A Multi Ant Colony Optimization Approach For The Traveling Salesman Problem

Mathurin Soh, Baudoin Nguimeya Tsofack, Clémentin Tayou Djamegni

▶ **To cite this version:**

Mathurin Soh, Baudoin Nguimeya Tsofack, Clémentin Tayou Djamegni. A Multi Ant Colony Optimization Approach For The Traveling Salesman Problem. Revue Africaine de Recherche en Informatique et Mathématiques Appliquées, A paraître. hal-02926738v1

HAL Id: hal-02926738

https://hal.science/hal-02926738v1

Submitted on 1 Sep 2020 (v1), last revised 6 Jul 2021 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Multi Ant Colony Optimization Approach For The Traveling Salesman Problem

Mathurin Soh, Baudouin Nguimeya Tsofack, Clémentin Tayou Djamegni

Research Unit in Fundamental Informatics, Engineering and Applications
University of Dschang
P.O. Box 67 Dschang, Cameroon
mathurinsoh@gmail.com ; mathurin.soh@univ-dschang.org
nguimeyabaudoin@yahoo.fr
dtayou@gmail.com

ABSTRACT. In this paper, we propose a new approach to solving the Traveling Salesman Problem (TSP), for which no exact algorithm is known that allows to find a solution in polynomial time. The proposed approach is based on optimization by ants. It puts several colonies in competition for improved solutions (in execution time and solution quality) to large TSP instances, and allows to efficiently explore the range of possible solutions. The results of our experiments show that the approach leads to better results compared to other heuristics from the literature, especially in terms of the quality of solutions obtained and execution time.

RÉSUMÉ. Dans ce papier, nous proposons une nouvelle approche de solution approchée au problème de voyageur de commerce (Traveling Salesman Problem in english), dont on ne connait pas d'algorithme exact permettant de trouver une solution en un temps polynomial. L'approche proposée est basée sur l'optimisation par des fourmis. Elle met plusieurs colonies en compétition pour la recherche des solutions améliorées (en temps d'exécution et en qualité de la solution) aux instances de TSP de grandes tailles, et permet d'explorer plus efficacement les plages de solutions possibles. Les résultats issus de nos expériences nous montrent, que l'approche permet de déterminer des résultats meilleurs par rapport aux autres heuristiques (ACS-LKH, et AG-LKH), issues de la littérature, notamment en qualité des solutions obtenues et en temps d'exécution.

KEYWORDS : Ants, Colony, Heuristic, Multi-colony, Traveling Salesman Problem

MOTS-CLÉS : Colonie, Fourmis, Heuristique Multi-colonie, Problème du Voyageur de Commerce

1. Introduction

The Traveling Salesman Problem (TSP) is one of the most important combinatorial optimization problems. Its statement is simple, and yet it remains one of the most difficult problems in operational research because it is considered NP-difficult. TSP is a classic combinatorial optimization problem that has been the subject of studies in the field of operations research and computer science since the beginning of the years 1950s. As a result, a large number of techniques were developed to solve this problem. Much of the work on TSP is only motivated not by practical applications, but rather by the fact that it provides a framework ideal for platforms for the study of general methods that can be applied to a wide range of combinatorial optimization problems. Indeed, many applications TSP guidelines bring research to life and help guide future work. In the TSP, a number of cities are indicated, and are linked to others by arcs. A salesman begins his or her tour from any city, and traveling only once to each city, he or she returns to the city of departure once again. The problem is to find the shortest route of the salesman from a given city, visiting n cities once and then finally returning to the origin city.

Despite the multitude of methods available to solve the basic or applied TSP, none of the methods have so far been successful in resolving large instances[1, 2]. Indeed, one is generally confronted with a combinatorial explosion and a limitation of the memory resources of the computers when the size of the instances increases. However, problems related to TSP are increasing and are frequent in our society and must be resolved. For most instances of TSP, the optimal solution is rarely reached even by the best efficient heuristics developed on this subject. In this work, we are interested in the use of ant colony optimization. It is a question for us of propose another solution to this problem. So we develop a new heuristic involving several colonies of ants.

In the rest of this paper, we present in section 2, the TSP and its applications. Section 3 is devoted to a review of the literature on the resolution of TSP by heuristics. In section 4, we present our contribution. Section 5 is devoted to the experimental results, elucidated from tests and discussions that are performed with TSP instances from the TSPLIB library. Section 6 concludes our work.

