A Hybrid Algorithm Based on Multi-colony Ant Optimization and Lin-Kernighan for solving the Traveling Salesman Problem

Mathurin Soh, Baudoin Nguimeya Tsofack, Clémentin Tayou Djamegni

ARIMA

épisciences

A Hybrid Algorithm Based on Multi-colony Ant Optimization and Lin-Kernighan for solving the Traveling Salesman Problem

Mathurin Soh[*1], **Baudoin Nguimeya Tsofack**[1], **Clémentin Tayou Djamegni**[1,2]

[1]URIFIA, Department of Mathematics and Computer Science, Faculty of Science, University of Dschang, Cameroon
[2]Department of Computer Science, Fotso Victor Institute of Technology, University of Dschang, Cameroon

*E-mail : mathurinsoh@gmail.com

DOI : 6011a5016b8a1
Submitted on XXX 2021 - Published on XXX 2021
Volume : **35** - Year : **2021**
Special Issue : **Data Intelligibility, Business Intelligence and Semantic Web**
Editors : *Ghislain Atemezing, MONDECA, Gaoussou Camara,Idrissa Sarr,Bruce Watson*

Abstract

In this article, a hybrid heuristic algorithm is proposed to solve the Traveling Salesman Problem (TSP). This algorithm combines two main metaheuristics: optimization of multi-colony ant colonies (MACO) and Lin-Kernighan-Helsgaun (LKH). The proposed hybrid approach (MACO-LKH) is a so-called insertion and relay hybridization. It brings two major innovations: The first consists in replacing the static visibility function used in the MACO heuristic by the dynamic visibility function used in LKH. This has the consequence of avoiding long paths and favoring the choice of the shortest paths more quickly. Hence the term insertion hybridization. The second innovation consists in modifying the pheromone update strategy of MACO by that of the dynamic λ-opt mechanisms of LKH in order to optimize the solutions generated and save in execution time, hence the relay hybridization. The significance of the hybridization, is examined and validated on benchmark instances including small, medium, and large instance problems taken from the TSPlib library. The results are compared to four other state-of-the-art metaheuristic approaches. It results in that they are significantly outperformed by the proposed algorithm in terms of the quality of solutions obtained and execution time.

Keywords

Heuristic, Hybridization, multi ant colony , Traveling Salesman Problem.

I INTRODUCTION

The field of distribution or collection of goods, logistics face optimization problems like the famous Traveling Salesman Problem (TSP). The TSP is a problem in combinatorial optimization studied in Operations Research, computational mathematics, and artificial intelligence. The Traveling Salesman Problem particularly attracts the attention of many researchers in recent years. Indeed, the search for exact or optimal cost to TSP remains a challenge for scientific

community. To date, we note that several works have already been subject of intense research related to the TSP. Among these works, we note the efficiency of heuristics and metaheuristics for the resolution of this combinatorial optimization problem and other NP-difficult problems. In litterature, we find heuristics and metaheuristics such as Ants Colony Optitmization (ACO) which are capable of solving small instances of TSP. Despite these advances, the TSP remains difficult to solve when the size of the instances increase. It is a trend to combine ACO with other algorithms to solve very large scale of the traveling salesman problem. With the concept of hybridization that we adopt for this work, the exploitation of several resolution techniques is a new opportunity offered to researchers in the field. In this paper, we present a new hybrid model of ACO with multiple colonies by LKH. Our aims is to use the insertion and parallel hybridization that we explain below, to hybridize the ACO heuristic with the LKH heuristic in order to decrease the computation time of the sequential MACO heuristic and to a certain extent improve the quality of the solutions obtained in the TSP resolution. We opted for insertion hybridization because it is new and efficient. The rest of the work is organized as follows: Section I provides a non-exhaustive state of the art of TSP resolution methods. We also briefly give a formulation of the traveling salesman problem. Section II presents in a detailed manner, the meta-heuristics used in the paper framework. Section III is devoted to the study of hybrid methods. It gives an overview of the different types of hybridization and emphasizes above all the type of hybridization we used: insertion hybridization and parallel hybridization. And finally section IV presents the hybrid approach MACO-LKH the tests and the results obtained. We conclude our work by summarizing the main results obtained and giving new perspectives on the basis of the work carried out.

II PROBLEM STATEMENT

The Traveling Salesman Problem (TSP) is defined as follows: given n points (city) and the distances between each point, find a path of minimum total length that passes exactly once through each point and returns to the starting point[**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[**10**]. The problem is to determine a turn or Hamiltonian circuit, *i.e.* passing once and only once through the n city, and that is of minimum cost. It is classified as a NP-difficult problem [**1**, **2**], because there is no known method of resolution that provides exact solutions in reasonable time for large instances (large number of city) of the problem.

