# Performance Evaluation of Ant Colony Systems for the Single-Depot Multiple Traveling Salesman Problem

Raluca Necula[1], Mihaela Breaban[1], and Madalina Raschip[2]

[1] Faculty of Computer Science, Alexandru Ioan Cuza University, Iasi, Romania
{raluca.necula,pmihaela}@info.uaic.ro
[2] Information Management Institute, University of Neuchatel, CH-2000 Neuchatel, Switzerland
mionita@info.uaic.ro

**Abstract.** Derived from the well-known Traveling Salesman problem (TSP), the multiple-Traveling Salesman problem (multiple-TSP) with single depot is a straightforward generalization: several salesmen located in a given city (the depot) need to visit a set of interconnected cities, such that each city is visited exactly once (by a single salesman) while the total cost of their tours is minimized. Designed for shortest path problems and with proven efficiency for TSP, Ant Colony Systems (ACS) are a natural choice for multiple-TSP as well. Although several variations of ant algorithms for multiple-TSP are reported in the literature, there is no clear evidence on their comparative performance. The contribution of this paper is twofold: it provides a benchmark for single-depot-multiple-TSP with reported optima and performs a thorough experimental evaluation of several variations of the ACS on this problem.

**Keywords:** multiple-TSP, ant colony optimization, clustering

## 1 Introduction

The traveling salesman problem (TSP) is one of the most famous combinatorial optimization problems. The multiple traveling salesman problem (MTSP) is an extension of this well-known problem that involves further a new parameter - the number of salesmen. It can accommodate easily related real-world problems, especially routing and scheduling problems.

The problem is NP-hard and is difficult to solve. One solution is to transform the problem, with $m$ salesmen and $n$ cities, into a TSP with $n + m - 1$ cities by adding $m - 1$ artificial depots $n + 1, ..., n + m - 1$. The resulting TSP is highly degenerate and more difficult to solve than an ordinary TSP with the same number of cities.

There are various approaches in literature to solve variants of the multiple TSP. A detailed overview of the problem's variants and methods used for solving them is available in [1]. Exact algorithms, like cutting planes [2] or branch and bound [3] were developed, but they can obtain optimal solutions only for small size instances in acceptable time. Due to the combinatorial complexity of the problem, heuristics to solve real-world instances were also employed. For large dimensions of the problem, heuristic approaches like k-opt approach [4], or meta-heuristics like Tabu Search [5], Genetic Algorithms [6], Ant Colony Optimization [7], Neural Networks [8] are necessary to solve it. The most frequently used class of algorithms for solving the multiple TSP problem is Genetic Algorithms. For example, in [6] the authors present a new chromosome representation that works with the classical genetic operators for the TSP. The representation dramatically reduces the number of redundant solutions.

With a proven efficiency on the standard TSP, ant algorithms were also adapted for the multiple TSP. In [9] the ant colony optimization is used to solve the multiple TSP with the ability constraint. The problem imposes that the number of cities which are traveled by each salesman to be upper bounded by a value. The problem was converted to TSP. In [7] two objectives were considered: minimizing the total tour length of all the salesmen and minimizing the maximum tour length of each salesman. The ACO algorithm follows the MAX-MIN Ant System scheme and integrates a local improvement procedure. Computational results show the competitiveness of the algorithm.

Another criterion, much less addressed, but important in real-world applications is the balancing of workloads amongst salesmen. In [10], the authors present a comparison of evolutionary computation algorithms and paradigms for the euclidean multiple traveling salesman problem. The authors are concerned with the first level of optimization, the optimal subdivision of cities into groups. To this end, the chromosome representation makes use of the neighborhood attractor schema which is a variation of k-means. The shrink-wrap algorithm is used to determine the circuit path lengths. Another clustering approach to solve the multiple TSP is proposed in [11]. The clustering algorithm minimizes the variation of distances traveled within each cluster. A good balance of workloads among clusters is achieved.

The current paper aims at proposing and evaluating several variations of the Ant Colony System (ACS) for the single-depot multiple-TSP. The paper is structured as follows. Section 2 defines the problem as an integer linear program (ILP) that can be solved to optimality with dedicated software. Section 3 reviews the standard formulation of the Ant Colony System. A number of 5 variants of the standard ACS adapted for the single-depot multiple TSP problem are subsequently described in Section 4. Experiments and results are presented in Section 5.

## 2 The single-depot multiple traveling salesman problem

There are several integer linear programming formulations for the multiple-TSP. We have used the variant that restricts the minimal number of nodes that a salesman may visit [1]. Such restrictions appear in real-life applications where the purpose is to have a good balance of workloads for the salesmen.

Let $G = (V, A)$ be a directed graph where $V$ is the set of nodes and $A$ is the set of arcs and $C = (c_{ij})$ is the cost(distance) matrix associated with each arc $(i, j) \in A$. Let $n$ be the number of cities and $m$ be the number of salesmen. All salesmen are located at the depot city 1. They start and end at the same depot, and each other node is located in only one tour. The number of nodes a salesman can visit lies within a predetermined interval. The problem is to find the tours of each salesman such that the previous restrictions are satisfied and the overall cost of visiting all nodes is minimized.

Let $x_{ij}$ be a binary variable that is equal to 1 if the arc $(i, j)$ is contained in the optimal solution and 0 otherwise. $u_i$ denotes the number of nodes visited on that salesman's path from the origin to node $i$, for any salesman, i.e. the position of node $i$ in a tour. Let $L$ be the maximum number of nodes a salesman may visit, and $K$ the minimum number of nodes a salesman must visit.

