We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

5,000
Open access books available

125,000
International authors and editors

140M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Chapter

Multi-Strategy *MAX-MIN* Ant System for Solving Quota Traveling Salesman Problem with Passengers, Incomplete Ride and Collection Time

Bruno C.H. Silva, Islame F.C. Fernandes, Marco C. Goldbarg and Elizabeth F.G. Goldbarg

Abstract

This study proposes a novel adaptation of *MAX-MIN* Ant System algorithm for the Quota Traveling Salesman Problem with Passengers, Incomplete Ride, and Collection Time. There are different types of decisions to solve this problem: satisfaction of the minimum quota, acceptance of ride requests, and minimization of travel costs under the viewpoint of the salesman. The Algorithmic components proposed regards vehicle capacity, travel time, passenger limitations, and a penalty for delivering a passenger deliverance out of the required destination. The ant-based algorithm incorporates different sources of heuristic information for the ants and memory-based principles. Computational results are reported, showing the effectiveness of this ant-based algorithm.

Keywords: Traveling Salesman, integer programming, transportation, shared mobility, Ant Colony Optimization

1. Introduction

The lives of ordinary consumers have changed almost beyond recognition in the past 20 years. First, with the introduction of high-speed internet access; but, more recently, with the arrival of mobile computing devices such as smartphones and tablets. According to data from the 2017 Gallup World Survey [1], 93 of adults in high-income economies have their cell phones, while 79% in developing economies. In India, 69% of adults have a cell phone, as well as 85% in Brazil and 93% in China [1]. Smartphones and the internet have created a novel digital ecosystem where the adoption of new paradigms is increasingly fast, and each innovation that appears and presents itself to the market can disrupt an entire segment.

In the transportation segment, a central theme is how the digital revolution has created opportunities to consider new models of delivering services under the paradigm of Mobility as a Service (MaaS) [2]. There is a growing interes%t in *MaaS* due to the notion of a sharing economy. Millennials own fewer vehicles than previous generations [3]. As evidenced by the ascension of on-demand mobility

platforms, they are quickly adopting car sharing as a mainstream transportation solution. Investments in new travel patterns have become a priority to enable the transformation of opportunities in the urban mobility segment into new revenue streams.

This study deals with a novel optimization model that can improve the services provided by on-demand mobility platforms, called Quota Traveling Salesman Problem with Passengers, Incomplete Ride, and Collection Time (QTSP-PIC). In this problem, the salesman is the vehicle driver and can reduce travel costs by sharing expenses with passengers. He must respect the budget limitations and the maximum travel time of every passenger. Each passenger can be transported directly to the desired destination or an alternate destination. Lira et al. [4] suggest pro-environmental or money-saving concerns can induce users of a ride-sharing service to agree to fulfill their needs at an alternate destination.

The QTSP-PIC can model a wide variety of real-world applications. Cases related to sales and tourism are the most pertinent ones. The salesman must choose which cities to visit to reach a minimum sales quota, and the order to visit them to fulfill travel requests. In the tourism case, the salesman is a tourist that chooses the best tourist attractions to visit during a vacation trip and can use a ride-sharing system to reduce travel expenses. In both cases, the driver negotiates discounts with passengers transported to a destination similar to the desired one.

The QTSP-PIC was introduced by Silva et al. [5]. They presented a mathematical formulation and heuristics based on Ant Colony Optimization (ACO) [6]. To support the ant algorithms, they proposed a Ride-Matching Heuristic (RMH) and a local search with multiple neighborhood operators, called Multi-neighborhood Local Search (MnLS). They tested the performances of the ant algorithms on 144 instances up to 500 vertices. One of these algorithms, the Multi-Strategy Ant Colony System (MS-ACS), provided the best results. They concluded that their most promising algorithm could improve with learning techniques to choose the source of information regarding the instance type and the search space.