2.1 Mathematical formulation of the TSP

Mathematically, the Traveling Salesman Problem can be formulated as follows: Let n city and C_{ij}, the cost or distance corresponding to the trip $i \rightarrow j$. Let the variable X_{ij}, which is 1 if the

tour contains the trip $i \rightarrow j$, and 0 otherwise. The problem is written as [**6**]:

$$\begin{cases} MinZ = \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} X_{ij} \\ \quad \sum_{J=1}^{n} X_{ij} = 1 \quad \forall i \\ \quad \sum_{i=1}^{n} X_{ij} = 1 \ \ \forall j \\ \sum_{i \in Q}^{n} \sum_{j \in Q}^{n} X_{ij} \geq 1 \ \ \forall Q \\ \quad X_{ij} \in \{0, 1\} \forall i, \forall j \end{cases} \tag{1}$$

(1) Each vertex i can only accept one outgoing arc towards another top j.

(2) Each vertex j can only accept one arc entering from another vertex i.

(3) In any subset Q of vertices, select | Q | bows would allow to form Hamiltonian circuits.

2.2 TSP Applications

Globally, the TSP provides an example of a study of an NP-complete problem whose solution methods can be applied to other discrete mathematics problems. It has several real life applications even in its purest formulation, such as planning, manufacture of microchips, transportation, networks, computer wiring, vehicle routing, job-shop scheduling and logistics. For example, to find the shortest route for school buses or, in industry, to find the shortest distance that the mechanical arm of a machine must travel to drill holes in a printed circuit board.

III STATE OF THE ART

The problem has since attracted many researchers and thousands of different approaches and algorithms have been developed. In this section, we will focus on hybrid methods. Hybridization is a trend observed in much work done on metaheuristics in recent years to solve TSP. It allows to take advantage of the one to fill what is seen as a limit in the other. In literature there are two classes of hybridizations: on the one hand we have hybridization between metaheuristics and exact methods. This is the case of Cotta[**14**] which proposes a hybridization between a genetic algorithm and the exact Branch and Bound method to replace the recombination operator. Jahira[**17**] proposes a hybrid algorithm between the genetic algorithm and the Branch and Bound method to solve the traveling salesman problem.

In the same vein, Chabrier et al [**16**] hybridized a local search with a Branch and Price algorithm to solve the problem of vehicle routes[**14**]. The execution of algorithms is carried out in parallel while keeping communication between the methods. Mavrovouniotis, Muller, and Yang [**23**] integrated the memetic ACO algorithm with local search operators to improve solutions in the population. They apply these local search operators to the best solution found in the population in order to possibly improve this solution; A similar idea is used in this article, but instead of local search operators, LKH is used.

On the other hand, we have the hybridization between metaheuristics and metaheuristics. Among the works carried out in this direction, we have that of Martin and Otto [**17**] who inserted the descent method in a simulated annealing algorithm to solve the traveling salesman problem. This type of hybridization is referred to as low-level relay hybridization in which another algorithm is incorporated to form a new algorithm. Stûtzle and Hoos [**18**] incorporate a local search function in an ant colony algorithm to solve the traveling salesman problem. This low-level co-evolutionary hybridization consists of incorporating one or more single solution-based metaheuristics into a solution population metaheuristic. The advantage of this type of hybridization is that it compensates for the operating power of a local search and that exploration of a global search. Fotso Laure et al in 2008 [**3**] propose two new hybrid heuristics for the TSP. The first between a Genetic (AG) and heuristic (LK) algorithm and the second between the ant colony algorithm (ACS) and heuristics (LK). The heuristics obtained were called respectively AG-LK and ACS-LK [**1**]. These authors use a single colony. The results of this experiment have sufficiently demonstrated the effectiveness of these hybrid approaches on several instances of TSP [**3**]. Unfortunately the solution time resulting from this experience remains enormous. To improve the efficiency of the hybrid heuristics proposed by Fotso et al, Nguimeya, et al in 2016 implemented two new hybrid heuristics for the TSP. The first between a Genetic algorithm (AG) and heuristics (LKH) which is an improvement of LK by helsgaun [**2**] and the second between the ant colony algorithm (ACS) and heuristics (LKH). The heuristics obtained were called respectively "AG-LKH" and "ACS-LKH". The hybridization strategies differ from one author to another [**1**].

