Ant colony optimization for TSP

1st Jose Daniel Lopez Hernandez
School of Science and Engineering
Tecnologico de Monterrey
Puebla, Mexico
A01114386@itesm.mx

***Abstract*—Ant colony algorithms have had several different methods related to them, from the original Ant colony optimization presented in 1996 to the optimizations of Ant system with Ant-cycling, Ant-density and Ant-quantity, and the simplification known as SACO (simple ant colony optimization). Ant colony is an algorithm based on graphs, it is used to find the shortest distance between points emulating real ants, one of the most well known graph problems is the NP-Hard problem of the Traveling Salesman Problem or TSP, in this paper, a specialized Ant colony algorithm will be shown that will work for solving the TSP problem.**

***Index Terms*—Ant colony, ant system, TSP, heuristics**

I. Introduction

In his doctoral thesis, Marco Dorigo [1] published about a new optimization method which centered around graphs, this method presented an evolutionary algorithm based on the perceived behaviour of ants. The idea was that ants always seemed to find the shorted route to their intended destination, normally food, but sometimes could be the route back to the colony, further experimentation by biologists discovered that pheromones were left were antes walked, so the route most ants took normally was the one laced with the most pheromones [2]. Aligned with this vision, graph theory distance problems were quick to apply this combinatorial idea of a solution, by lacing edges with pheromones based on the total distance traversed by the "ant" future ants would decide to go through the most edge with more pheromones in it based on a probabilistic decision [3]. This solution was then applied to known problems such as the TSP and optimization of some control PID [4] and usually gives better results at the cost of iteration time [5].

Ant colony is a very robust algorithm so it runs in some problems. For low complexity graphs, it usually has worse efficiency than brute force approaches, and while it has usually good results, the parameters in the method are very sensible to changes, so wrong tuning is usually a problem. Furthermore, ACO tends to get lost in local solutions due to chance, thanks to the way ants choose paths [6].

II. Background

Ant Colony Optimization is a Metaheuristic classified as both, implicit and explicit, while also being a nature-based Metaheuristic akin to evolutionary algorithms. However, ant colony is not an evolutionary algorithm even though it includes population, this is because individuals in ACO do not evolve, they do not get better at solving the problem, but instead the "road" becomes easier to transverse optimally thanks to the pheromones in it [7].

TSP is a very popular problem to test with Ant Colony optimization, [6] discusses the importance of the parameters in ACO when trying to optimize runtime, value and quality of the algorithm. There are several types of ACO that have been implemented including but not limited to:

- Ant System (AS): This is the ACO being tested here, this is the first algorithm created by Marco Dorigo [1].
- Ant colony system (ACS): Which aimed to close on paths much sooner by adding more pheromones per iteration thanks to only the best ant being able to update pheromones [8].
- Elitist ant system: In this algorithm the best visited trails in any generation always receives more pheromones even if it has not been used in the current generation, this happens in unison with the normal updating of pheromones [9].
- Max-min ant system (MMAS): Only the best route received pheromones, however every route starts with maximum pheromones, in order to avoid stagnation, all routes restart at the highest value of pheromones if stagnation start happening [10]
- ASrank: All solutions of a generation are ranked, and only the n best are updated, while shorter solutions deposit more, longer solutions of the accepted ones deposit less [11]

Other methods such as Parallel ant colony optimizations (PACO), continuous orthogonal ant colony (COAC) or Recursive ant colony optimization are very similar to the ones discussed already, but have key differences infered from their names. Applications of these algorithms are not limited to combinatorial applications either, by applying simple ideas of DAC conversions on the search space, it is possible to model ant colonies without much difference [12].

One of the main problems of the ACO however, is the big complexity it has being known as $\Omega(N^2)$ making it unfeasable for big complexity problems. This complexity comes from having to set the route for M ants, N times. M being the number of ants, and N being the number of ants. Furthermore, each time the next node is going to be assigned, every route has to be evaluated until one passes the probabilistic test, this

Identify applicable funding agency here. If none, delete this.

then, is further complicated by generation number [13].