2. The Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is defined as follows: given n points (cities) and the distances between points, find a path of minimum total length that passes exactly once through each point and return to where it all started[1, 10]. The distance can also be seen as the cost in general. This combinatorial optimization problem, therefore, consists in searching for the best solution among several possible choices. However, it is easy to state but difficult to solve. The problem is to determine an Hamiltonian circuit, i.e. passing once and only once through the n cities, and that is of minimum cost. It's classified as a NP-difficult problem [1, 2, 3], because there not known method of resolution to obtain exact solutions in reasonable time for large instances (large number of cities) of the problem. For these large instances, one is very often satisfied with the approximate solutions because after an explicit enumeration, the number of Hamiltonian paths is equal to (n-1)!/2 [10].

Mathematically, the TSP can be formulated as follows: Let n cities and C_{ij}, the cost or distance corresponding to $i \rightarrow j$ Let the variable X_{ij}, which equals 1 if the tour contains the arc $i \rightarrow j$, and 0 otherwise. The problem is stated as follows[6]:

$$\begin{cases} MinZ = \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} X_{ij} \\ \sum_{J=1}^{n} X_{ij} = 1 \quad \forall i & (1) \\ \sum_{i=1}^{n} X_{ij} = 1 \quad \forall j & (2) \quad [1] \\ \sum_{i \in Q}^{n} \sum_{j \in Q}^{n} X_{ij} \geq 1 \quad \forall Q & (3) \\ X_{ij} \in \{0, 1\} \forall i, \forall j \end{cases}$$

Where Q represents a subset of 1,..,n and its complement. The constraints (3) express that the permutation of the n cities must be a trick, i.e. it can't not to exist subtours.

TSP has applications in transportation, networks and logistics. For example, finding the shortest route for pickup buses... school or in industry, for collection/distribution, to find the shortest distance that the mechanical arm of a machine will have to travel to drill holes in a printed circuit board[6, 7, 10].

3. Problem position

Existing algorithms for solving TSP are not efficient enough. When the number of cities increases, the time or space required to solve the problem increases exponentially. The ant colony system solves the problem within a reasonable period of time, and produces optimal solutions [2]. But when the size of the problem increases, it takes a long time to produce the result, which is often not optimal. Thus, a modification is necessary to solve the larger problem of the traveling salesman.

In this work, we propose an approach to solve the TSP based on Ant Colony Systems (ACS) heuristic, using several colonies in competition to find competitive results capable of minimizing known execution times or of giving optimal or almost optimal solutions to instances where a solution is still difficult to achieve with current methods.

This new approach would allow more effective exploration of the beaches of possible solutions, and determine satisfactory iteration techniques for build the path to the optimal solution as quickly as possible out of thousands of possible paths. We are particularly interested in the multi-colony approach for three reasons: First, we wish to offer greater exploration in terms of the possibility of ant deployment. Secondly, we want to obtain a multiplicity of potential solutions to the problem during the optimization process. Thirdly, we want to get good solutions more quickly, because of the competition at the colony level.

4. Review of art

Several approaches have been used to address the TSP. From exact to approximate methods. The simplest approach is the exact methods approach, which consists of listing all possible solutions and taking the best solution. These exact approaches are effective only for small instances where the explicit enumeration of solutions is possible in reasonable time [2, 7, 8]. For large instances, most of the work is based on heuristic and meta-heuristic methods. In the literature, there are two types of heuristics: Those that build a tour from scratch and those that are effective at improving an existing tour called Tour Enhancement Heuristics. Research into heuristic methods for solving TSP has included Dorigo's results. et al [2, 6] who, inspired by the collective behaviour of ants and their similarities with TSP, proposed the first ant colony algorithm for TSP: "'*the Ant System (AS)*". Unfortunately, the purely probabilistic choice of the AS makes it less efficient in terms of diversification, thus favoring the risks of premature convergence. To improve the performance of AS on large problems, Dorigo et al [2], modify the AS transition rule to provide a better balance between exploring new areas of the solution space and exploiting the accumulated knowledge of the problem to intensify the solution. To this end, they introduce a new rule that depends on a $q_0(0 < q_0 < 1)$ parameter. Q is a randomly generated number, evenly distributed between $[0.1]$. With this new version, the choice of the next cities is not only probabilistic but also according to a q parameter. The resulting algorithm is called ACO [3]. Only ACO has shown some weaknesses in terms of to explore new avenues and intensify solutions.