IV PRESENTATION OF SOME HEURISTICS AND META-HEURISTICS USED TO SOLVE THE TSP

Metaheuristics are a family of stochastic methods which consist in solving optimization problems. One of the advantages of these is their ability to optimize a problem from a minimum amount of information, however they do not offer any guarantee as to the optimality of the best solution found. Metaheuristics are gaining more and more popularity and are constantly evolving. As a result, we can note a large number of metaheuristics that currently exist.

4.1 Ant colony algorithm

This metaheuristic is inspired by collective depositing and tracking behaviors observed in ant colonies [**20**]. In fact, ants communicate with each other indirectly by depositing chemical substances, called pheromones, on the ground. This type of indirect communication is called stigmergy. Indeed, if an obstacle is introduced on the path of the ants,the latter will, after a search phase, all tend to take the shortest path between the nest and the obstacle. The higher the pheromone level in a given location, the more likely an ant will be attracted to that area. The ants that reached the nest the fastest through the food source were those that took the shortest branch of the route.

This is a new version of ACO that we developed in a previous work [**19**] and that we use in this work as a starting algorithm for the design of our hybrid algorithm.

Step 1: Construction of the path by each ant (Solution)

Initially (at time t = 0), the algorithm positions m ants on n city and the intensity of the trace for all pairs of city (ij) is set to a small positive value T_0 in the pheromone matrix. A taboo list is maintained to ensure that a citie cannot be visited twice during the same round. Each ant k will

therefore have its own list of Vk-tabu city which will keep in memory the city already visited. During one iteration of the algorithm, several ants take turns visiting a sequence of city. A cycle (N_c) is completed when the last of the m ants has completed its construction.
After the initialization phase, starting from this sequence of city 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 (1). It allows colonies to favor the shortest paths during the different iterations. During a round, each ant k of a colony L records in its tabu list (memory) the list of city already visited. The probabilistic formula for the selection of a node by the ant k of colony L is defined by the expression of p $_{ij}^{k,l}$ (2).

$$p_{ij}^{k,l}(t) = \begin{cases} \frac{[\eta_{ij}]^{\beta}[\tau_{ij}(t)]^{\alpha}}{\Sigma u \in v_{k_l}[\eta_{iu}]^{\beta}[\tau_{iu}(t)]^{\alpha}} if j \in v_{k_l} 0 \quad else \end{cases} \tag{2}$$

Where v_{k_l} represents the set of nodes or city not visited by ants from colony L.

Step 2: pheromone deposit

When an ant moves from citie i to citie j, it leaves a certain amount of pheromone (value) on the arc (ij). A matrix which is the pheromone matrix records information about the use of the arc (ij). At each step of the turn this matrix is ??updated so that the more this use has been important in the past, the greater the likelihood that these bows will be used again in the future. The evolution of the update equation is as follows:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \triangle\tau_{ij} \tag{3}$$

The evaporation equation of the pheromone matrix is described by the formula

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) \tag{4}$$

The inverse of the distance between city η_{ij} = $\frac{1}{C_{ij}}$ called visibility is static information used to guide the choice of ants to nearby city and avoid too many city. distant. The α and β parameters are used to determine materiality of the intensity of the trace and of the visibility in the construction of a solution. This is based on a compromise between visibility (η_{ij}) and the quantity of pheromone (t_{ij}) present between i and j at cycle t. These parameters are the same as those used in OCF [**2**].

4.2 Procedure of Lin and Kernighan (LK)

4.2.1 The basic algorithm

The K- opt algorithm is based on the K- Optimality concept:
Definition: A visit is said to be k optimal (or simply k - Opt) if it is impossible to get a shorter visit by replacing k links with any other set of k links [**17**]. From this definition, it is obvious that every optimal k-visit is also K ' optimal for 1 <k '< k.

It's also easy to see that a tour that contains n city is optimal if and only if it is n - optimal. Unfortunately, the number of operations to test all k-exchanges is increasing rapidly. In a naive implementation, testing a k-link exchange has a time complexity of O (n ^{k}). Accordingly, the

Algorithm IV.1 MACO Algorithm.

Inputs m: number of ants per colony, L:Number of colonies, n:number of city

```
Nc:=0
Dij:=0          // Global matrix
dij:=t0         //  local matrix of pheromones
 Initialize  t0  // with each ant's home city
Place each colony randomly in a starting point (city)
For K from 1 to m
 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
  Initialize the Dij matrix with the best colony
While(Nc<Nmax)
 For i from 1 to n
  For j from 1 to L
  For K from 1 to m
        Select the city Vi to be added to the current tour with formula(1)
        Evaluate the solution of the ant K on route(i,j)
        Update globally the trace according to city pair(i,j) if it's better than
        the previous ants according to formula 5 and 6 (evaporation)
        Insert as you go (i,j) in the taboo list so as to construct gradually solution K
S:= Best solution (Each colony provides a partial solution)
```