In this study, a *MAX-MIN* Ant System (*MM*AS) adaptation to the QTSP-PIC, called Multi-Strategy *MAX-MIN* Ant System (MS-*MM*AS), is discussed. *MM*AS improves the design of Ant System [6], the first ACO algorithm, with three important aspects: only the best ants are allowed to add pheromone during the pheromone trail update; use of a mechanism for limiting the strengths of the pheromone trails; and, incorporation of local search algorithms to improve the best solutions. Plenty of recent studies proved good effectiveness of the *MM*AS in correlated problems to QTSP-PIC [7–10]. However, none of these explored the Multi-Strategy (MS) concept.

In the traditional ant algorithms applied to Traveling Salesman Problem (TSP), ants use the arcs' cost as heuristic information [6]. The heuristic information adopted is called visibility. When solving the QTSP-PIC, different types of decisions must be considered: the accomplishment of the minimum quota, management of the ride requests, and minimization of travel costs. The MS idea is to use different mechanisms of visibility for the ants to improve diversification. Every ant decides which strategy to use at random with uniform distribution. The MS proposed in this study extends the original implementation proposed in [5]. MS-*MM*AS also incorporates RMH and MnLS and uses a memory-based technique proposed in [11] to avoid redundant work. In MS-*MM*AS, a hash table stores every solution constructed and used as initial solutions to a local algorithm. When the algorithm constructs a new solution, it starts the local search phase if the new solution is not in the hash table.

The benchmark for the tests consisted of 144 QTSP-PIC instances. It was proposed by Silva et al. [5]. Numerical results confirmed the effectiveness of the MS-*MM*AS by comparing it to other ACO variants proposed in [5].

The main contributions of this chapter are summarized in the following.

- The extension of the MS concept proposed in [5] with a roulette mechanism that orients the ants to choose their heuristic information based on the best quality solutions achieved;
- Improvement of the *MM*AS design with a memory based technique proposed in [11];
- Presentation of a novel *MM*AS variant that combines the improved MS concept and memory-based principles and assessment of its performance;
- Experiments on a set of QTSP-PIC instances ranging: 10 to 500 cities; and 30 to 10.000 travel requests. The results showed that the proposed *MM*AS variant is competitive regarding the other three ACO variants presented in [5] for the QTSP-PIC.

The remainder of this chapter is organized as follows. Section 2 presents the QTSP-PIC and its formulation. Section 3 presents the Ant Colony Optimization metaheuristic and the implementation design of the MS-*MM*AS. Section 4 presents experimental results. The performance of the proposed ant-based algorithm is discussed in Section 5. Conclusions and future research directions are outlined in Section 6.

2. Problem definition

The TSP can be formulated as a complete weighted directed graph $G = (N, A)$ where N is the set of vertices and $A = \{(i,j) \mid i,j \in N\}$ is the set of arcs. $C = \left[c_{ij}\right]$ is the arc-weight matrix such that c_{ij} is the cost of arc (i,j). The objective is to determine the shortest Hamiltonian cycle in G. Due to its applicability, many TSP variants deal with specific constraints [12]. Awerbuch et al. [13] presented several quota-driven variants. One of them, called Quota Traveling Salesman Problem (QTSP), is the basis for the problem investigated in this study. In the QTSP, there is a bonus associated with each vertex of G. The salesman has to collect a minimum quota of bonuses in the visited vertices. Thus the salesman needs to figure out which cities to visit to achieve the minimum quota. The goal is to find a minimum cost tour such that the sum of the bonuses collected in the visited vertices is at least the minimum quota.

The QTSP-PIC is a QTSP variant in which the salesman is the driver of a vehicle and can reduce travel costs by sharing expenses with passengers. There is a travel request, associated with each person demanding a ride, consisting of a pickup and a drop off point, a budget limit, a limit for the travel duration, and penalties associated with alternative drop-off points. There is a penalty associated with each point different from the destination demanded by each person. The salesman can accept or decline travel requests. This model combines elements of ride-sharing systems [14] with alternative destinations [4], and the selective pickup and delivery problem [15].