The problem is formulated as the following:

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} c_{ij}x_{ij} && (1)\\
\text{s.t.} \quad & \sum_{j=2} x_{1j} = m && (2)\\
& \sum_{j=2} x_{j1} = m && (3)\\
& \sum_{i=1}^{n} x_{ij} = 1,\ j = 2, .., n && (4)\\
& \sum_{j=1}^{n} x_{ij} = 1,\ i = 2, .., n && (5)\\
& u_i + (L-2)x_{1i} - x_{i1} \leq L - 1,\ i = 2, .., n && (6)\\
& u_i + x_{1i} + (2-K)x_{i1} \geq 2,\ i = 2, .., n && (7)\\
& x_{1i} + x_{i1} \leq 1,\ i = 2, .., n && (8)\\
& u_i - u_j + Lx_{ij} + (L-2)x_{ji} \leq L - 1,\ 2 \leq i \neq j \leq n && (9)\\
& x_{ij} \in \{0, 1\},\ \forall (i, j) \in A. && (10)
\end{aligned}
$$

Constraints (2) and (3) ensure that exactly $m$ salesmen depart from and return to the depot. Constraints (4) and (5) are the degree constraints, i.e. exactly one tour enters and exits each node. Constraints (6) and (7) are bounding constraints, their corresponding inequalities serve as upper and lower bound constraints on the number of nodes visited by each salesman. Inequality (8) forbids a vehicle from visiting only a single node. The inequalities from (9) ensure that $u_j = u_i + 1$ if and only if $x_{ij} = 1$. Constraints (9) are the classical subtour elimination constraints that prevent the formation of any subtour between nodes in $V \setminus \{1\}$.

The formulation is valid if $2 \leq K \leq \lfloor (n-1)/m \rfloor$ and $L \geq K$. Constraint (8) becomes redundant when $K \geq 4$.

## 3 The Ant Colony System

The fundamental idea of the ant colony algorithms is inspired by the way the natural ants succeed in finding food. The ants communicate via the pheromone trails in order to find the shortest paths from the nest to the food sources. Although initially designed to solve shortest-path problems in graphs, the application of ant colony algorithms is not restricted to this class of problems; they have been applied successfully to other various combinatorial optimization problems as well.

The algorithms investigated in this paper are based on the original version of the Ant Colony System designed by Dorigo and Gambardella for the Traveling Salesman problem [12], this one being the most popular from the class of ant algorithms. In this section we review the main steps of the standard algorithm for solving TSP, leaving the presentation of its variations for multiple-TSP for the next section.

In ACS, several ants are placed in the nodes of the graph representing the TSP instance. At each iteration, each ant builds a tour in the graph; on each edge it traverses, the ant lays a constant quantity of pheromone. At the end of each iteration, after all ants built their tours, the best solution recorded so far, called global solution, is updated (if necessary). Then, the edges on the path given by the global solution receive an extra quantity of pheromone which is inverse proportional with the cost of the global solution. Three important phases, discussed next, are thus involved in this optimization process: node selection at route construction, local pheromone update performed each time an ant traverses an edge and global pheromone update that highlights the route with the minimum cost identified so far.

### 3.1 Route selection

Each ant builds a route by iteratively selecting a node it has not visited yet. At each step in this process, the nodes not selected yet form the candidate set. The node to be added to the current route is chosen relative to the current position of the ant, in a probabilistic manner, from the candidate set. A probability of selection is assigned to each node in the candidate set which is inverse proportional with the distance to the current position of the ant (the selection of closer nodes is thus encouraged) and direct proportional with the quantity of pheromone laid on the edge connecting the node to the current position. The probability assigned to a node $s$ in the candidate set $C$, considering that node $r$ is the current position of the ant, is computed with equation (11):

$$p(r,s) = \frac{\tau(r,s) \cdot \eta^{\beta}(r,s)}{\sum_{u \in C} \tau(r,u) \cdot \eta^{\beta}(r,u)} \tag{11}$$

where $\tau$ is the pheromone, $\eta$ is the inverse of the cost measure (distance) $\delta(r,s)$, and $\beta$ is a parameter that specifies the relative importance of pheromone vs. distance.

The product $\tau(r,s) \cdot \eta^{\beta}(r,s)$ can be viewed as the fitness of node $s$ while the probabilistic selection based on the probabilities defined in equation (11) corresponds to the *roulette wheel* selection in genetic algorithms. Equation (11) is the standard selection scheme in Ant System, the precursor of ACS. To encourage exploitation, ACS introduces also a kind of elitist selection: based on a given rate $q_0$ - static numerical parameter in ACS playing the role of a probability, the best node is deterministically selected.

The selection of the next node for tour construction in ACS can be synthesised by the following equation:

$$s = \begin{cases} \arg\max_{s \in C} \tau(r,s) \cdot \eta^{\beta}(r,s), & \text{if } rand(0,1) < q_0 \\ S, & \text{otherwise} \end{cases}$$

where $q_0$ is a parameter and $S$ is a random variable with the probability distribution given by equation (11).

## 3.2 Local pheromone update

Each time an ant builds a route, it lays a constant quantity of pheromone on each edge it traverses. The pheromone of each edge traversed is updated by equation (12):

$$\tau(r,s) = (1-\rho) \cdot \tau(r,s) + \rho \cdot \tau_0 \tag{12}$$

where $\rho \in (0,1)$ is a local pheromone decay parameter and $\tau_0 = (n \cdot L_{NN})^{-1}$ is a quantity set in the first iteration of the algorithm by building in a greedy manner based on a nearest-neighbor heuristic a tour and computing its length denoted by $L_{NN}$.