values k = 2 and k = 3 are commonly used. It is a disadvantage that k must be specified in advance because it is difficult to know which k to use to achieve the best compromise between current time and quality of solution. Lin and Kernighan corrected this drawback by introducing a powerful k variable algorithm which changes the values of k during its execution, by deciding on each iteration what the value of k should be. At each step of the iteration the algorithm examines, for ascending values of k, whether and exchange of k links can make it possible to obtain a shorter visit. At each step the algorithm considers an increasing set of potential exchanges (starting with k = 2). If the crawl is successful in finding a new, shorter visit, then the actual visit is replaced with this new visit. With a feasible visit, the algorithm performs exchanges that repeatedly reduce the length of the current visit, until a visit is reached and no other exchange can improve it [**14**].

4.3 Lin - Kergnighan - Helsgaun algorithm

It is the modified and extended version of LK algorithm. Indeed A central rule in the original algorithm is the heuristic rule which restricts the inclusion of links in the visit to the five nearest neighbors to a given citie. This rule directs the search to a shorter visit and reduces the search effort substantially. However, there is a certain risk of not being able to find the optimal cost. Helsgaun amends this rule.

V STUDY OF HYBRID METHODS

Hybridization is a technique which consists in combining the characteristics of two different methods to derive the advantages of the two methods [**4**] [**10**]. In the literature, the hybridization of metaheuristics can be divided into two main parts: hybridization of metaheuristics with metaheuristics and hybridization of metaheuristics with exact methods [**15**] [**10**].

5.1 Hierarchical classification of metaheuristics

This classification is characterized by the level and mode of hybridization. The level of hybridization can be low (Low-Level) or high (High-Level) [**2**]. In the low level, a metaheuristic replaces an operator of another method which encompasses it. On the other hand, in high level hybridization, each metaheuristic keeps its property during hybridization [**15**, **2**]. Each level of hybridization generates two modes of cooperation namely, relay mode and co-evolutionary mode. In relay mode, the methods are executed sequentially, that is to say the result of the first method is the start of the following method [**15**]. When the different methods work in parallel to explore the search space, we speak of co-evolutionary mode. The combination of modes and levels gives four classes of hybridization which are: low-level relay hybridization, low-level co-evolutionary hybridization, high-level relay hybridization and high-level co-evolutionary hybridization [**2**].

5.2 Low-level relay hybridization

It encompasses single solution-based metaheuristics in which another method is incorporated to form a new algorithm[**2**].example: the descent method can be inserted in a simulated annealing algorithm.

5.3 Low-level co-evolutionary hybridization

It consists in incorporating one or more metaheuristics based on a single solution in a metaheuristic with a population of solutions [**4**] [**15**]. The advantage of this type of hybridization is to compensate for the exploitation power of a local search and that of the exploration of a global search[**2**] .

5.4 High-level relay hybridization

It takes place when metaheuristics are used sequentially i.e. the final solution (s) of the first metaheuristic is the initial solution (s) of the following metaheuristic [**15**]. In this procedure, all the methods keep their integrity. Example introduced Taboo research at the end of a genetic algorithm to improve the solutions obtained [**15**].

5.5 High level coevolutionary hybridization

In this case, the metaheuristics used work in parallel by exchanging information between them in order to find the optimal solution of the problem posed [**2**] [**15**].

5.6 Flat classification of hybrid metaheuristics

This is another classification of hybrid methods characterized by the type of hybridized methods, their field of application and the nature of their functions [**15**]. Depending on the type of hybridization, there are homogeneous hybrid methods and heterogeneous hybrid methods. In the homogeneous hybrid methods, the algorithms used are based on the same metaheuristics and in the heterogeneous hybrid methods the metaheuristics used are different[**15**]. The field of application of hybridized metaheuristics makes it possible to distinguish two main classes of hybridization, global hybridization and partial hybridization. Global hybridization takes place when all the hybridized methods are applied to the whole search space[**15**].

Partial hybridization, on the other hand, breaks down a problem into sub-problems where each has its own search space.

VI PROPOSED METHOD: HYBRIDIZATION OF MULTI ANTS COLONY ALGORITHM AND LKH; (MACO-LKH)

6.1 Hybridization Aspects to exploit on replace

In this part, we discuss the structure of our algorithm. It is known that the MACO or LKH algorithms are very efficient for difficult problems and TSP in particular [**4**], but the disadvantage of MACO is the consumption of time [**19**]. On the other hand , LKH must always start its execution with a good starting solution.