Beside these so-called construction heuristics, we have improvement heuristics such as the K-opt Heuristic $(K = 2, 3, ...)$ which improves solutions by testing whether permutations of k paths produce more optimal rounds [1] or the heuristic of Lin and Kernighan (LK) [3] which is an improvement of the K-opt heuristic with k variables, or the Lin and Kernighan heuristic modified by Helsgaun (LKH). These heuristics efficiently improve solutions starting from an already known solution. Another class of heuristics is hybrid heuristics, which has become increasingly popular in recent years. These hybrid heuristics can be divided into several groups. On one side we have the hybridization of construction heuristics. between them, which consists in replacing a mechanism of one heuristic by a mechanism of another heuristic. to make up for what is limited in one by what is seen as an advantage in the other. In this regard, we can quote hybridization (AG-ACO) [4, 10]. On the other hand, we have the hybridization of construction heuristics (AG, ACO ...) with the heuristics of the researches. local or heuristic enhancement (LK, LKH, RL). An example of this technique was the one proposed by [10]. This work proposes for the resolution of the TSP, two new hybrid heuristics named "AG-LK" and "ACS-LK" resulting respectively from the hybridization of a Genetic Algorithm (AG) and the heuristic (LK), and the second between the ant colony algorithm (ACS) and the heuristic (LK). However, the authors use only one colony. The results obtained from this experiment have sufficiently demonstrated the efficiency of these hybrid approaches on several instances of TSP [10] [3]. Unfortunately, the resolution time resulting from this experiment remains enormous.

To improve the efficiency of the hybrid heuristics proposed in [4], Nguimeya and al in 2016 implemented two hybrid heuristics for TSP. The first one between a Genetic Algorithm (AG) and the heuristic (LKH) which is an improvement of LK by Helsgaun [3] and the second one between the Ant Colony System algorithm (ACS) and the heuristic (LKH). The resulting heuristics were called respectively "AG-LKH" and "ACS-LKH".

Hybridization strategies differ from one author to another [4].

As we can see, the search for solutions to TSP is a permanent quest in the field of computer science, especially in operational research. Several authors have already worked in this direction by approaching various approaches, as this art review clearly shows. Only none of these methods has been able to overcome TSP. In the following section, we propose a new hybrid heuristic for TSP.

5. Proposition of a Multi-Colony Ant Optimization Approach for TSP

5.1. Basic idea

To achieve the above objectives, we present the basic ACO proposed by Dorigo et al in [2]. We then modify this algorithm using multiple colonies. This leads us to a new version that we call MACO (ACO-multi colonies). MACO proposes a new way to iterate ants from different colonies by defining more or less one iteration step to optimize the decision of the artificial ants to take preferably the right paths while avoiding the pitfalls of the local optimum. Strategies exchanges or communication between ants during their movements will be done in real time in order to help ants detect failures or wrong paths as quickly as possible. The updating of pheromones by ants is done as in the ACO [2]. The resulting algorithm is implemented in C language and tested on TSP instances from the TSPLIB library [5].

5.2. Description or Mechanism of ACO Approaches

. For the study, we consider that our problem is modeled in the form of a graph. Each node of the graph represents a city or task to be performed, d_{ij} the distance between cities i and j, and the couple (i, j) the path between these two cities. We will also consider the following:

– The ant represents a solution to the problem at hand;

– Computer pheromones are values associated with found solutions. These values depend on the quality of the solutions.

5.2.1. Step 1: Build a path for each ant (Solution)

To move from one node (city) of the graph to another, the iteration and movement of the ants is as follows: Initially (at time $t = 0$), the algorithm sets m ants to n cities and the intensity of the trace for all city pairs (i, j) is set to a small positive To value in the pheromone matrix. A taboo list is maintained to ensure that a city cannot be visited twice during the same tour. Each ant k will therefore have its own list of Vk-tab cities that will keep in memory the cities already visited. During an iteration of the algorithm, several ants take turns visiting a sequence of cities. One cycle (NC) is finished by the time the last of the m ants have finished building.