Let $G(N, A)$ be a connected graph, where N is the set of vertices and $A = \{(i,j) \mid i,j \in N\}$ is the set of arcs. Parameter q_i denotes the quota associated with vertex $i \in N$ and g_i the time required to collect the quota. c_{ij} and t_{ij} denote, respectively, the cost and time required to traverse edge $(i,j) \in A$. Let L be the set of passengers. List $l_i \subseteq L$ denotes the subset of passengers who depart from $i \in N$. Let $org(l)$ and $dst(l) \in N$ be the pickup and drop-off points requested by passenger l. The

salesman departs from city $s = 1$, visits exactly once each city of subset $N' \subseteq N$ and returns to s. The quota collected by the salesman must be at least K units. Along the trip, the salesman may choose which travel requests to fulfill. The travel costs are shared with vehicle occupants. The number of vehicle occupants, or passengers, cannot exceed R. Each passenger $l \in L$ imposes a budget limit w_l and a trip's maximum duration b_l. Let h_{lj} be the penalty to deliver passenger $l \in L$ at city $j \in N$, $j \neq dst(l)$. The value of variable h_{lj} is computed in the final cost of the tour if passenger l is delivered to city j. If $j = dst(l)$, then $h_{lj} = 0$. The objective of the QTSP-PIC is to find a Hamiltonian cycle $\Psi = (N', A')$ such that the ride-sharing cost and eventual penalties are minimized, and the quota constraint is satisfied.

The QTSP is NP-hard [13]. It is a particular case of the QTSP-PIC, in which the list of persons demanding a ride is empty and the time spent to collect the bonus in each vertex is zero. Thus, QTSP-PIC also belongs to the NP-hard class.

Silva et al. [5] presented an integer non-linear programming model for the QTSP-PIC. They defined a solution as $S = (N', Q', L', H')$, where N' is a list of vertices that represents a cycle, Ψ, such that the minimum quota restriction, K, is satisfied; Q' is a binary list in which the i-th element is 1 if the salesman collects the bonus from city i, $i \in N'$; L' is a binary list in which the l-th element is 1 if the salesman accepts the l-th travel request; and H' is a list of integers in which the l-th element is the index of the city where the l-th passenger leaves the car. If $L'[l] = 0$, then $H'[l] = 0$. The cost of solution S, denoted by $S.cost$, is calculated by Eq. (1).

$$S.cost = \sum_{i,j \in N'} \frac{c_{ij}}{1 + \sum_{l \in L'} v_{ij}^{l}} + \sum_{l \in L'} h_{lH'_l} \tag{1}$$

3. Ant Colony Optimization

In the Ant Colony Optimization, artificial ants build and share information about the quality of solutions achieved with a communication scheme similar to what occurs with some real ants species. Deneubourg et al. [16] investigated the behavior of Argentine ants and performed some experiments, where there were two bridges between the nest and a food source. He observed that the ants initially walked on the two bridges at random, depositing pheromone in the paths. Over time, due to random fluctuations, the pheromone concentration of one bridge was higher than the other. Thus, more ants were attracted to that route. Finally, the whole colony ended up converging towards the same route. The behavior of artificial ants preserves four notions of the natural behavior of ants:

- Pheromone deposit on the traveled trail;
- Predilection for trails with pheromone concentration;
- Concentration of the amount of pheromone in shorter trails;
- Communication between ants through the pheromone deposit.

Pheromone is a chemical structure of communication [17]. According to Dorigo et al. [18], pheromone enables the process of stigmergy and self-organization in which simple agents perform complex and objective-oriented behaviors. Stigmergy is a particular form of indirect communication used by social insects to coordinate their activities [18].

Considering the context of ant algorithms applied to the TSP, when moving through the graph G, artificial ants tend to follow paths with higher pheromone deposits rates. As ants tend to deposit pheromone along the path they follow, as more ants choose the same path, the pheromone rate tends to increase in these paths. This cooperation mechanism induces artificial ants to find good solutions, as it works as a shared memory that is continuously updated and can be consulted by every ant in the colony [19].