In the MACO algorithm described previously, for each ant, the path of a city i to a city j depends on:

- The list of city already visited, which gives the possible choices at each transition, when the ant k is over the city ;
- The inverse of the distance between city called visibility η_{ij}
- The amount of pheromone deposited on the ridge connecting two city, called intensity of the track. This quantity defines the attractiveness of a track, and it is modified after the passage of an ant.

The multi-colony MACO algorithm as described above is made up of three main parts:

- Initialization;
- Construction of paths (Generations of paths) according to the probabilistic formula (2);
- Pheromone deposition and evaporation: Evaluate the solution of each ant in each colony.

To hybridize the two heuristics **MACO** and **LKH**, the visibility function in the MACO heuristic is replaced by the mechanism of the LKH heuristic to be more efficient in restricting the nearest neighbors which results in a gain in execution time. Then the pheromone track update process is replaced by the λ-opt mechanisms used by LKH heuristics in order to accelerate the convergence towards the optimal solution and in another register optimize the optimal solution.

6.2 Hybridization of LKH and MACO: (MACO-LKH)

We opted for an insertion hybridization. That is, we hybridize the MACO and LKH heuristics by inserting LKH into MACO. In clear the main characteristics are:

- The visibility equation between city i and j disappears. This information is replaced by the mechanisms of the LKH process in order to direct the ant more quickly nearby city and thus avoid too long solutions
- After each round of any ant, there is no longer any direct deposition of pheromone at the end, but rather triggering of the LKH process on this round until the best possible round of ant k and then pheromone deposition according to MACO equation(3).

In fact, the algorithm is defined in two steps:

Step 1: MACO algorithm

- Generation of solutions by ants from different colonies;
- Pheromone deposit.

Step 2: Heuristic LKH

- Optimization of solutions;
- Injection of the best ant into the communication matrix.

The algorithm is the following:

Algorithm VI.1 MACO-LKH ALGORITHM

Inputs m: number of ants per colony, L:Number of colonies, n:number of city, T: the displacement step,

```
Nc:=0
Tc:=0
Dij:=0
dij:=0
local pheromone matrix
Initialize t$_{0}$ // with each ant's home city
Place each colony randomly in a starting point (city)
initialization Pheromone = t0;
FOR j from 1 to L
FOR K from 1 to m}
Randomly build a tour by each ant
Application of the LKH method after each T transition
Pheromone deposit by the best ant
Solutions construction
                While(Nc<Nmax)
                For i from 1 to n
            For j from 1 to L
                K:= 1
        While( K < m)
For each ant K of colony L carry out the transition T by selecting for each transition the citie j to be
K = k + 1
Application of the LKH Process on each partial solutions obtained
Insert progressively (i, j) in tabou list so as to build
gradually the solution of K
Perform the global update of the trace according to the best ant of colony L according to equations 3
and 4 after performing the dynamic k−opt operations of LKH on the partial solution
Nc := Nc + 1
S := 1 the best solution
```

VII EXPERIMENTATIONS

In this section, we present the numerical results obtained by the proposed algorithm MACO-LKH.

7.1 Implementation environment

The MACO-LKH approach has been implemented in C language on instances of the TSPLIB online instance library for the TSP on a server with the following characteristics: *4 GHZ processor, 08 GHZ RAM, 500 GO DD* after 100 executions. In the absence of the real data of the problem, we were satisfied with the data instances from the TSPLIB library to evaluate the efficiency of algortithm.

7.2 MACO Algorithm Parameters

parameters	Rôle	Values
Initial amount of pheromone	track quality information	0
Number of colonies	/	50
q_0	rate of intensification and diversification	0.95
1 -ρ	pheromone evaporation rate	0.8
N_{max}	maximum number of iterations	100
α	pheromone intensity control	1
β	visibility control	1

Table 1: MACO Algorithm Parameters

7.3 Results - Discussions and Analysis

To evaluate the efficiency of our algorithm, we conducted a number of experiments. For testing, the algorithm stops in the following cases:

- Either the number of iterations set by the algorithm at the start has been reached. In our case, we set the number of iterations for each instance of the problem to 100;
- Either the optimal solution is reached and no longer evolves.

By making a comparative study between the heuristic MACO-LKH with the heuristics ACS-LKH [**4**], AG [**10**], MACO [**19**], AG -LKH [**4**] which make it from the best heuristics for the TSP of the literature, and under identical test conditions, we obtain the results of table 1,2,3 and 4 below.