## 3.3 Global pheromone update

Usually, ants search for food in a neighborhood of the best tour. In order to make the search more directed, after all ants have completed their tours, the globally updating phase of the pheromone follows. Only the edges included in the best global solution receive pheromone in this phase. The pheromone level is updated by applying the following rule:

$$\tau(r,s) = (1-\alpha) \cdot \tau(r,s) + \alpha \cdot \Delta\tau(r,s) \tag{13}$$

where

$$\Delta\tau(r,s) = \begin{cases} (L_{gb})^{-1}, & \text{if } (r,s) \in \text{global-best-tour} \\ 0, & \text{otherwise} \end{cases}$$

where $\alpha \in (0,1)$ is the pheromone decay parameter and $L_{gb}$ is the length of the globally best tour encountered. By this rule, only the edges that belong to the globally best tour receive reinforcement.

## 4 Algorithms investigated for multiple-TSP

### 4.1 Problem decomposition with k-Means followed by ACS for TSP (kM-ACS)

One possible approach to tackle the multiple-TSP problem is to decompose the problem instance into a number of small subproblems equal to the number of salesmen; subsequently, each subproblem can be solved by the standard ACS resulting in one (near)optimal subtour for the corresponding salesman. The problem decomposition can be performed by means of clustering algorithms. If the problem instance is expressed in the form of a cost matrix encoding the distances between all the cities, a graph clustering algorithm is necessary to group closely interconnected cities. If the problem instance provides two-dimensional coordinates of the cities, standard cluster analysis may be performed, by considering each city as an item in the data set, characterized by two numerical attributes corresponding to the two coordinates. Among the popular clustering algorithms, k-Means seems to be the most appropriate choice due to its reduced time and space complexity and, more important, due to the fact that it is known to generate groups of equal volumes. The latter characteristic is desirable for the multiple-TSP problem, where we aim to obtain balanced subtours.

To split one single-depot multiple-TSP instance into several TSP instances, we need to adapt the k-Means algorithm. Our problem formulation necessitates that the city corresponding to the depot is visited by each salesman. This constraint imposes that the depot should be included in each of the TSP subproblems generated. Since k-Means is a crisp-clustering algorithm, when applied the multiple-TSP instance, it will generate disjoint sets of cities. To obtain an optimal clustering of the cities where all groups share the depot city, the assignment step is changed in k-Means: at every iteration in k-Means, all the cities are assigned to the cluster based on their distances to the centroids, with the exception of the depot which is assigned to every cluster; thus, all the centroids will be biased towards the location of the depot. This modification discourages the clustering algorithm from forming isolated groups, at far distances from the depot.

After the cities are clustered into $m$ groups (where $m$ is the number of salesmen), $m$ independent runs of the ACS are performed, one for each group; in this step, $m$ smaller TSP instances are practically solved. The final solution for the multiple-TSP problem is obtained by aggregating the solutions obtained in the ACS runs, as subtours to be assigned to the $m$ salesmen.

This version of the algorithm is denoted in the experimental section by $kM-ACS$.

### 4.2 ACS with global-solution pheromone update (g-ACS)

Several salesmen can compete towards building the subtours in one run of the ACS, idea found also in [9]. In such an approach, for a problem instance with $m$ salesmen, $m$ salesmen are placed at the depot location - each ant corresponding to a team of $m$ salesmen, their tours describing a candidate solution to the multiple-TSP problem. At each step, one salesman is chosen at random (selection with replacement). The selected salesman chooses the next city in its subtour in agreement with the route selection mechanism in ACS, detailed in Section 3.1; the candidate set of nodes consists of the nodes that are not selected in any of the $m$ subtours under construction. This process is repeated until a complete solution is obtained (when the candidate set of nodes is empty). Each traversed edge receives the quantity of pheromone computed with equation (12), regardless of which salesman traverses it. The equivalent of the nearest-neighbor tour for TSP used to initialize parameter $\tau_0$, is generated for multiple-TSP by randomly chosen the salesman to perform the next move, followed by a greedy choice of the node from the candidate set (the node at the shortest distance).

At each iteration of the algorithm, several groups of salesmen, each of size $m$, build complete solutions as explained above. From these solutions, the one with the smallest cost - measured as the sum of the costs of the $m$ subtours - is considered to be the global best at the given iteration. The best-so-far solution (global best - the best solution encountered during the run) is updated if necessary. Then, the edges encountered in the subtours of the global best solution are updated with equation (13), where $L_{gb}$ is the total cost of the global solution.

This version of the ACS algorithm is denoted in the experimental section as $g - ACS$.

## 4.3 ACS with subtour pheromone update (s-ACS)

Another version we propose, denoted as $s - ACS$, is a small variation of the previous algorithm. The difference between the two consists only in the third phase of the ACS algorithm - the global pheromone update. $s - ACS$ does not use the same pheromone quantity on each edge, but differentiates the edges based on the subtour to which they are assigned. Thus, we refine/update equation (13) by including subtour information as follows:

$$\tau(r,s) = (1-\alpha)\cdot\tau(r,s) + \alpha\cdot\Delta\tau(r,s) \tag{14}$$

where

$$\Delta\tau(r,s) = \begin{cases} (L_k)^{-1}, & \text{if } (r,s) \in \text{subtour k in global-best-tour} \\ 0, & \text{otherwise} \end{cases}$$

According to the new equation, each edge included in the best solution encountered so far (the global best solution) receives a quantity of pheromone inverse proportional with the cost of the subtour to which it belongs.