The base-line of the Ant Colony Optimization is the algorithm Ant System [6]. In the TSP application, N is the set of vertices to visit. Ants construct solutions iteratively. Every iteration, the ant chooses the next vertex based on heuristic information, η, and pheromone trails, τ. Initially, pheromone trails have the same amount of pheromone, τ_0, computed by Eq. (2), where n is the number of vertices in N and $Cost(D)$ is the value of the TSP tour built by a greedy heuristic.

$$\tau_{ij}^0 = (n \times Cost(D))^{-1} \tag{2}$$

The k-th ant iteratively adds new vertices to the solution. The algorithm uses Eq. (3) to compute the probability of the k-th ant to move from vertex i to j at the t-th iteration, where $\eta_{ij} = \frac{1}{c_{ij}}$ is the heuristic factor, $\tau_{ij}(t)$ is the pheromone in arc (i,j) in the t-th iteration, and Λ^k is the list of vertices not visited by the k-th ant. Coefficients α and β weight the influence of the pheromone and heuristic information, respectively. They are user-defined parameters. If $\alpha = 0$, the probability computed by Eq. (3) depends only on the heuristic information. So, the ant algorithm behaves like a greedy method. If $\beta = 0$, ants tend to select paths with higher pheromone levels. It may lead the algorithm to early stagnation. So, balancing the values of α and β is critical to guarantee a suitable search strategy [5].

$$\Upsilon_{ij}^k(t) = \frac{\left[\tau_{ij}(t)\right]^\alpha \cdot \left[\eta_{ij}\right]^\beta}{\sum_{w \in \Lambda^k}\left[\left[\tau_{iw}(t)\right]^\alpha \cdot \left[\eta_{iw}\right]^\beta\right]}, \qquad j \in \Lambda^k \tag{3}$$

Eqs. (4) and (5) show the formulas used to update pheromone trails, where ρ is the evaporation coefficient, $Cost\left(W^k\right)$ is the cost of the route W^k built by the k-th ant, and $\Delta\tau_{ij}^k$ is the pheromone deposited on arc (i,j) by the k-th ant, computed by expression (6).

$$\tau_{ij} = (1-\rho) \times \tau_{ij} + \rho \times \Delta\tau_{ij}, \qquad \rho \in [0,1] \tag{4}$$

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^k \tag{5}$$

$$\Delta\tau_{ij} = \begin{cases} \dfrac{1}{Cost\left(W^k\right)}, & \text{if arc } (i,j) \in W^k. \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

The ant algorithms proposed after AS improved its implementation design with elitist pheromone update strategies and local search algorithms to improve solutions [6]. Two well-known variants of AS are the Ant Colony System (ACS) [20] and *MAX-MIN* Ant System [21]. Silva et al. [5] presented AS and ACS adaptations for the QTSP-PIC. Section 3.1 presents the *MAX-MIN* Ant System algorithm.

3.1 Multi-strategy *MAX-MIN* ant system

*MM*AS uses Eq. (3) to compute the probability of an ant to move from vertex i to j. Besides, it incorporates improvements to avoid search stagnation and a pheromone update rule that limits pheromone concentration rates. Eq. (7) presents the pheromone update rule. Limits τ_{max} and τ_{min} prevent stagnation of pheromone values.

$$\tau_{ij} = max\left\{\tau_{min}, min\left\{\tau_{max}, (1-\rho)\times\tau_{ij} + \rho\times\Delta\tau_{ij}^{best}\right\}\right\}, \qquad \rho\in[0,1] \tag{7}$$

$$\Delta\tau_{ij}^{best} = \begin{cases} \dfrac{1}{Cost\left(W^{best}\right)}, & \text{if arc}(i,j)\in W^{best}. \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

There are three possibilities for the best route (W^{best}) considered in the algorithm: the best route in the current iteration, the best route found so far, and the best route since the last time pheromone trails were reinitiated. In the *MM*AS original design [21], these routes were chosen alternately. The initial value of pheromone trails was t_{max}. If the algorithm reached stagnation, i.e., the best current route remained the same for several iterations, the pheromone value reinitialized to t_{max}. Assigning t_{max} to pheromone trails produces a small variability among pheromone levels at the start of the search [21].

The implementation of the *MM*AS for the QTSP-PIC extends the original proposal [21] with the following adaptions:

- Ants start at vertex s;
- Ants include vertices in the route up to reach the minimum quota;
- Solution S^k, built by the k-th ant, is computed by assigning passengers to route W^k with the *RMH* algorithm [5];
- Use of the MS concept.