At t $\neq$0, from a sequence of cities already visited, each ant k chooses the next city to add to its list Vk-taboo using a probabilistic rule so the formula is given in equation (1).

$$\eta_{ij} = \frac{1}{d_i j}$$

$$p_{ij}^{k} = \frac{[\eta_{ij}]^{\beta}[\tau_{ij}(t)]^{\alpha}}{\Sigma u \in v_k [\eta_{iu}]^{\beta}[\tau_{iu}(t)]^{\alpha}} \quad (1)$$

$$\tau_{ij}(t+1) = \rho \, P_{ij}^{k}(t) + \triangle \, \tau_{ij} \quad (2)$$

[2]

It's based on a compromise between visibility ($\frac{1}{d_{ij}}$) and the amount of pheromone (t_{ij}) present between i and j. The parameters α and β are used to determine the relative importance of track intensity and visibility, respectively in the construction of a solution. This mode of iteration allows to privilege the shortest paths since ants will need fewer iterations to get to the end.

5.2.2. Step 2: Pheromone deposition and solution evaluation

When an ant moves from city i to city j, it leaves a certain amount of pheromone (value) on the arc (i, j). A so-called pheromone matrix records information about the use of the (i, j) arc. At each stage of the round, the matrix is updated so that the greater the use of the (i, j) arc in the past, the more pheromone it has been used, the higher the probability that these arcs will be used again in the future. In this version of ACO, it says, there is an overall update of the pheromone trace intensity after each transition based on the assessment solutions found. For example, an ant moving from node i to j will be able to signal the following ants that are arriving in i if the displacement in j is a justified displacement. Equation (2) allows the pheromones to be updated.

$$\tau_{ij}(t+1) = \rho \, \tau_{ij}(t) + \triangle \, \tau_{ij}$$

[3]

5.2.3. Ant Colony Optimization algorithm (ACO)

The Ant Colony Optimization algorithm is stated in algorithm 5.1

5.3. Multi-colony ant colony optimization approach for TSP

In this section, we present an approach called Multi Ant Colony Optimization for the TSP (MACO), based on colony optimization of the [2] ants between multiple colonies. To do so, we first present some characteristics and principles of ACO approaches, then we approach the description of our MACO resolution approach and finally we present the MACO algorithm.

We put several colonies of artificial ants in competition with each other depending on the size of the problem to win in terms of exploring solutions. In this way, the different colonies build their tour independently and collaborate through a kind of shared memory (global pheromone matrix) by stimulating a pseudo-parallelism to reach the common goal. Indeed, each colony is placed in a departure city, a travel step is authorized for each colony in turn. If an ant belonging to a colony during its iteration finds a better solution than the one known by the other ants in the different colonies, it quickly informs the others by updating the solution through the global communication matrix. To optimize the solution time, information about the pheromone matrix is given in real time immediately after each transition and not delayed.

Algorithme 5.1 : ACO Algorithm

```
Begin
    Inputs m : number of ants per colony ;
    n : number of cities ;
    N c ← 0 ;
    Initialize Taboolist // with the origin town of each ant ;
    Initialize the matrix τij While ( Nc < Nmax) and ( convergence not reached)
     Do
        For i ← 1 to n Do
            For j ← 1 to m Do
                Select the city j to be added to the current tour with the formula p
                  k
                  ij (t) ;
                Update locally the trace according to cities(i,j) ;
            EndFor
        EndFor
    EndWhile
    For each ant k Do
        Evaluate the solution k for each of steps ;
        Insert the solution k in the TabooList ;
        Update globally the trace according to the best solution of the cycle ;
    EndFor
    NC ← NC + 1;
End
```

5.4. New Multi-Colony ACO Approach (MACO)

5.4.1. Description of the approach MACO

Our approach called MACO is based on ACO approaches with the difference that MACO uses several ant colonies to allow a better exploration of the paths and optimizes the solutions found by another LKH improvement heuristic. Indeed, one of the major flaws in single colony [2] ACO is the difficulty for the colony to be able to explore efficiently the solution ranges especially for large instances. With MACO, we intend to solve the problem using several colonies. The different colonies of the MACO algorithm are independent and build their solutions independently. while working closely together to achieve the final goal through communication mechanisms made possible by a communication matrix and values called pheromone.

Each colony (the ants) represents a solution to the problem being treated. The pheromone matrix is common to all colonies. It is used as a shared memory to allow the ants to communicate with each other to guide each other's search. This matrix is dynamic and is used by the colonies to incrementally build solutions. Each ant has a simple behavior, but the exchanges between ants are made by the intermediary of of the pheromone matrix, allow the gradual production of a solution that is close, if not optimal. Ants communicate indirectly via dynamic modifications of the pheromone tracks and thus build a solution to a problem, based on their collective experiences.