III. Algorithmic approach

In this paper, a normal Ant system will be implemented in python. The steps are declared as follows:

- Creating the graph: By using classes for both the graph and each individual node, a fully connected graph of n nodes is created. TSP needs a fully connected graph. All of the edges will be assigned a random cost between 1 and 10.
- Initializing pheromones: A separated pheromone matrix is created which will include all of the possible routes to be taken, this graph will be referred to as τ, while τ_{ij} will refer to the pheromone level in edge from node i to node j. It is important to note that τ_{ij} and τ_{ji} are two different edges. All edges will be initialized with a very small value of .001 to avoid divisions by zero when evaluating the first generation.
- Initializing ants: All ants will start at node 0, with a cost of 0 and a route with only node 0 in it.
- Filling routes: As mentioned, ants will start at node 0 and one ant at a time will fill their route matrix. This is based on a probabilistic choice referenced in equation 1 where:
  1) alpha is the pheromone relatively importance parameter [6].
  2) beta is the relative importance of heuristic factor
  3) eta is the inverse of the cost of the edge being evaluated
  4) $J_k(i)$ refers to the list of nodes connected to the node i
  5) $P_{ij}^k(t)$ is read as the probability of the edge from node i to node j for ant k being chosen at the time t.

  The selected node to evaluate is selected at random each time, until no nodes are left. If no node is chosen by the probabilistic a random non-visited node will be chosen. This is repeated every generation for each ant.
- Pheromone evaporation: This step removes part of the pheromones in each edge by a fixed rate, ρ is the user defined variable that defines evaporation, as is used in equation 2.
- Pheromone actualization: Finally the ants update the pheromone in each edge they visited by the amount specified in equation 3 where $\Delta\tau_{ij}^k$ is defined as $\frac{Q}{L_k}$ where Q is an user defined pheromone multiplicator (normally initialized as 1) and L_k is the total cost of ant k to complete its route.

$$P_{ij}^k(t) = \begin{cases} \frac{(\tau_{ij}(t))^\alpha * (\eta_{ij}(t))^\beta}{\sum_{s \in J_k(i)} (\tau_{is}(t))^a * (\eta_{is})^\beta} & if j \in J_k(i) \\ 0 & otherwise \end{cases} \quad (1)$$

$$\tau_{ij}(t) = \tau_{ij}(t) * (1 - \rho) \quad (2)$$

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \Delta\tau_{ij}^k \quad (3)$$

In the results section the highest number of nodes for which an optimal solution was found will be shown, however, since the TSP is a factorial complexity problem being O(n!) the biggest graph for which an optimal solution was found is of 14 nodes. For higher amount of nodes different parameters will be tuned to get the best answer as possible and the change of these will be showed. Ten runs of every problem will be done and optimal values, average value and worse value will be shown.

IV. Results

Table I shows 10 runs of the algorithm for graphs of n nodes. The parameters in these runs are: Number of ants: 10, Generations: 1000, α:1.5, ρ:0, β:5, Q:1

	Best Value	Worse Value	Average Value	Average Execution time	Optimal Value
n = 14	23	25	24.4	7.5 seconds	23
n = 13	22	24	23.4	6.4 seconds	22
n = 12	28	31	28.3	4.8 seconds	28
n = 11	19	19	19	4.1 seconds	19
n = 10	28	29	28.4	3.7 seconds	28

TABLE I
Optimal solutions for n number of nodes

Table II shows 10 runs of the algorithm for a graph of 30 nodes. The parameters in these runs are: Number of ants: 30, Generations: 100, α: varying, ρ:varying, β:varying, Q:1 For further visualization figure 1 shows this in a convergence graph for a single run of the program with each variable set.

Best Value	Worse Value	Average Value	α	β	ρ
74	91	86	1	1	0
52	63	57.4	2	1	0
53	78	69.5	5	1	0
63	82	74.2	1	2	0
60	73	67.9	1	5	0
45	65	50.5	1	1	.2
38	50	42.9	1	1	.5
40	49	44.6	5	5	.5
97	112	105.9	.5	.5	0