Problems (size)	MACO-LKH			MACO		
	Best solution	Average times	Success/100	Best solution	Average times	Success/100
Rd100 (100)	7910	0.9	100	7910	1.33	100
Ts225 (225)	126643	12	100	126643	12.5	100
A280 (2579)	2579	1.04	99	2579	1.6	99
Lin 318 (318)	42029	0.340	100	42029	0.341	100
Att532 (532)	27686.0	11.7	100	27686.0	11.74	100
Si535 (535)	48450	45	100	48450	46.5	100
U574 (574)	36905	32	100	36905	32.9	100
Gr666 (666)	294358	89	100	294358	94	100

Table 2: Comparison of **MACO-LKH** and **MACO** metaheuristics in terms of solution quality and computation time for some small size problem instances from the TSPlib library

Problems (size)	MACO-LKH			MACO		
	Best solution	Average times	Success/100	Best solution	Average times	Success/100
Average Size Problem Instances						
U724 (724)	41910	40.5	100	41910	43	100
Rat783 (783)	8806	1.12	100	8806	2.8	100
Si1032 (1032)	92650	50	100	92650	65.2	100
U1432 (1432)	152970.2	270	82	152970.2	287	82
Vm1084 (1084)	239297	90.5	80	239297	92	80
Pcb1173 (1173)	56892	99.9	97	56892.2	141.2	98
Big Size Problem Instances						
D2103 (2103)	80450.0	186.02	96	80450.0	194.1	98
U2319 (2319)	234256.11	246.9	66	234256.11	259	66
D15112	1573090	155	30	1573090.2	159	26
D18512 (18512)	645240	2659.2	34	645240.7	2696.5	34
Rl11849 (11849)	923289.2	263	88	923289.2	289	88

Table 3: Comparison of **MACO-LKH** and **MACO** metaheuristics in terms of solution quality and computation time on average size instances and Big size instances of some problems from the TSPlib library

Problems	MACO-LKH			GA-LKH			ACS - LKH		
	Best solution	Average times	Success /100	Best solution	Average times	Success/ 100	Best solution	Average times	Success/ 100
Rd100 (100)	7910	0.9	100	7910	11.2	100	7910	5.6	100
Ts225 (225)	126643	12	100	126643.1	64	85	126643	39	65
A280 (280)	2579	1.04	99	2579	1.68	100	2579	1.61	98
Pr439 (439)	107217	93	100	107230	63.3	80	107223	58.3	84
Lin 318 (318)	42029	0.340	100	4229	1.6	80	4229	0.8	100
Si535 (535)	48450	45	100	48450	84	98	48450	78	96

Table 4: Comparison of **MACO-LKH**, **AG-LKH** and **ACS-LKH** metaheuristics in terms of solution quality and computation time for some small size problem instances from the TSPlib library

Problems	MACO-LKH			GA-LKH			ACS - LKH		
	Best solution	Average times	Success /100	Best solution	Average times	Success/ 100	Best solution	Average times	Success/ 100
Average Size Problem Instances									
U724 (724)	41910	40.5	100	41915	192	77	41911.9	98	62
U1432 (1432)	152970.2	270	82	152979	394	45	152976.2	310	32
Vm1748 (1748)	336557.0	94.42	88	336557.2	111.07	80	336557	172.86	88
Pcb1173 (1173)	56892.03	99.9	97	56896.6	930.5	63	56895.9	732.1	53
Vm1084 (1084)	239297	90.5	80	239298	216.8	76	239297.9	119	81
Si1032 (1032)	92650	50	100	92655.1	92	89	92655.9	96.3	65
Rl1323 (1323)	270199	10	86	270199.9	98	60	270199.3	68.6	89
Big Size Problem Instances									
D2103 (2103)	80450.0	186.02	96	80554.9	319.5	47	80553	289.0	22
Usa13509 (13509)	19982859.02	2110.1	22	Undifine	Undifine	Undifine	1998470	2113.03	9
D15112 (15112)	1573094.2	155.2	30	Undifine	Undifine	Undifine	Undifine	Undifine	Undifine
D18512 (18512)	645240	2659.2	34	Undifine	Undifine	Undifine	Undifine	Undifine	Undifine
Rl11849 (11849)	923289.2	263	88	Undifine	Undifine	Undifine	Undifine	Undifine	Undifine

Table 5: Comparison of **MACO-LKH**, **AG-LKH** and **ACS-LKH** metaheuristics in terms of solution quality and computation time on average size instances and Big size instances of some problems from the TSPlib library

7.3.1 Description of the structure of the tables

The different tables are classified according to the size of the input data (small, medium and large size).