## 4.4 ACS with global-solution pheromone update and bounded tours (gb-ACS)

Another variation we propose, is to enforce within the $g - ACS$ algorithm the bounds imposed by the problem with regard to the number of cities to be included in each subtour.

Although in the phase of solution construction, the salesman designated to add one more city to its subtour is uniformly drawn at random (i.e. each salesman has the same probability), experimental studies showed that the solution built is sometimes highly unbalanced with regard to the number of cities included in the subtours. This is justified, since the law of large numbers from statistics is not applicable to the size of the problems we deal with.

The bounds on the size of each subtour are imposed by changing the way salesmen are selected when constructing the solution. Here, we use a selection strategy without replacement: in a first step, the lower bound $K$ on the number of cities to be included in each subtour is guaranteed by sampling at random from a population of size $K \cdot m$ without replacement, consisting of $K$ instances of each of the $m$ salesmen. When this sampling procedure ends (all instances are used during solution construction), all $m$ subtours consist of the minimum number of cities allowed. A similar salesman selection scheme begins (without replacement) with a population of size $(L-K) \cdot m$ ($L-K$ instances of each of the $m$ salesmen) to guarantee that the maximum bound imposed on the size of each subtour is not exceeded. This sampling phase ends when a complete solution is built.

This variant of the $g - ACS$ algorithm where bounds are enforced, is called (gb-ACS).

### 4.5 ACS with subtour pheromone update and bounded tours (sb-ACS)

The bounds on the size of the subtours are imposed in the same manner in the ACS algorithm with pheromone updates based on the cost of the subtours ($s - ACS$). This leads to new variant denoted as $sb - ACS$.

## 5 Experiments

### 5.1 Problem instances

Although multiple-TSP is, by its formulation, only one step away from the standard TSP problem which provides very popular benchmarks such as TSPLIB[3], there is no freely available benchmark for multiple-TSP at this moment to test the performance of various heuristics. To fill this gap, we formulate several problem instances for multiple-TSP and provide exact solutions by solving them in CPLEX[4].

The problem instances we propose are based on the TSPLIB benchmark. Specifically, we extracted from TSPLIB four instances (listed in Table 1) of various size and distribution and we transformed them in multiple-TSP instances by setting the number of salesmen to be, by turn, 2, 3, 5 and 7. The selected TSPLIB instances were chosen so that to reflect different distributions of cities

[3] `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`

[4] `http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/`

(uniform vs. non-uniform) and different positions of the depot city (center, bottom left). These settings generate from each TSPLIB instance, 4 multiple-TSP instances. All the instances are formulated by listing the 2-dimensional coordinates of the cities. The depot, for each instance, is considered to be the first city in the file. The problem instances along with the distribution of cities and position of depot city, the optimal solutions and their visualisations are publicly available as multiple-TSPLIB.[5]

Table 1: Multiple TSP instances

| TSP instance | n (#cities) | m (#salesmen) | MTSP instance | K (lower bound) | L (upper bound) |
|---|---|---|---|---|---|
| eil51 | 51 | 2 | eil51-m2 | 23 | 27 |
| | | 3 | eil51-m3 | 15 | 20 |
| | | 5 | eil51-m5 | 7 | 12 |
| | | 7 | eil51-m7 | 5 | 10 |
| berlin52 | 52 | 2 | berlin52-m2 | 10 | 41 |
| | | 3 | berlin52-m3 | 10 | 27 |
| | | 5 | berlin52-m5 | 6 | 17 |
| | | 7 | berlin52-m7 | 4 | 17 |
| eil76 | 76 | 2 | eil76-m2 | 36 | 39 |
| | | 3 | eil76-m3 | 21 | 30 |
| | | 5 | eil76-m5 | 12 | 17 |
| | | 7 | eil76-m7 | 7 | 15 |
| rat99 | 99 | 2 | rat99-m2 | 46 | 52 |
| | | 3 | rat99-m3 | 27 | 36 |
| | | 5 | rat99-m5 | 13 | 30 |
| | | 7 | rat99-m7 | 9 | 22 |

As stated in Section 2, we impose bounds on the number of cities each salesman needs to visit. If considered as a problem with a single objective - that of minimising the total cost of visiting all the cities - and no bounds are imposed on the number of cities to be visited by each salesman, the multiple-TSP with single depot is an ill-posed problem: the optimal solution is obtained when one salesman visits all the cities while the rest do not have assigned any work. This is obvious by comparing the total costs obtained on the same TSPLIB instance with different numbers of salesmen: when increasing the number of salesmen, the total cost increases. The reason is evident if we think that each new salesman, needs to start and end its tour at the depot; thus, the city corresponding to the depot is visited many times, each of these visits adding new costs to the solution.

For example, when the constraints 6 and 7 are omitted from the ILP formulation of multiple-TSP, the optimum solution for the MTSPLIB instance eil51-m5 (51 cities and 5 salesmen) has the following structure: 4 salesmen visit, each one,

[5] `www.infoiasi.ro/~mtsplib`

only two cities - including the depot (because of constraint (8) in the ILP formulation, the minimum allowed number of cities is 1), while the fifth salesman visits 46 cities.