TABLE II
Variation in solutions caused by parameters

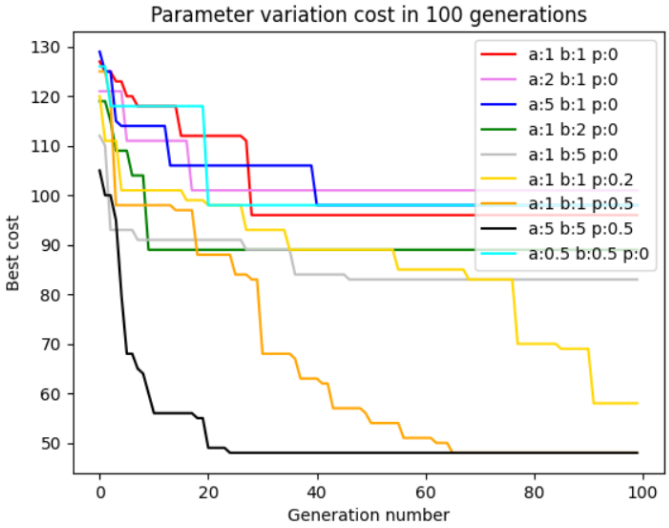

Fig. 1. Convergence graph for variable variance

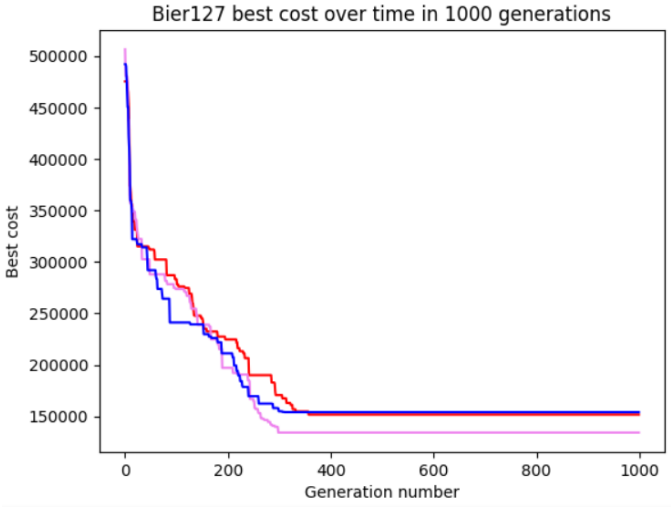

Fig. 2. AS convergence graph for Bier127

In table III a comparison using the problem eil51[1] and problem eil76[2], will be made with other algorithms, namely Biogeography-based optimization (BBO), The Discrete Rat Swarm Optimization (DRSO) and genetic algorithm with jumping gene and heuristic operators (GA-JGHO), all of these algorithm results were found in reference [14] except for Lin-Kernighan which is taken from reference [15].

	Optimal	AS	BBO	DRSO	GA-JGHO	Lin–Kernighan
eil51	426	451	974	432	429	428
eil76	538	613	1492	569	545	546

TABLE III
COMPARISON OF ALGORITHMS IN PROBLEMS EIL51 AND EIL76

Finally using Bier127[3] the best result for ant system was a cost of 134314, giving a convergence graph as shown in figure 2. In comparison Lin-kernighan obtained a cost of 119432 in the best result. The optimal result is 118282. The variables for the AS were: Number of ants: 31, Generations: 1000, α:2, ρ:.7, β:5, Q:100

V. DISCUSSION

Ant system algorithm performed well in most problems, not reaching optimal results after problems of 15 nodes, but getting near. AS was the first kind of Ant colony devised, and as such it has the most weaknesses. Implementations such as the ones mentioned in Background were created to tackle individual problems from AS, namely high number of ants or complex enough problems could overwhelm them. In table III it is easy to see that while not the worst algorithm, the results are not close to the target solution even if they are not considered bad. That table only shows the best result however, countless trials were done with different parameter values.

On the other hand, further study into the ant systems variables does show it has potential to get a better result in theory, while table III was the best result after some variable tinkering, table II shows how much it is possible to affect the result with only varying the variables, with some optimization, maybe even AS is able to get a near optimal result in eil51 or even eil76. However, it might also be the case that the algorithm is hard stuck on a non-optimal solution. A different problem of optimization of these variables exists, but it is outside the scope of this paper.

All this does not mean that the algorithm is useless, for example in table I found the optimal solutions to specific cases of the TSP, while low in number, it comes with a lot of time improvement than a brute force approach, while AS took 7 seconds per execution for 14 nodes, getting the optimal value by brute force took an hour and a half.