- The first column (problems) represents the instances of the problem and the numbers in brackets represent the size of the problem addressed;
- The best solution column represents the best solution obtained after 100 iterations during the experiment;
- The average times column is the average times taken by each metaheuristic used to produce its best solution during the experiment;
- The Success column represents the number of times the algorithms find their best solution out of 100 iterations made;
- The green boxes represent those which have the best execution time or quality of solutions at the end of the 100 iterations among all the algorithms which take part in the experiment;
- The blue boxes represent the boxes where no algorithm of the experiment could reach the optimal solution including MACO-LKH. In the other cases, the optimal solution is reached at least once by one of the methods (generally MACO-LKH and MACO).

7.3.2 *Analysis: Interpretation of the results of the different tables*

The tables 1,2, represent the comparison between the MACO and MACO-LKH approaches. On these tables, we can observe that MACO-LKH and MACO both reached the optimal solution on all instances of small and medium-sized problems except for the case Pcb1173 (1173) where the best solution obtained by MACO is slightly above above the optimal solution.

However, we can see from these same tables that the optimal solution is reached more quickly with the hybrid MACO-LKH method than with the MACO method. This was the first objective sought: to further reduce the execution times of the MACO method.

With regard to the improvement of the quality of optimal solution obtained, the hybrid method MACO-LKH it is slightly demarcated on certain instances (of large sizes in general) compared to the method MACO with a solution slightly higher than that of MACO. Which was the second objective sought. Even if it did not make itself felt too much overall.

The tables 3, 4 show the comparison between the hybrid method MACO-LKH and the hybrid methods AG-LKH and ACS-LKH which are hybrid metaheuristics which have also proven themselves [**4**].

- On the quality aspect of optimal solution, MACO-LKH is clearly superior to the other two (AG-LKH and ACS-LKH). This is thanks to the new hybridization strategy which uses the optation strategy of the LKH method for the choice of the next city to visit (limiting the search to a certain number of nearest neighbors thanks to the LKH mechanism). Cost differences between MACO-LKH and AG-LKH, ACS-LKH algorithms can be up to 15%;
- On the execution time aspect, MACO-LKH is once again above the AG-LKH and ACS-LKH methods for all the instances used during our experiment. The differences between the execution times of MACO-LKH methods and AG-LKH, ACS-LKH methods can be up to 30 $\%$ on average. In some cases, the execution times of the AG-LKH and ACS-LKH methods are indefinite while MACO-LKH succeed in finding a time (sometimes enormous) and a solution (not necessarily optimal).

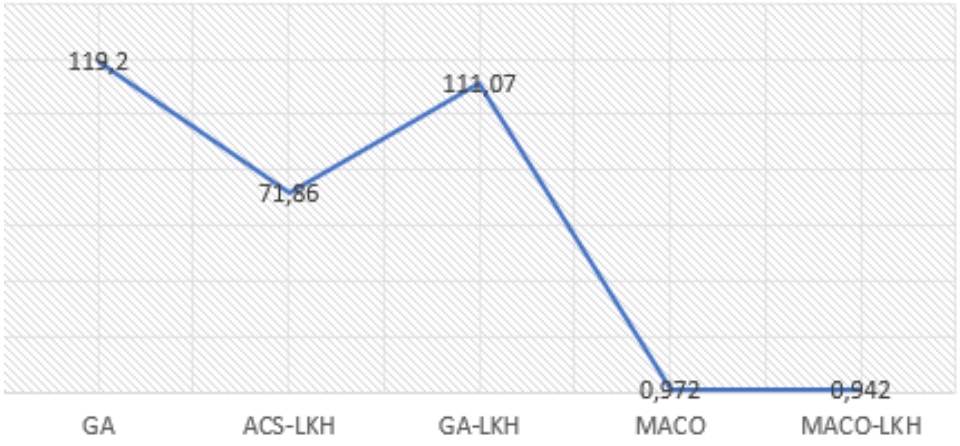

Figure 1: Comparison MACO, AG-LKH, ACS-LKH,MACO-LKH in terms of running times

The figure 1 present a comparison of the performance of our approach MACO-LKH compared to the improved AG -LKH, ACS -LKH, and MACO approaches in terms of runtime.

VIII CONCLUSION

In this work, we proposed a new hybrid low-level and co-evolutionary heuristic between MACO and LKH metaheuristics to form a new method called (MACO-LKH). It has been applied to effectively solve the TSP. The tests carried out on several TSP instances of small, medium, and large size have shown that the proposed hybrid method MACO-LKH is able to provide a better optimal solution with faster execution times than the heuristics ACS-LKH, MACO, GA-LKH. The proposed MACO-LKH algorithm thus shows improvements in both cost and execution time compared to other methods (MACO, ACS-LKH-GA-LKH.

A limit related to our approach is the use of much more resources (memory and processor). Because for most large instances, this execution time remains enormous. Also, for these large instances, the optimal solution is only achieved at around 65 - 85% on average.