5.5. Description of the principle of the MACO algorithm

Step 1: Initializing the algorithm: Initially, there's no pheromone on the arcs. The global pheromone matrix is initialized with 0 values for each pair of nodes. To initialize

this matrix for the first time, the ants move randomly and in parallel per colony and build their first round. Each colony has its own local communication matrix. Once all the points have been visited, the ants return to their starting point (city). At the end of this first stage, the ants modify the global communication matrix by updating its entries with the best colony found (the best solution at the end of a tour).

Step 2: Construction of solutions: After the initialization phase, from this sequence of cities already visited, within the different colonies, the ants move this time on the different nodes of the graph according to a probability and therefore the equation is that of formula (3). It allows the colonies to favour the shortest paths during the different iterations. During a tour, each ant k of a colony L records in its taboo list (memory) the list of cities already visited. The new probabilistic formula for selecting a node by the ant k of colony L is defined by the expression $p_{ij}^{k,l}$ (3). This probability is modeled on the one used in the ACO heuristic [2] described above.

$$p_{ij}^{k,l}(t) = \begin{cases} \frac{[\eta_{ij}]^{\beta}[\tau_{ij}(t)]^{\alpha}}{\sum u \in v_{k_l}[\eta_{iu}]^{\beta}[\tau_{iu}(t)]^{\alpha}} & \text{if j} \in v_{k_l} \\ 0 \quad \text{else} \end{cases} \quad (3)$$

[4]

where v_{k_l} is the set of nodes (cities) not visited by ants in colony L.

Description of the Pheromone Matrix: It's an array of i rows and j columns. At $t = 0$ each entry of the matrix is initialized with the value 0. $\tau_{ij}(t)$ represents the intensity of the pheromones on the (i, j) edge at the t cycle. During the construction of a path, after each transition of an ant k belonging to a colony L, ants from different colonies use directly the global communication matrix to communicate and immediately update pheromones as they evolve. The pheromone matrix evolution equation is described in equation (4) by :

$$\tau_{ij}(t+1) = \rho\, \tau_{ij}(t) + \triangle\, \tau_{ij} \quad (4)$$

[5]

Equation (4) is the same as equation (2) in ACO [2]. The main purpose of updating after each transition is to simulate a kind of parallelism in order to save computing time by giving the information on the global communication matrix in real time and not in delayed time. As in ACO, to reduce failures, pheromones evaporate[2]. The pheromones have to be reduced (value) on each path depending on the quality of the solution. This allows the ants after a certain time to avoid taking certain paths that can lead to failure. This characteristic is modeled via the parameter ρ ($0 < \rho < 1$) . The evaporation equation of the pheromone matrix is described by the formula (5).

$$\tau_{ij}(t+1) = (1 - \rho)\, \tau_{ij}(t) \quad (5)$$

[6]

The inverse of the distance between cities η_{ij} = $\frac{1}{C_ij}$ called visibility is a static information used to guide ants to nearby cities and avoid cities that are too far away. The parameters α and β are used to determine the relative importance of the ant's presence in the area. the intensity of the trace and the visibility in the construction of a solution. This is based on a trade-off between visibility (η_{ij}) and the amount of pheromone (t_{ij}) present between i and j at cycle t. These parameters are the same as those used in ACO[2]. Each colony corresponds to a potential solution to the problem, i.e. a complete tour. When recovering a solution, each ant in the different colonies provides part of the solution to the problem. The assembly of the different sub-solutions provides the solution to the original problem. If at a given time t all the ants in a colony have the same solution, the iterations are stopped. The procedure finally obtained for optimization by multi-colony ants is the algorithm (5.2).

5.6. Complexity of MACO Algorithm

The study of the complexity of our MACO algorithm is done by subdividing the algorithm into 4 sections to calculate the complexity.

- Initialization: Since we have assumed a total interconnection between cities, we have a complexity of O (n^2 + m. L).

-Constructions of the path, and progressive deposition of pheromones plus evaluation of solutions: Complexity is O (n^2 . m. L).

Pheromone Evaporation: Complexity is O (n^2).

- Evaluating paths after each transition: Complexity O (n.m.L). For the L colonies, we compare the towers of m ants in each colony. Each tower has a length of n elements.

In summary, we obtain the global complexity by adding the complexity of step 1, to the product of the total number of rounds (i.e. NCmax) by the global complexity of steps 2 to 4. This gives O (n^2 + m. L + NCmax . n^2 . m.L).
or finally **O(NCmax . n^2 . m.L)**.