The use of several salesmen in practice for single-depot-multiple-TSP does not aim to obtain better costs compared to the case when a single salesman is used, but aims at shortening the time needed to serve all the clients or is triggered by the limited capacity of each agent - problem closely related to the Capacitated Vehicle Routing. From the first point of view, single-depot-multiple-TSP is inherently a bi-objective optimization problem: the total cost should be minimized while maintaining the subtours as balanced as possible; the bi-objective formulation will be the scope of our future studies. The second approach, which imposes upper bounds on the capacity of the agents and lower bounds to enforce a balanced distribution of the total work is employed in this study. In fact, imposing these bounds on the number of cities visited by each salesman, increases significantly the complexity of the multiple-TSP problem being solved. Obtaining with CPLEX the optimal solution for TSP instances lasted at most 2 minutes (in case of rat99 instance), whilst in case of MTSPLIB instances, when no bounding constraints were considered (constraints 6 and 7 were removed from the ILP formulation of multiple-TSP), the maximum running time for CPLEX to reach the optimal solution was 17 minutes. This is a major difference compared with the time required by CPLEX to solve MTSPLIB instances when these bounding constraints are included in the ILP formulation of multiple-TSP; the reported solution for the rat99-m7 problem instance was obtained with CPLEX after 14 hours. All the runs were performed on a PC with 8 GB RAM, processor Intel Core i5-4590S 3.00 GHz using 4 process- ing units (multi-threaded implementation of CPLEX). In comparison, one run of any ACS variant described in this paper required on average 1 second and 11 milliseconds in a single-threaded implementation.

With the aim of obtaining balanced solutions, we chose to set the bounds ($L$ and $K$) on the number of cities to be included in each subtour, by running the k-Means clustering algorithm, modified towards biasing the clusters to include the depot city (as described in section 4.1). Because k-Means is an iterative heuristic aiming to minimize the within-cluster variance, its results being highly dependent on the initialization step, we performed 30 k-Means runs on the same problem instance to split the cities in $m$ groups; the partition with the lowest sum of the within-cluster variances was used to set $K$ equal to the size of the smallest cluster and $L$ equal to the size of the largest cluster.

## 5.2 Parameters setup

The standard ACS algorithm introduces some numerical parameters which, usually, are empirically optimized.

As suggested in [12], we set $q_0 = 0.9$, $\alpha = 1.0$, $\beta = 2.0$, $\rho = 0.1$ in all tested algorithms.

The number of ants in $kM - ACS$ was set to 10 for each TSP subproblem (corresponding to each of the $m$ clusters), corresponding to a total of $10 \cdot m$

ants used to solve each of the MTSP problem instances. The number of ants used in all the other versions is also set to $10 \cdot m$ in the following way: at each iteration of these algorithms 10 groups of ants, each of size $m$, build in parallel, independently, 10 complete candidate solutions. Thus, given the same number of iterations for all algorithms investigated here, the computational cost is the same - if omitting the cost involved by the clustering algorithm in $kM - ACS$.

Each algorithm is ran for 1 400 iterations for the problem instances with 51 and 52 cities, 1 800 iterations for eil76 problem instances and 2 200 iterations for rat99 problem instances.

### 5.3 Results

We provide results for each investigated algorithm regarding both the total cost of the solutions and the balancing degree of the subtours.

For all algorithms, we report the average cost of the solutions over 50 runs.

In order to evaluate the balancing degree, for each solution returned at the end of a run we compute the amplitude of the costs of its subtours as the difference between the cost of the longest subtour and the cost of the shortest subtour of the solution. We report the average amplitude over 50 runs for each algorithm. Please note that the amplitude is computed taking into account subtour costs and not subtour cardinalities (the number of cities in each subtour), because the two criteria are not equivalent.

Figure 1 illustrates the evolution of the best cost (best solution) during the run of the algorithms for the eil76-m5 problem; the curves are obtained as averages over 10 runs for each algorithm. As anticipated, kM-ACS shows the highest convergence rate, because it solves in parallel several smaller and simpler problems. However, the performance with regard to the best solution it can achieve is limited, because of the problem decomposition scheme which imposes rigid boundaries on the candidate solution space; in this regard, the non-deterministic character of k-Means introduced by random initializations is an advantage. The best performance on this problem instance is recorded by gb-ACS - the bounded version of g-ACS which uses the cost of the global solution (and not of the individual subtours) for pheromone update.

Tables 2 to 5 report the costs of the optimal solutions and their corresponding amplitudes, as obtained with CPLEX, and the performance of all the ACS variants we propose - as averages over 50 runs. For each problem instance and ACS variant, along with the average cost we also report the standard error of the mean multiplied with the 0.975 quantile of the Student's t distribution (used to compute confidence intervals). Where CPLEX was stopped before finding the optimal solution, both the upper (corresponding to the best known solution) and the lower bound on the cost are given; the amplitude in this case is computed on the best known solution.

Figures 2 to 5 illustrate, for each problem, the distribution of the costs, the amplitudes of subtour costs and confidence intervals for the mean of the costs, for the solutions obtained in 50 runs of each algorithm. On each chart corresponding to a TSP instance, 4 groups are emphasized, corresponding to

Table 2: The performance of the ACS variants for the eil51 instance

| problem | $m$ | measure | optimum | kM-ACS | g-ACS | s-ACS | gb-ACS | sb-ACS |
|---|---|---|---|---|---|---|---|---|
| eil51 | 2 | cost | **442.32** | 454.30 ±0.84 | 452.66 ±1.77 | 454.96 ±2.04 | 452.22 ±1.48 | 453.81±1.63 |
| | | amplitude | **14.18** | 57.69 | 72.29 | 47.54 | 30.42 | 28.79 |
| | 3 | cost | **464.11** | 500.00 ±0.24 | 485.73 ±3.44 | 489.64 ±3.59 | 479.51 ±3.37 | 483.39 ±3.75 |
| | | amplitude | 41.78 | **25.97** | 98.47 | 98.68 | 49.41 | 47.51 |
| | 5 | cost | **[519.10, 529.70]** | 563.58 ±0.52 | 582.36 ±3.58 | 590.63 ±4.64 | 585.76 ±4.34 | 598.61 ±5.16 |
| | | amplitude | 62.19 | 114.35 | 104.65 | 99.11 | 58.09 | **57.45** |
| | 7 | cost | **[584.02, 605.21]** | 634.47 ±0.04 | 674.78 ±4.32 | 680.38 ±3.84 | 688.26 ±3.57 | 699.47 ±4.34 |
| | | amplitude | 86.17 | 77.44 | 100.19 | **62.13** | 70.01 | 68.27 |