The ants in the *MM*AS, use arc costs to compute heuristic information. In the MS-*MM*AS, ants use four sources for this task, listed in the following.

- Cost oriented: uses c_{ij} as heuristic information, such that $\eta_{ij} = \frac{1}{c_{ij}}$;
- Time oriented: uses t_{ij} as heuristic information, such that $\eta_{ij} = \frac{1}{t_{ij}}$. This heuristic information guides ants to vertices that lead to travel time savings.
- Quota oriented: q_j is used as heuristic information, $\eta_{ij} = \frac{q_j}{c_{ij}}$. This heuristic information guides ants to go to vertices that lead to the maximization of the quota collected.
- Passenger oriented: the heuristic information is $\eta_{ij} = \frac{|L_j|}{c_{ij}}$. This strategy orients ants to maximize the number of travel requests fulfilled.

In the MS concept proposed in [5], every ant decides which strategy to use at random with uniform distribution. A roulette wheel selection improves this

concept. The proportion of the wheel assigned to each heuristic information is directly related to the quality of solutions achieved. So, ants learn, at each iteration, the best heuristic information. At the final iterations, ants tend to use the heuristic information that proved to be most promising.

Algorithm 1 presents the pseudo-code of the MS-*MM*AS. It has the following parameters: maximum number of iterations (*maxIter*), number of ants ($m \in \mathcal{Z}_{>0}$), pheromone coefficient ($\alpha \in \mathcal{R}_{>0}$), heuristic coefficient ($\beta \in \mathcal{R}_{>0}$), evaporation factor ($\rho \in [0, 1]$), and pheromone limits ($\tau_{max}, \tau_{min} \in \mathcal{R}_{>0}$). It also has the following parameters and variables:

- N: set of vertices;
- ξ: index of the heuristic information source;
- W^k: route built by the k-th ant;
- S^k: solution produced after applying the *RMH* heuristic [5] to route W^k;
- W^i: the best route built in the i-th iteration;
- S^i: the best solution produced in the i-th iteration;
- W^*: the best route found so far;
- S^*: the best solution found so far;
- W^{best}: route used as input to the pheromone updating procedure;
- Π: hash table that stores every solution S^i constructed and used as initial solution to the local search algorithm;

Algorithm 1: MS-*MM*AS($maxIter$, m, α, β, ρ τ_{max}, τ_{min})

```
1. Π ← ∅
2. Initialize pheromone trails
3. For k = 1 to m.
4.     W^k[2] ← random_city(N\{s})
5. For i = 1 to maxIter
6.   For k = 1 to m
7.       ξ ← chose_heuristic_information()
8.       W^k ← build_route(α, β, ξ)
9.       S^k ← assign_passengers(W^k)
10.      Update(W^i,S^i)
11.  If S^i ∉ Π
12.      S^i ← MnLS(S^i)
13.      Store(Π, S^i)
14.  Update(W*,S*)
15.  W^best ← alternate(maxIter, i, W^i, W*)
16.  Pheromone_update(W^best,ρ, τ_max, τ_min)
17. Return S*
```

The algorithm sets τ_{max} as the initial value of pheromone trails (step 2). Since ants begin at vertex s, the second vertex is selected randomly with uniform distribution (steps 3 and 4). The k-th ant decides which heuristic information, ξ, to use (step 7) and builds a route (step 8). The algorithm uses the *RMH* heuristic to assign passengers to W^k, completing a solution (step 9). The algorithm updates W^i and S^i (step 10). The MnLS algorithm is applied to S^i (step 12) if the solution $S^i \notin \Pi$. After the local search, the algorithm stores S^i in the hash table Π. At the next iteration, the current S^i is the starting solution of the local search if it is not in Π. This procedure prevents redundant work. The algorithm updates the best route and the best solution found so far, W^* and S^* (step 14). Similar to the original design of *MM*AS, W^i is assigned to W^{best} at the first 25% iterations or if i ranges from [50%,75%] of *maxIter*. W^* is assigned to W^{best} if i ranges from [25%,50%] of *maxIter* or if it is greater than or equal to 75% of *maxIter* (step 15). This procedure improves diversification by shifting the emphasis over the search space. W^{best} is used to update pheromones (step 16). Finally, the algorithm returns S^*.