6. Experimental Results - Discussion and Interpretation

In order to demonstrate the performance of the approach we propose, we perform computer simulations on a machine with the following characteristics: Intel Dual core, 2Ghz processor, 2Gb RAM, Kali Lunix operating system. In all the experiments in this section, we let us define as starting parameters in a first time $m = 100$ and $L = 10$, then in a second time $m = 100$ and $L = 50$. These experiments are carried out on the reference problems Lin105, Pr124, LIN318, att532, ALi535, rat783, std1655, Vm1748, pr2392, Usa13509 and pla33810 all from the TSPLIB[5] library with publicly available TSP instances.

Under the same conditions, we subjected them to our heuristics and compared the results obtained with the best current heuristics for TSP at the time of testing. Several simulations (100 in total) allowed us to assess the relevance and effectiveness of our approach. By making a comparative study between the MACO heuristic with the basic AS heuristic and ACS obtained by Dorigo [2] as well as ACS-LK one of the best heuristics for the TSP from the literature proposed in [10] or those obtained in [4] (AG -LKH, ACS -LKH) and under identical test conditions we obtain the results of Figures 1, 2, 3 and 4 respectively.

AS				
INSTANCES OF TSP	Size	best soluiton	Time	success/100
etl51	51	252,46	/	/
st70	70	675	/	/
LIN 105	105	ND	/	/
Pr124	124	ND	/	/
LIN318	318	ND	/	/
Attr532	532	ND	/	/
attr535	535	ND	/	/
Rat783	783	ND	/	/
std1655	1655	ND	/	/
Vm1748	1748	ND	/	/
pr2392	2392	ND	/	/
Usa13509	13509	ND	/	/
Pla33810	33810	ND	/	/

Figure 1. *Results AS*

Instances of TSP	size	Initial_ACS		
		best solution	Time	success/100
d198	198	585000	/	15
pcb442	442	595000	/	51
Attr532	532	830858	/	28
Rat783	783	991276	/	9
fl 1577	1577	942000	/	22

Figure 2. *Results ACS-DORIGO [2]*

MACO					ACS_LK				
Instances of TSP	Size	best soluiton	Time	success/100	Instances	Size	best soluiton	Time	success /100
LIN 105	105	14379	0,03	100	Lin 105	105	14379	0,03	100
Pr124	124	58537	0,03	100	Pr124	124	58537	0,92	100
LIN318	318	42029	0,34	100	Lin318	318	42029	0,8	100
Attr532	532	276787,7	5,3	100	Att532	532	276787.7	13,26	100
attr535	535	202339	1,94	100	Ali535	535	202339	30,94	100
Rat783	783	88060	0,28	100	Rat783	783	88060	0,28	100
std1655	1655	6218,6	72,18	100	std1655	1655	62128.6	73,28	100
Vm1748	1748	336557	41,86	100	Vm1748	1748	336557	71,86	100
pr2392	2392	378032,2	0,97	100	Pr2392	2392	378034.25	1,19	90
Usa13509	13509	19849706	21103,3	100	Usa13509	13509	19984330	1864,74	84
Pla33810	33810	ID	/	./	Pla33810	33810	ID	/	/

Figure 3. *Comparison of MACO algorithm with ACS-LK hybrid algorithm*

Algorithme 5.2 : MACO Algorithm

```
Begin
    Input m : number of ants per colony ;
    L: Number of colonies;
    n :number of cities;
    N_c ← 0 ;
    D_ij ← 0 // Global matrix ;
    d_ij ←t_0 // local matrix of pheromones ;
    Initialize Taboolist // with each ant's home city;
    // The initialization of the TabooList and the Dij matrix ;
    Place each colony randomly in a starting point (city) ;
    // Parallel construction of the tours by the different colonies;
    For K ← 1 to m Do
        Construction of a tour by each ant at random;
        Gradual deposition of pheromones in the dij matrix of each colony;
        Evaluation and selection of the best colony;
        Initializing the Dij matrix with the best colony;
    EndFor
    // Construction of optimal solution ;
    The colonies are positioned in a departure city;
    While ( N_c < N_max) Do
        For i ← 1 to n Do
            For j ← 1 to L Do
                For K ← 1 to m Do
                    Select the city V_i to be added to the current tour with formula
                      (1);
                    Evaluate the solution of the ant K on route (i,j) ;
                    Perform the global update of the trace according to city pair
                      (i,j) if it's better than the previous ants according to formula
                      5.2 and 5.3 (evaporation) ;
                    Insert as you go (i,j) in the taboo list so as to construct
                      gradually solution K;
                EndFor
            EndFor
        EndFor
    EndWhile
    S ← Best solution : Each colony provides a partial solution ;
End
```

