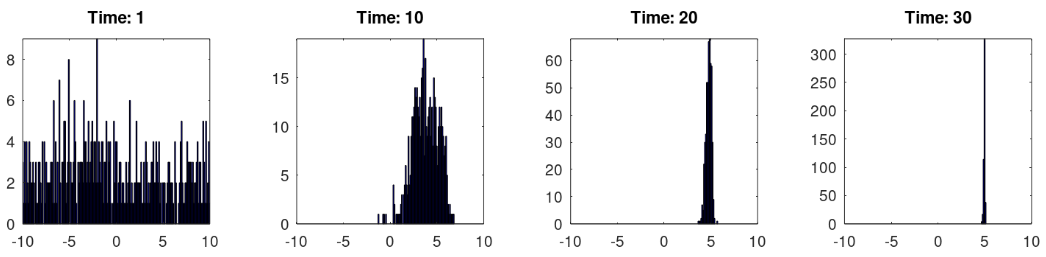

Figure 1: Ant Optimization on $f(x) = -(x - 5)^2$

Using the function $f(x, y) = -(x-5)^2 - y^2$, the Ant Optimization terminates after 48 iterations while the Particle Swarm Optimization terminates after 103 iterations.

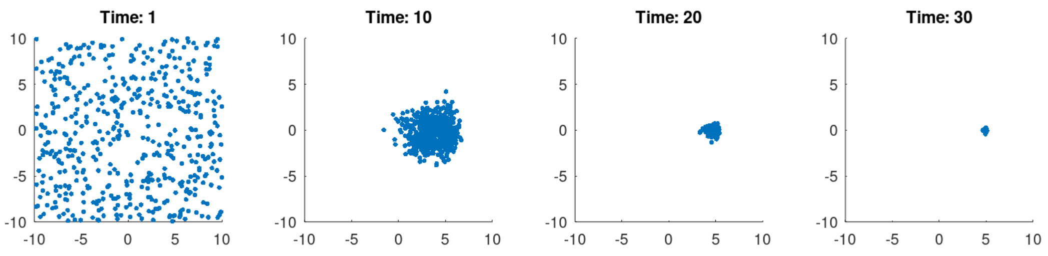

Figure 2: Ant Optimization on $f(x, y) = -(x - 5)^2 - y^2$

In both cases, the Ant Optimization is twice as fast as the Particle Swarm Optimization. However, this result may be due the parameters selected.

3.2 Periodic Functions

Using the function $f(x) = \sin(x)$, the Ant Optimization terminates after 50 iterations while the Particle Swarm Optimization did not terminate after 2000 iterations. However, in the latter algorithm, looking at the histogram of the positions of the particles, we saw majority of the particles clustered around an optimal point, but some particles wandered off to other optimal points. This prevented the code from terminating due to the terminating condition imposed. A closer look at the histogram shows that particles started to converge to different optimal points after 40 to 50 iterations.

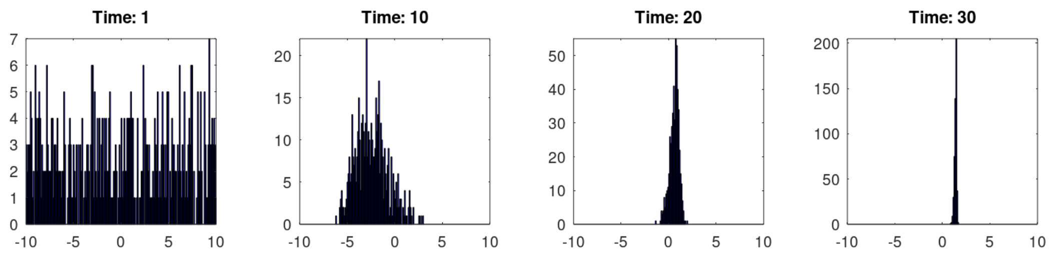

Figure 3: Ant Optimization on $f(x) = \sin(x)$

Using the function $f(x, y) = \sin(x) + \cos(y)$, the Ant Optimization terminates after 49 iterations while the Particle Swarm Optimization did not terminate after 2000 iterations. However, in the latter algorithm, looking at the scatter plot of the positions of the particles, we saw majority of the particles clustered around an optimal point, but some particles hesitated to join the others at the same optimal point. Consequently, this prevented the code from terminating due to the terminating condition imposed. A closer look at the scatter plot did not show any conclusive pattern since particles still move a lot after 500 iterations.