4. Experiments and results

This section presents the methodology for the experiments and results from the experiments. Section 4.1 presents the methodology. Section 4.2 presents the parameters used in the MS-*MM*AS algorithm. Section 4.3 presents the results.

4.1 Methodology

The experiments were executed on a computer with an Intel Core i5, 2.6 GHz processor, Windows 10, 64-bit, and 6GB RAM memory. The algorithms were implemented in C ++ lan and compiled with GNU g++ version 4.8.4. The benchmark set proposed in [5] was used to test the effectiveness of the MS-*MM*AS. The sizes of those instances range from 10 to 500 vertices. Small instances have up to 40 vertices, medium up to 100, and large more than 100 vertices. The instances are available for download at https://github.com/brunocastrohs/QTSP-PIC.

The best, average results, and average processing times (in seconds) are reported from 20 independent executions of the MS-*MM*AS. Experiments are conducted to report the distance between the best-known solutions and the best results provided by the MS-*MM*AS. The variability in which the MS-*MM*AS achieved the best-known solutions stated in the benchmark set is also calculated. With these experiments, it is possible to conclude if the MS-*MM*AS algorithm was able to find the best-known solution of each instance and with what variability this happens.

The Friedman test [23] with the Nemenyi post-hoc procedure [24] are applied, with a significance level 0.05, to conclude about significant differences among the results of the MS-*MM*AS and the other three ACO variants proposed in this [5]. The instances were grouped according to their sizes (number of vertices) for the Friedman test. There are eight groups of symmetric (asymmetric) instances, each of them contains nine instances, called $g<n>$, where $<n>$ stands for the size.

4.2 Parameter tuning

The IRACE software was used, presented by [22], to tune the parameters of the MS-*MM*AS algorithm. 20 symmetric and 20 asymmetric instances were submitted

to adjust the parameters. Those instances were selected at random. The IRACE uses the *maxExperiments* and *maxTime* parameters as stopping criteria. This parameters were set as follows: $maxExperiments = 10^3$; and, $maxTime = \infty$.

For the asymmetric instance set, the parameters were defined as follows: $maxIter = 31$; $m = 51$; $\alpha = 3.08$; $\beta = 10.31$; $\rho = 0.52$; $\tau_{max} = 0.8$; and $\tau_{min} = 0.2$. For the symmetric instance set, the parameters were: $maxIter = 29$; $m = 57$; $\alpha = 2.92$; $\beta = 9.53$; $\rho = 0.67$; $\tau_{max} = 0.7$; and $\tau_{min} = 0.2$.

4.3 Results

In this section, the results of the MS-*MM*AS are tested and compared to those produced by the other three ACO variants proposed in [5]: AS, ACS, and MS-ACS.

Table 1 presents the comparison between the ant algorithms. The best results obtained by MS-*MM*AS were compared with those achieved by each ant algorithm proposed in [5]. The results are in the $X \times Y$ format, where X and Y stand for the number of instances in which the ant algorithm X found the best solution and the number of instances in which the ant algorithm Y found the best solution, respectively.

Table 1 shows that the MS-*MM*AS was the algorithm that reported the best solution for most instances. This algorithm performed best than other ACO variants due to its enhanced pheromone update procedures. The MS implementation with roulette wheel selection proved to be effective at finding the best heuristic information used by the ants during the run. **Table 1** also shows that the MS-*MM*AS provides results with better quality than the MS-ACS in the most symmetric cases. The MS-ACS was superior to the MS-*MM*AS in seventeen asymmetric cases and fourteen symmetric instances. Was observed that the pseudo-random action choice rule of MS-ACS [20], which allows for a greedier solution construction, proved to be a good algorithmic strategy for solving large instances.

Tables 2 and **3** shows the ranks of the ant algorithms based on the Friedman test [23] with the Nemenyi [24] post-hoc test. The first column of this Tables presents the subsets of instances grouped according to their sizes. The other columns of this Tables present the p-values of the Friedman test and the ranks from the Nemenyi post-hoc test. In the post-hoc test, the order ranks from a to c. The c rank indicates that the algorithm achieved the worst performance in comparison to the others. The a rank indicates the opposite. If the performances of two or more algorithms are similar, the test assigns the same rank for them. In this experiment, the significance level was assigned with 0.05.