In the results presented in figure 6, the green lines represent the cases where the gains observed with MACO are related to costs and turnaround times. The brown lines symbolize the cases where the observed gain is only at the level of execution time, the optimal solution having already been reached. We note a better execution time for MACO compared to those obtained so far with the best algorithms (ACS-LK, AG-LK, ACS-LKH, ACS) at the time of the tests. We also obtain a better and always optimal cost compared to the best solutions known so far with deviations ranging from 0.1 to nearly 1000 distance units on some reasonably sized instances (Usa13506, Pr2392...). The numbers in brackets represent the execution times.

		ACS_LKH			AG_LKH			MACO		
Instances of TSP	Size	best soluiton	Time	success/ 100	best soluiton	Time	success/ 100	best soluiton	Time	success/ 100
LIN 105	105	14379	0,03	100	14379	0,03	100	14379	0,03	100
Pr124	124	58537	0,92	100	58537	0,46	100	58537	0,03	100
LIN318	318	42029	0,8	100	42097,4	1,6	100	42029	0,34	100
Attr532	532	276787,7	15,62	100	27691	13,74	100	276787,7	13,26	100
attr535	535	202339	30,94	100	202339	38,26	100	202339	1,94	100
Rat783	783	88060	0,28	100	88060	0,32	100	88060	0,28	100
std1655	1655	62128,6	73,28	100	63128	151,5	100	6218,6	72,18	100
Vm1748	1748	336557	71,86	100	336557	111,07	100	336557	41,86	100
pr2392	2392	378034,3	1,19	90	378034,8	1114,1	80	378032,2	0,97	100
Usa13509	13509	19984330	2113,3	100	ID	ID	ID	19849706	21103,3	100
Pla33810	33810	ID	/	/	ID	/	/	ID	/	./

Figure 4. *Comparison MACO algorithm with ACS-LKH and AG-LKH algorithms*

Figures 5 and 6 present a comparison of the performance of our approach compared to the improved AG -LKH and ACS -LKH approaches, using the above-mentioned reference problems and terms of solution quality and runtime respectively.

The results of our experiments clearly show the superiority of our algorithm over others, especially in terms of solutions quality and in terms of time taken to reach this optimal solution. However, for some of the instances for which we do not yet know a solution, our approach could not give an optimal solution.