Table 3: The performance of the ACS variants for the berlin52 instance

| problem | $m$ | measure | optimum | kM-ACS | g-ACS | s-ACS | gb-ACS | sb-ACS |
|---|---|---|---|---|---|---|---|---|
| berlin52 | 2 | cost | **7753.89** | 8836.80 ±27.15 | 8043.92 ±46.91 | 8036.08 ±43.46 | 8057.38 ±43.06 | 8122.44 ±46.44 |
| | | amplitude | 2195.32 | **1872.41** | 2463.58 | 2389.42 | 1961.50 | 2047.07 |
| | 3 | cost | **8106.85** | 9009.18 ±11.83 | 8653.86 ±147.04 | 8806.95 ±64.40 | 8795.52 ±48.13 | 8839.37 ±42.63 |
| | | amplitude | 4055.24 | **1065.40** | 3562.22 | 3586.32 | 3010.65 | 3042.33 |
| | 5 | cost | **[8894.50, 9126.33]** | 10335.03 ±1.88 | 10164.58 ±90.66 | 10343.52 ±93.34 | 10660.46 ±92.58 | 10866.66 ±106.15 |
| | | amplitude | 3259.84 | **2135.51** | 3546.92 | 3590.37 | 2844.70 | 2916.53 |
| | 7 | cost | **[9415.99, 9870.02]** | 11966.20 ±2.26 | 11993.31 ±137.66 | 12125.55 ±121.75 | 12451.16 ±104.28 | 12712.41 ±97.88 |
| | | amplitude | 4012.18 | **2154.70** | 3428.38 | 3528.07 | 2781.41 | 2712.72 |

Table 4: The performance of the ACS variants for the eil76 instance

| problem | $m$ | measure | optimum | kM-ACS | g-ACS | s-ACS | gb-ACS | sb-ACS |
|---|---|---|---|---|---|---|---|---|
| eil76 | 2 | cost | **558.59** | 594.21 ±0.93 | 580.77 ±2.91 | 583.41 ±3.04 | 579.68 ±2.39 | 578.96 ±2.81 |
| | | amplitude | 112.98 | **15.39** | 97.93 | 92.85 | 47.13 | 56.07 |
| | 3 | cost | **579.30** | 642.89 ±0.98 | 622.91 ±3.41 | 630.67 ±5.09 | 613.76 ±3.26 | 619.19 ±4.12 |
| | | amplitude | **15.99** | 94.64 | 96.18 | 103.34 | 64.87 | 69.18 |
| | 5 | cost | **[623.88, 680.67]** | 740.35 ±0.23 | 747.49 ±4.75 | 760.05 ±5.02 | 734.61 ±3.66 | 744.94 ±4.39 |
| | | amplitude | 126.55 | **57.82** | 128.36 | 125.19 | 76.37 | 81.80 |
| | 7 | cost | **[675.38, 759.90]** | 820.35 ±0.10 | 873.65 ±6.61 | 883.63 ±5.44 | 894.70 ±4.68 | 911.06 ±5.00 |
| | | amplitude | 99.62 | **85.12** | 157.59 | 158.02 | 93.92 | 100.20 |

Table 5: The performance of the ACS variants for the rat99 instance

| problem | $m$ | measure | optimum | kM-ACS | g-ACS | s-ACS | gb-ACS | sb-ACS |
|---|---|---|---|---|---|---|---|---|
| rat99 | 2 | cost | **[1296.35, 1350.73 ]** | 1485.56 ±2.98 | 1398.01 ±8.72 | 1391.89 ±7.35 | 1382.05 ±4.45 | 1389.08 ±5.45 |
| | | amplitude | 63.51 | 167.15 | 201.06 | 163.57 | **59.87** | 68.18 |
| | 3 | cost | **[1357.30, 1519.49]** | 1672.11 ±3.33 | 1691.56 ±11.64 | 1707.20 ±12.07 | 1661.04 ±8.89 | 1651.68 ±11.57 |
| | | amplitude | 243.63 | 360.63 | 336.76 | 352.97 | **144.33** | 151.61 |
| | 5 | cost | **[1523.95, 1855.83]** | 1996.04 ±1.23 | 2260.74 ±14.03 | 2297.05 ±16.83 | 2286.73 ±16.12 | 2337.94 ±19.45 |
| | | amplitude | 396.28 | 469.82 | 399.53 | 417.60 | 242.86 | **238.43** |
| | 7 | cost | **[1712.1467, 2291.8207]** | 2361.55 ±0.74 | 2859.98 ±22.36 | 2878.97 ±21.11 | 3004.37 ±26.22 | 2984.42 ±17.38 |
| | | amplitude | 422.92 | 435.21 | 407.19 | 425.72 | 288.40 | **281.99** |

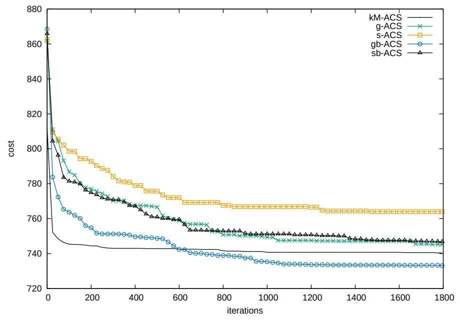

Fig. 1: Evolution of the cost of the best solution during the run of the algorithms for eil76-m5, averaged over 10 runs.

the 4 distinct settings of the number of salesmen (2,3,5 and 7). The increase in the total cost with the increase of parameter $m$ is evident. For each of the four groups in each chart, we have 6 sets of solutions: corresponding to the best known solution - obtained with CPLEX (this behaves as lower bound for the ACS-based algorithms), to $kM-ACS$, to the two versions of ACS adapted for multiple-TSP that do not enforce bounds on the size of the subtours and the two ACS versions with bounds.