Further testing done on other problems[4] revealed that optimality was not reachable after the examples in table I, but not far, for example in ulysses22 and bays29 the error for both did not go over 5%. In figure 2 the convergence graph for Bier127 problem is shown, this problem has 127 nodes. The best result obtained by AS was 134314, giving an approximate error of

[1]problem found at https://github.com/coin-or/jorlib/blob/master/jorlib-core/src/test/resources/tspLib/tsp/eil51.tsp

[2]problem found at https://github.com/coin-or/jorlib/blob/master/jorlib-core/src/test/resources/tspLib/tsp/eil76.tsp

[3]problem found at https://github.com/coin-or/jorlib/blob/master/jorlib-core/src/test/resources/tspLib/tsp/bier127.tsp

[4]Library of instances for TSP https://github.com/coin-or/jorlib/tree/master/jorlib-core/src/test/resources/tspLib/tsp

13%. On the other hand Lin-Kernighan obtained 119432 with an error of .9%.

VI. Conclusions

The algorithm presented here has some room for improvement, however it is already a functional optimization algorithm for the TSP with optimal results for nodes lower than 14, other than that, some of the problems with known solutions proved to be hard for the AS, which approximated results within 10% of the optimal solution, proving it to be capable, but not perfect.

One of the most troublesome things was understanding how the variables affected the results, so thanks to the testing done, it was shown that high values for the variables do make the results better in some cases, while in others they might be a hindrance, specially ρ which worked very well with high number of ants, but in lower numbers it was complicated for ants to keep up with the pheromone evaporation. Furthermore, it was universally true that values of α lower than 1 gave worse results, while higher values varied a lot, on the other hand β always gave better results when increased.

Definitely one of the biggest area of opportunity for this algorithm is to try to optimize the AS variables with a meta-heuristic such as a genetic algorithm or such.

References

[1] M. Dorigo, *Ant Colony optimization*. The MIT Press, 2004.

[2] D. E. Jackson and F. L. W. Ratnieks, "Communication in ants," *Current Biology*, vol. 16, 7 2006.

[3] A. Carbonaro and V. Maniezzo, *The Ant Colony Optimization Paradigm for Combinatorial Optimization*, pp. 539–557. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[4] M. Ünal, A. Ak, V. Topuz, and H. Erdal, *Ant Colony Optimization (ACO)*, pp. 31–35. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[5] O. Sammound *et al.*, "A comparative study of ant colony optimization and reactive search for graph matching problems," *6th European conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP 2006)*, pp. 287–301, 4 2006.

[6] X. Wei, "Parameters analysis for basic ant colony optimization algorithm in tsp," *International Journal of u-and e-service, science and technology*, 2014.

[7] F. Bennis and R. Bhattacharjya, *Nature-Inspired Methods for Metaheuristics Optimization Algorithms and Applications in Science and Engineering: Algorithms and Applications in Science and Engineering*. 01 2020.

[8] M. Dorigo and L. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[9] S. Negulescu, C. Oprean, C. Kifor, and I. Carabulea, "Elitist ant system for route allocation problem," *8th WSEAS International Conference on APPLIED INFORMATICS AND COMMUNICATIONS*, 01 2008.

[10] T. Stutzle and H. Hoos, "Max min ant system," *Future generation computer systems*, vol. 16, pp. 889–914, 200.

[11] C. Gagné, M. Gravel, S. Morin, and W. L. Price, "Impact of the pheromone trail on the performance of aco algorithms for solving the car-sequencing problem," *The Journal of the Operational Research Society*, vol. 59, no. 8, pp. 1077–1090, 2008.

[12] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *European Journal of operational research*, vol. 185, pp. 1155 – 1173, 2006.

[13] H. Ismkhan, "Accelerating the ant colony optimization by smart ants, using genetic operator," *International Journal on Computational Science Applications*, vol. 4, 4 2014.

[14] C.-H. Tsai, Y.-D. Lin, C.-H. Yang, C.-K. Wang, L.-C. Chiang, and P.-J. Chiang, "A biogeography-based optimization with a greedy randomized adaptive search procedure and the 2-opt algorithm for the traveling salesman problem," *Sustainability*, vol. 15, p. 5111, 03 2023.

[15] K. Helsgaun, "An effective implementation of the lin–kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.