In perspective, we plan to use our algorithm on real data from logistics, transport, e-commerce problems... We also plan to parallelize the MACO heuristic to further reduce execution times on big data problems.
We also plan to use machine learning techniques to hybridize MACO and LKH.

REFERENCES

[1] DEHAN. MICHAËL Distribution en Belgique des médicaments issus du plasma sanguin: une étude de cas , *universite catholique de louvain,*pp 50-73, 2018.

[2] AHMIA. IBTISSAMAL, Une nouvelle metaheuristique pour les problemes d'optimisation combinatoire : la Monarchie Metaheuristique, *Universite des sciences et de la technologie houari boumediene*, *Phd Thesis ,* 2019

[3] K. HELSGAUN, An effective implementation of the Lin-Kernighan traveling salesman heuristic, *AIEEE Transactions on Evolutionary Computation* pp 106-130, 2000.

[4] BAUDOIN NGUIMEYA.M Soh.L P FOTSO, Algorithmes hybrides pour la résolution du problème du voyageur de commerce, *Proceedings of CARI* 2016.

[5] TSPLIB95, Traveling Salesman Problem Library , http://www.iwr.uniheidelberg.de/iwr/comopt/software/TSPLIB95, December,2021.

[6] T.STUETZLE, The Traveling Salesman Problem: State of the Art, *TUD SAP AG Worshop on Vehicle Routing* july 10, 2003.

[7] B. DELISLE, Parallélisation d'un algorithme d'optimisation par Colonies de Fourmis pour la résolution d'un problème d'ordonnancement industriel, *IEEE* pp 53-66, 2002.

[8] D.DJOHNSONL.A.GEOCH, The traveling salesman problem:A case study in local optimization, *Local Search in Combinatorial Optimization*,1997.

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

[10] B. TADUNFOCK TETI,L.P. FOTSO, Heuristiques du problème du voyageur de commerce, *Proceedings of CARI 2006*, pp 1-8, 2006.

[11] M. DORIGO, GAMBARDELLA, Ant Colony System : A cooperative learning approach to the traveling salesman problem , *AIEEE Transactions on Evolutionary*, pp 53-66, 1997.

[12] S. DIGABEL, Problème du voyageur de commerce (TSP) , 2018 .

[14] C. AUDET DENNIS, Mesh adaptive direct search algorithms for constrained optimization , *SIAM Journal on Optimization,* 2006 .

[15] HANAÂ HACHIMI, Hybridations d'algorithmes métaheuristiques en optimisation globale et leurs applications , *Thèse de doctorat, Université Mohammed,* 2013 .

[16] EMILIE DANNA AND DAVID L, How to select a small set of diverse solutions to mixed integer programming problems) , *Operations Research Letters,* 2009.

[17] R. MARTIN., Single-interval learning by simile within a simulated hebbian neural network. , *Computers Mathematics with Applications,* 1990.

[18] THOMAS STUTZLEHolger Hoos, Max-min ant system. , *Future Generation Computer Systems,* 2000.

[19] M.SOH, B.NGUIMEYA, C.TAYOU DJAMEGNI, A Multi Ant Colony Heuristic Approach For Solving The Traveling Salesman Problem , *Revue Africaine de la Recherche en Informatique et Mathematique Appliquees*, *INRIA*, *Volume 34*, 2021, DOI:10.46298/arima.6752

[20] M. DORIGO,V. Maniezzo, and A Colorni. Ant system : optimization by a colony of cooperating agents, *IEEE Trans. on Man. Cyber. Part B,* 2020.

[21] XIAOXIA ZHANG,LIXIN TANG, A New Hybrid Ant Colony Optimization Algorithm for the Traveling Salesman Problem , *The Logistics Institute, Northeastern University, Shenyang, China*, 2008.

[22] OLIEF ILMANDIRA RATU FARISI1, BUDI SETIYONO2, IMBANG , A Hybrid Firefly Algorithm Ant Colony Optimization for Traveling Salesman Problem, *The Logistics Institute, Northeastern University, Shenyang, China*, 2015.

[23] PETER STODOLA , KAREL MICHENKA , Jan Nohel MARIAN RYBANSKY, Hybrid Algorithm Based on Ant Colony Optimization and Simulated Annealing Applied to the Dynamic Traveling Salesman Problem, *Department of Intelligence Support, University of Defence, Kounicova* , 2020.

[24] IBRAHIM MOUSSA, Modèles de résolution approchée et efficace pour les problèmes des réseaux de transport et de télécommunication, *thèse*, *Université de picardie jules verne d'amiens*, 2015.