Evident in the boxplots but also from the size of the confidence intervals, kM-ACS is the most stable algorithm. With a quick convergence, this algorithm even achieves on some problem instances (eil51-m5, eil76-m7, rat99-5, rat99-7) the best cost (proven by statistical tests on the means). It seems that this algorithm is a good choice for very large instances, with high number of cities and many salesmen.

Among the two bounded variants - $gb-ACS$ and $sb-ACS$ - $gb-ACS$ converges more quickly, achieving at the same time better solutions.

Comparing the two unbounded variants - $g-ACS$ and $s-ACS$ - the same conclusion can be drawn: the version laying equal pheromone quantities on the edges of the best solutions (irrespective of the cost of the subtour) achieves better costs. However, $gb-ACS$ returns solutions which are highly unbalanced compared to $s-ACS$ and this is a real concern for its applicability in practice.

## 6 Conclusions

Inspired by the ant colonies behavior in nature, the Ant Colony System proved to be an efficient approach for searching for an optimal path in a graph. As shown in our study, this meta-heuristic can be easily adapted for the multiple traveling salesman problem. For real world, large instances of multiple-TSP, which become untractable with exact deterministic algorithms, the ant colony system is a viable solution. Future work will be conducted towards studying the bi-objective formulation of multiple-TSP, by means of ACSs.

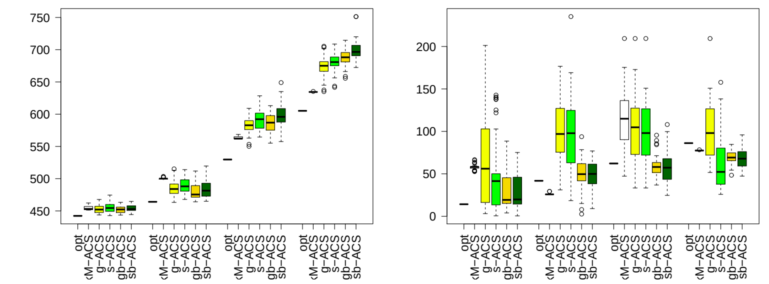

Fig. 2: Results obtained in 50 runs for eil51: a) total cost, b) amplitude of the costs of subtours; the groups correspond to different settings for $m$: 2, 3, 5, 7

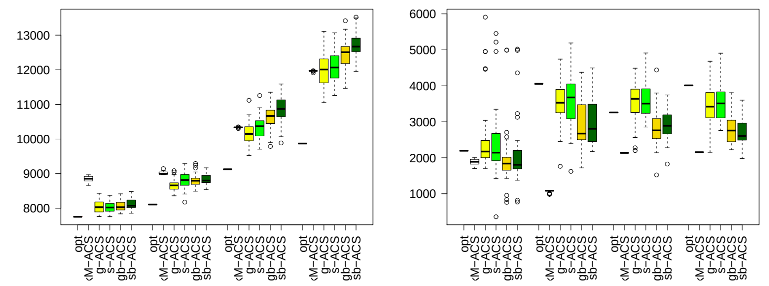

Fig. 3: Results obtained in 50 runs for berlin52: a) total cost, b) amplitude of the costs of subtours; the groups correspond to different settings for $m$: 2, 3, 5, 7

## References

1. Kara, I., Bektas, T.: Integer linear programming formulations of multiple salesman problems and its variations. European Journal of Operational Research, vol. 174 (3), 1449 - 1458 (2006)
2. Laporte, G.G., Nobert, Y.A.: Cutting planes algorithm for the m-salesmen problem. Journal of the Operational Research Society, 31, 1017-1023 (1980)
3. Ali, A., Kennington, J.L.: Exact solution of multiple traveling salesman problems. Discrete Applied Mathematics, 13, 259-276 (1986)
4. Russell, R.A.: An effective heuristic for the m-tour traveling salesman problem with some side conditions. Operations Research, 25(3), 517-524 (1977)
5. Ryan, J.L., Bailey, T.G., Moore, J.T., Carlton, W.B.: Reactive Tabu search in unmanned aerial reconnaissance simulations. In Proceeding of the 1998 winter simulation conference, 873-882 (1998)
6. Carter, A.E., Ragsdale, C.T.: A new approach to solving the multiple traveling salesperson problem using genetic algorithms. European Journal of Operational Research, vol. 175(1), 246 - 257 (2006)