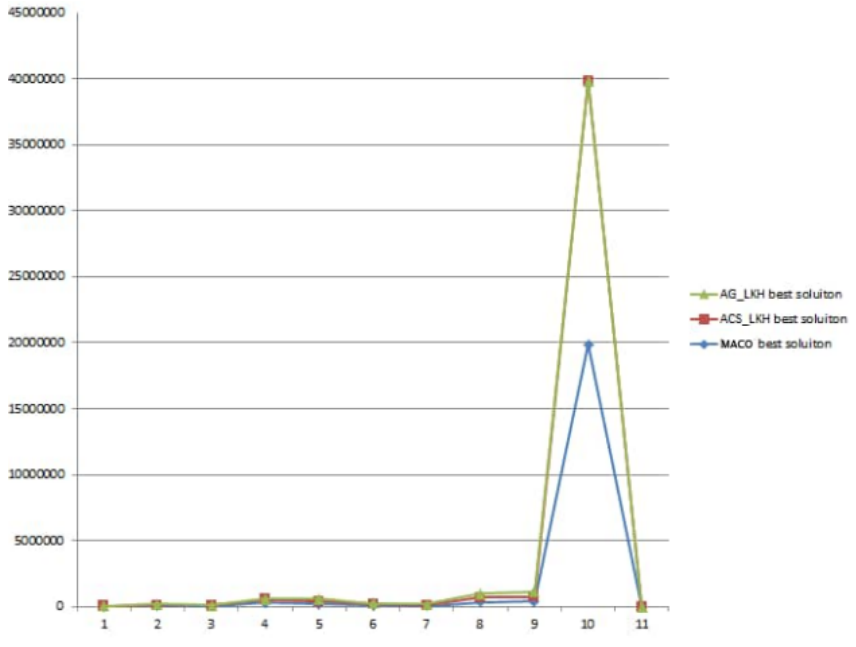

Figure 5. *Comparison MACO, AG-LKH, ACS-LKH in terms of solution quality*

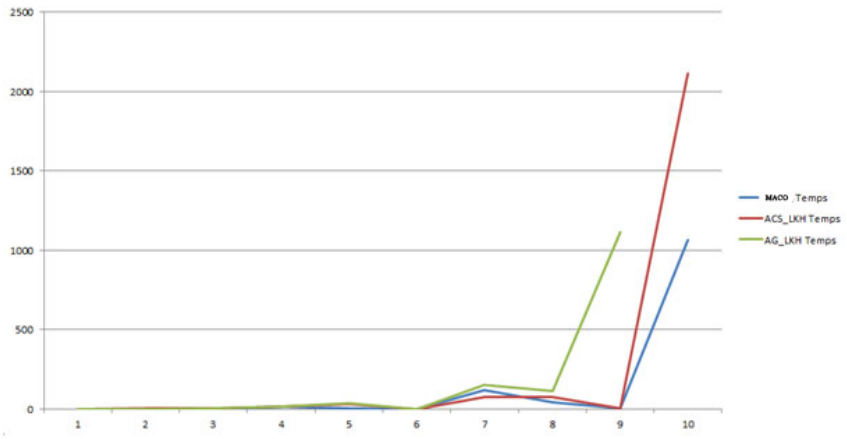

Figure 6. *Comparison of MACO, ACS-LKH, AG-LKH running times*

7. Conclusion and perspectives

In this work, we proposed a new multi-colony ant heuristic (MACO) to improve the search for optimal solutions to TSP and in optimal time. The proposed MACO algorithm is based on the use of several ant colonies. It shows clear improvements in both cost and execution time, proving that it is much more competitive than the best time heuristics for TSP. A limitation related to our multi-colony approach will probably be the increase in execution time needed to find a solution in some cases, or the use of much more resources (memory and processor). We believe that parallelization of the ACS [2] or MACO algorithm would be a considerable contribution to metaheuristics to reduce TSP resolution times.

8. References

[1] C. REGO, D. GAMBOA, F. GLOVER, C. OSTERMAN, "Traveling salesman problem heuristics: Leading methods, implementations and latest advances ", *European Journal of Operational Research* , vol. 211, pp 427-441, 2011.

[2] M. DORIGO, T. STUETZLE, "Ant Colony Optimization: Overview and Recent Advances", *IRIDIA Technical Report Series Technical Report* , vol. No.TR/IRIDIA/2009-013), num. 34 pages , 2009.

[3] K. HELSGAUN, "An effective implementation of the Lin-Kernighan traveling salesman heuristic", *AIEEE Transactions on Evolutionary Computation*, *Ochoa S.F., Roman GC. (eds)* vol. no 12, num. pp 106-130, 2000.

[4] B. NGUIMEYA TSOFACK, M. SOH, L.P. FOTSO, "Algorithmes hybrides pour la résolution du problème du voyageur de commerce", *Proceedings of CARI 2016*, *Ochoa S.F., Roman GC. (eds)* vol. , num. pp 63-74, 2016.

[5] TSPLIB95, "Traveling Salesman Problem Library ", *http://www.iwr.uniheidelberg.de/iwr/comopt/software/TSPLIB95*, Last visit:December 23,2019.

[6] T. STUETZLE, “The Traveling Salesman Problem: State of the Art”, *TUD SAP AG Worshop on Vehicle Routing*, *Ochoa S.F., Roman GC. (eds)* vol. 12, num. july 10, 2003.

[7] BP. DELISLE, “Parallélisation d’un algorithme d’optimisation par Colonies de Fourmis pour la résolution dun problème d’ordonnancement industriel”, *IEEE*, vol. 12, num. pp 5366, 2002

[8] C. CHAUHAN , R. GUPTA , K. PATHAK, “Survey of Methods of Solving TSP along with its Implementation using Dynamic Programming Approach”, *International Journal of Computer Applications (0975 8887)* vol. 52, num. 4, pp 12-19, 2012.

[9] I. ALAYA, “Optimisation multi-objectif par colonies de fourmis Cas des problèmes de sac à dos”, *PhD Thesis, Université de la Manouba*, vol. num. , 2009.

[10] B. TADUNFOCK TETI, L.P. FOTSO, “Heuristiques du problème du voyageur de commerce”, *Proceedings of CARI 2006*, *Fowler, D. and Dawson, L. (eds.)* vol. num. pp 1-8, 2006.