The p-values presented in **Tables 2** and **3** show that the performance of the ant algorithms was not similar, i.e., the null hypothesis [24] is rejected in all cases. In these Tables, can be observed that MS-*MM*AS ranks higher than AS and ACS for all subsets. The ranks of MS-ACS and MS-*MM*AS were the same in the most cases. This implies that the performance of only these two algorithms where similar, i. e., the relative distance between the results achieved by these two algorithms are short.

To analyze the variability of the results provided by each ant algorithm compared to the best results so far for the benchmark set, three metrics regarding the

	Asymmetric			**Symmetric**		
	AS	**ACS**	**MS-ACS**	**AS**	**ACS**	**MS-ACS**
MS-*MM*AS	68 x 0	68 x 1	45 x 17	66 x 1	68 x 2	48 x 14

Table 1.
Comparison between the ant algorithms.

		Asymmetric			
Subset	**p-value**	**AS**	**ACS**	**MS-ACS**	**MS-*MM*AS**
*g*10	0.003159	b	b	a	a
*g*20	0.000040	b	c	a	a
*g*30	0.000017	c	c	a	a
*g*40	0.000024	b	c	a	a
*g*50	0.000048	c	c	b	a
*g*100	0.000045	b	c	a	a
*g*200	0.000031	b	c	a	a
*g*500	0.000037	c	b	a	a

Table 2.
Results of Friedman's test and Nemenyi post-hoc test over asymmetric instances set.

		Symmetric			
Subset	**p-value**	**AS**	**ACS**	**MS-ACS**	**MS-*MM*AS**
*g*10	0.003543	b	b	a	a
*g*20	0.000205	b	c	a	a
*g*30	0.000045	c	c	b	a
*g*40	0.000059	b	c	a	a
*g*50	0.000035	b	b	a	a
*g*100	0.000024	b	b	a	a
*g*200	0.000045	c	b	a	a
*g*500	0.000098	c	b	a	a

Table 3.
Results of Friedman's test and Nemenyi post-hoc test over symmetric instances set.

	Asymmetric			
Metric	**AS**	**ACS**	**MS-ACS**	**MS-*MM*AS**
ν	4.30%	2.56%	6.8%	11.84%
Φ	0.2075333	0.2773624	0.0541799	0.0054741
Ω	0.2835401	0.4023659	0.1754952	0.5854892

Table 4.
Variability of the ants algorithms for asymmetric instances.

	Symmetric			
Metric	**AS**	**ACS**	**MS-ACS**	**MS-*MM*AS**
ν	2.01%	0.69%	8.75%	9.05%
Φ	0.2169756	0.2285948	0.0547890	0.0113889
Ω	0.3017957	0.3620682	0.1656022	0.6432748

Table 5.
Variability of the ants algorithms for symmetric instances.