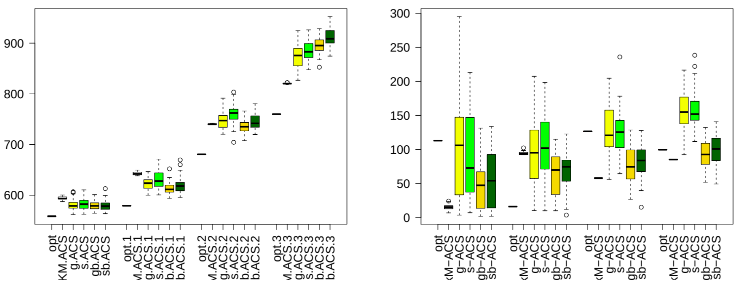

Fig. 4: Results obtained in 50 runs for eil76: a) total cost, b) amplitude of the costs of subtours; the groups correspond to different settings for $m$: 2, 3, 5, 7

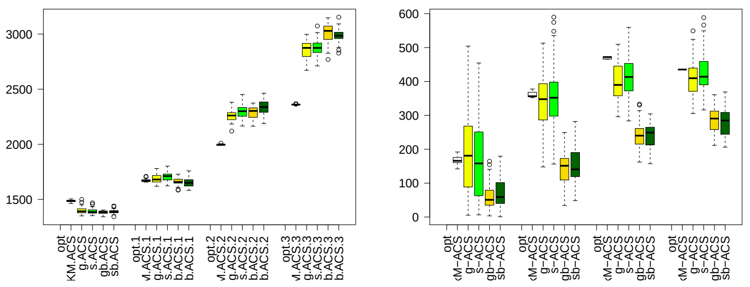

Fig. 5: Results obtained in 50 runs for rat99: a) total cost, b) amplitude of the costs of subtours; the groups correspond to different settings for $m$: 2, 3, 5, 7

7. Liu, W., Li, S., Zhao, F., Zheng, A.: An ant colony optimization algorithm for the Multiple Traveling Salesmen Problem. In Proceedings of 4th IEEE Conference on Industrial Electronics and Applications ICIEA 2009, 1533-1537 (2009)
8. Somhom, S., Modares, A., Enkawa, T.: Competition-based neural network for the multiple travelling salesmen problem with minmax objective. Computers & Operations Research, 26, 395-407 (1999)
9. Junjie, P., Dingwei, W.: An Ant Colony Optimization Algorithm for Multiple Travelling Salesman Problem. First International Conference on Innovative Computing, Information and Control ICICIC '06, vol. 1, 210–213 (2006)
10. Sofge, D., Schultz, A., De Jong, K.: Evolutionary Computational Approaches to Solving the Multiple Traveling Salesman Problem Using a Neighborhood Attractor Schema. Applications of Evolutionary Computing, editor S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, G. Raidl, LNCS 2279, Springer Berlin Heidelberg, 153-162 (2002)
11. Chandran, N., Narendran, T.T., Ganesh, K.: A clustering approach to solve the multiple traveling salesmen problem. International Journal of Industrial and Systems Engineering, vol. 1 (3), 372-387 (2006)
12. Dorigo, M., Gambardella, L.M.: Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary

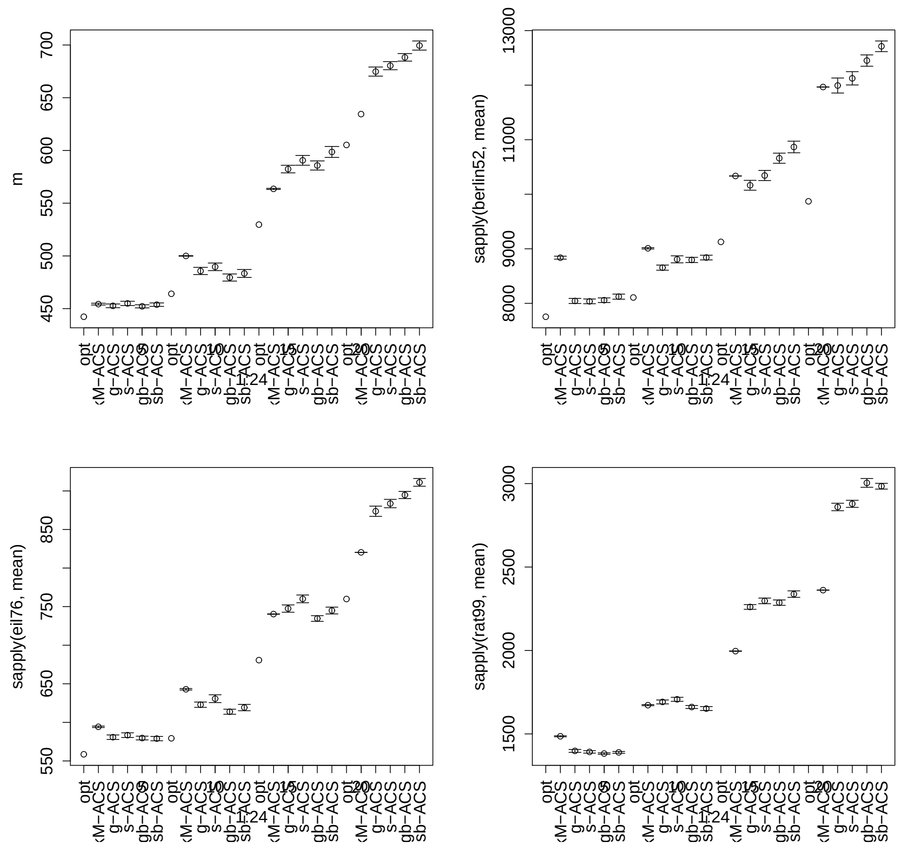

Fig. 6: Results obtained in 50 runs for a) eil51, b) berlin52, c) eil76 and d) rat99: confidence intervals for the means computed for total costs; the four groups on each plot correspond to different settings for $m$: 2, 3, 5, 7