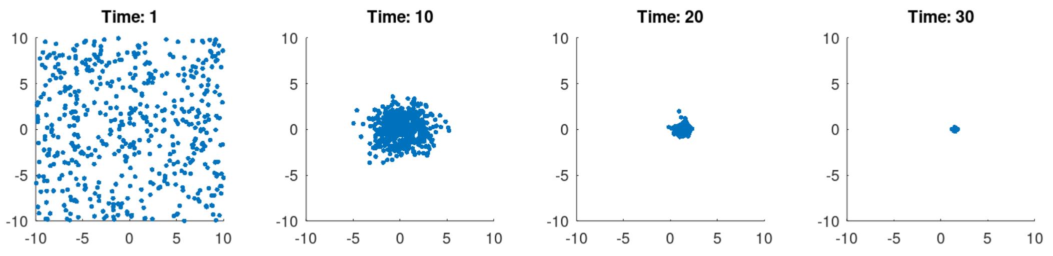

Figure 4: Ant Optimization on $f(x, y) = \sin(x) + \cos(y)$

In both cases, the Ant Optimization produced an optimal point under the termination condition imposed while the Particle Swarm Optimization did not. However, the non-convergence of the latter might be due to the conflict between p_{best} and g_{best} since there are multiple optimal points. A change in parameter selection might alleviate this problem.

3.3 Other Functions

Using the function $f(x) = \tan(\sin(x)) - \sin(\tan(x))$, the Ant Optimization terminates after 47 iterations while the Particle Swarm Optimization did not terminate after 2000 iterations. A similar problem arises with periodic functions.

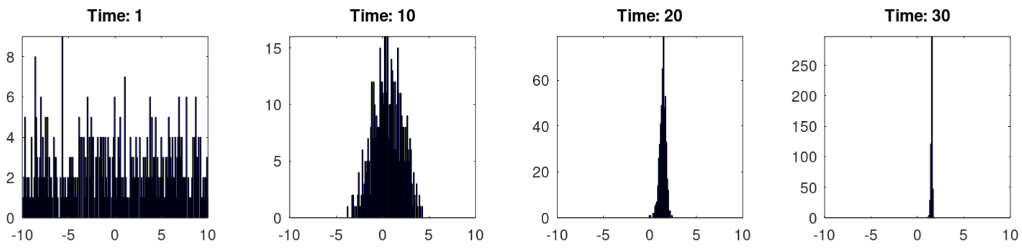

Figure 5: Ant Optimization on $f(x) = \tan(\sin(x)) - \sin(\tan(x))$

Using the Easom function $f(x, y) = \cos(x)\cos(y)\ e^{-(x-\pi)^2-(y-\pi)^2}$, the Ant Optimization terminates after 48 iterations while the Particle Swarm Optimization terminates after 118 iterations.

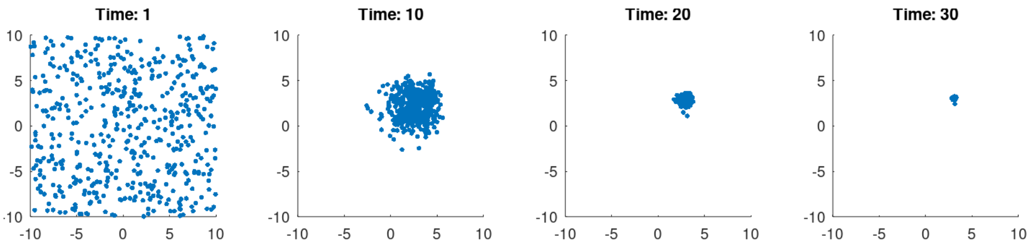

Figure 6: Ant Optimization on $f(x, y) = \cos(x)\cos(y)\ e^{-(x-\pi)^2-(y-\pi)^2}$

Using the Matyas function $f(x, y) = -0.26(x^2 + y^2) + 0.48xy$, the Ant Optimization terminates after 38 iterations while the Particle Swarm Optimization terminates after 148 iterations.

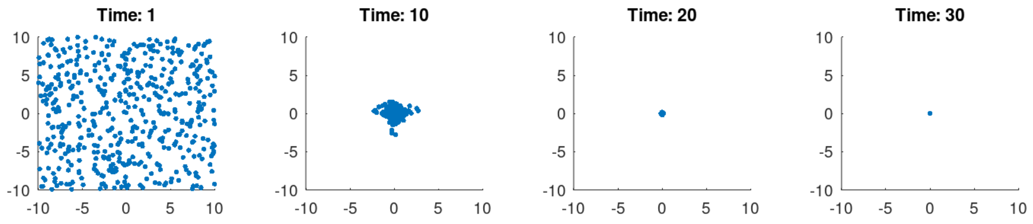

Figure 7: Ant Optimization on $f(x, y) = -0.26(x^2 + y^2) + 0.48xy$

Using the Bukin function N.6 $f(x, y) = -100\sqrt{|y - 0.01x^2|} - 0.01|x + 10|$, and restricting the domain to $(x, y) \in [-15, -5] \times [-3, 3]$, the Ant Optimization terminates after 61 iterations while the Particle Swarm Optimization does not terminate after 2000 iterations.

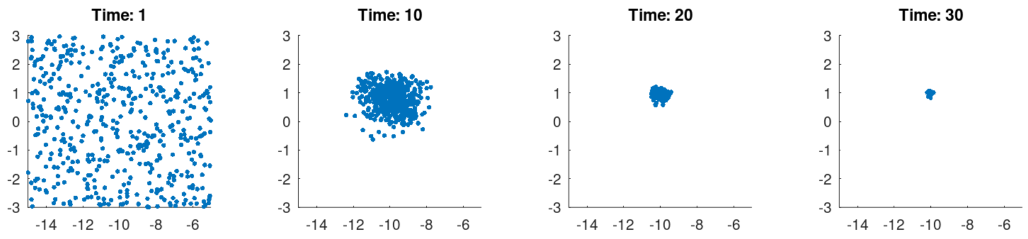

Figure 8: Ant Optimization on $f(x, y) = -100\sqrt{|y - 0.01x^2|} - 0.01|x + 10|$

Parameter Selection The sole parameter in the proposed Ant Optimization is c or the scale factor of the perturbation. In order to test the convergence of the algorithm with respect to parameter selection, we ran 100 simulations of the Ant Algorithm with 100 equally spaced values from $[0, 4]$. Due to computational limitations, a limit of 200 iterations was imposed to the simulations. Furthermore, the function $-x^2$ was used as the benchmark since it does not pose any complications such as periodicity. The result is shown below.

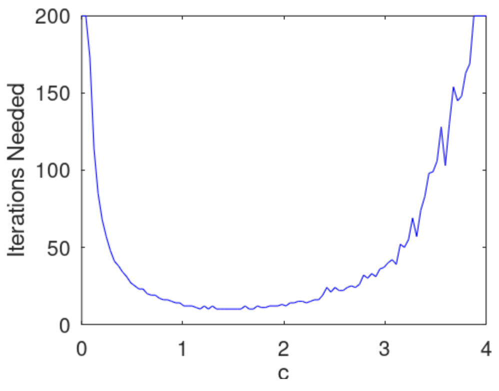

Figure 9: Parameter Selection

The plot above shows that the optimal value for c lying between 1 to 2, where the algorithm terminates, consequently converges after less than 20 iterations. Checking this hypothesis against the functions in the previous page confirms the parameter selection criteria. A value of $1 < c < 2$ makes the Ant Optimization terminate in about 20 iterations.

Mathematical Analysis As seen in Figure 9, the algorithm converges for values near $0.5 < c < 3.5$. However, for small values of $c < 0.01$, the particles hardly move. Thus, it takes a long time for them to converge, if possible. Even after 100 iterations, the particles did not move far from their original positions. On the other hand, for large values of $c > 4$, the particles move a lot that it offshoots and misses its original goal of approaching p_{max}. Therefore, it diverges. For values within the range $0.5 < c < 3.5$, there is a trade off between efficiency and accuracy. Setting $1 < c < 2$ makes the algorithm more efficient, but it can result in sub-optimal results since it does not explore the other points within the search domain. Setting c outside this range is less efficient but gives the particles enough time to explore the other points within the search domain, preventing sub-optimal results.

4 Conclusion

So far, we have observed that Ant Optimization seems to be able to produce the optimal points in a considerably few amount of iterations. Moreover, it also avoids the problems encountered by the Particle Swarm Optimization with periodic functions. However, this fast convergence might result in a premature convergence in mid optimum points similar to the Particle Swarm Optimization. A possible solution to this problem could be to increase the duration of the pheromones before they diffuse since our algorithm assumes that pheromones diffuse after only 1 iteration. Additionally, since the perturbation vector is dependent on the previous positions, a small initial domain might result in sub-optimal results since the particles do not have enough information (trail pheromones) to explore outside the domain. Thus, a sufficiently large initial domain is required to produce the optimal answer.

References

Carde, R. T., & Resh, V. H. (2009). Recruitment Communication. In *Encyclopedia of Insects* (2nd ed.). Amsterdam: Elsevier/Academic Press. Retrieved from https://www.sciencedirect.com/science/article/pii/B9780123741448002289 (OCLC: 775113353)

Chalissery, J. M., Renyard, A., Gries, R., Hoefele, D., Alamsetti, S. K., & Gries, G. (2019, November). Ants sense, and follow, trail pheromones of ant community members. *Insects*, *10*(11). Retrieved 2021-05-17, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6921000/ doi: DOI: 10.3390/insects10110383

Dorigo, M. (2007, March). Ant colony optimization. *Scholarpedia*, *2*(3), 1461. Retrieved 2021-05-28, from http://www.scholarpedia.org/article/Ant_colony_optimization doi: DOI: 10.4249/scholarpedia.1461

Understanding chemical communication in ant societies. (n.d.). Retrieved 2021-05-17, from https://sciencenode.org/feature/understanding-chemical-communication-in-ant-societies.php