	Asymmetric				Symmetric			
Instance	**Best**	**Average**	**Time**	**Percentage**	**Best**	**Average**	**Time**	**Percentage**
A-10-3	478.42	863.13	0.15	100%	545.92	996.34	0.15	25%
A-10-4	523.57	1069.33	0.14	80%	460.00	838.84	0.18	20%
A-10-5	482.60	690.08	0.14	5%	371.93	658.97	0.19	10%
A-20-3	519.67	936.47	0.35	5%	679.75	1363.59	0.32	20%
A-20-4	458.10	1145.88	0.38	0%	346.30	661.82	0.43	35%
A-20-5	398.75	669.82	0.35	10%	351.50	1006.24	0.48	5%
A-30-3	618.33	1180.70	0.48	10%	574.33	1469.68	0.50	5%
A-30-4	401.20	805.32	1.28	5%	654.80	1202.15	0.40	5%
A-30-5	475.83	1033.91	0.58	10%	464.05	911.56	0.66	5%
A-40-3	692.00	1060.67	2.83	5%	718.25	1399.04	0.72	10%
A-40-4	658.95	1088.41	2.52	5%	570.98	961.13	2.95	5%
A-40-5	460.90	900.51	3.11	5%	441.22	836.52	2.89	5%
B-10-3	729.50	925.35	0.13	5%	834.67	1485.47	0.20	20%
B-10-4	306.90	421.35	0.13	15%	493.58	757.45	0.16	10%
B-10-5	434.75	835.01	0.18	55%	726.35	1160.97	0.23	5%
B-20-3	805.42	1251.39	0.28	10%	950.00	1666.22	0.45	5%
B-20-4	848.62	1366.69	0.35	5%	822.82	1386.67	0.49	5%
B-20-5	895.17	1275.78	0.28	70%	660.22	1215.12	0.35	5%
B-30-3	747.75	1316.96	1.31	5%	718.67	1358.11	0.93	5%
B-30-4	723.27	1301.57	1.51	5%	650.35	1272.86	0.69	5%
B-30-5	700.75	1205.96	1.39	5%	504.68	1091.11	0.89	5%
B-40-3	964.42	1574.00	2.02	0%	889.83	1682.76	1.97	5%
B-40-4	1195.62	2134.73	1.20	5%	743.82	1508.16	2.09	0%
B-40-5	819.28	1537.71	1.11	10%	749.82	1351.64	0.85	5%
C-10-3	359.25	604.70	0.17	55%	597.83	697.51	0.05	0%
C-10-4	307.10	514.66	0.20	5%	408.45	514.65	0.15	85%
C-10-5	566.58	783.35	0.17	10%	409.60	846.51	0.23	10%
C-20-3	650.25	978.84	0.46	10%	629.92	1063.99	0.36	0%
C-20-4	563.78	938.50	0.72	5%	441.65	1019.96	1.06	10%
C-20-5	739.22	1056.77	1.02	0%	711.87	1095.84	0.63	5%
C-30-3	837.58	1198.09	1.05	5%	830.17	1221.65	0.61	10%
C-30-4	754.10	1144.99	2.47	5%	745.92	1096.55	1.16	5%
C-30-5	560.18	998.28	2.29	10%	490.80	931.81	2.21	5%
C-40-3	1008.00	1541.54	2.59	5%	607.00	946.57	3.45	10%
C-40-4	695.30	1172.76	2.06	10%	699.80	1136.17	2.25	0%
C-40-5	623.33	1097.22	3.24	10%	475.67	898.80	7.82	0%

Table 6.
Results of the MS-MMAS executions for small instances.

	Asymmetric				Symmetric			
Instance	**Best**	**Average**	**Time**	**Percentage**	**Best**	**Average**	**Time**	**Percentage**
A-50-3	1058.33	2128.68	2.31	5%	1000.33	1943.13	2.91	5%
A-50-4	774.93	1473.57	4.48	5%	783.40	1345.19	3.53	5%
A-50-5	673.42	1314.54	4.16	5%	583.08	1008.33	2.59	0%
A-100-3	1431.42	2046.96	53.44	5%	1514.08	2292.08	15.66	20%
A-100-4	1456.47	2705.07	23.12	5%	1165.90	1595.91	37.11	5%
A-100-5	1106.17	1778.38	40.57	10%	980.28	1366.09	58.20	5%
A-200-3	2806.75	3610.22	424.60	0%	2793.33	3272.00	257.42	0%
A-200-4	2388.88	3196.15	381.07	0%	2199.45	2807.06	79.32	10%
A-200-5	1753.00	2286.08	1380.26	10%	2086.82	3237.85	55.37	10%
A-500-3	6878.42	7165.39	1641.92	33%	6331.75	7679.65	732.94	10%
A-500-4	5572.42	5637.26	4939.84	0%	5030.48	5080.66	913.82	0%
A-500-5	4389.92	4539.44	16990.77	50%	4610.95	4696.91	73.97	33%
B-50-3	1338.42	2356.24	5.82	10%	966.42	1770.88	4.60	5%
B-50-4	951.87	1757.61	5.00	5%	772.67	1490.01	1.47	5%
B-50-5	1083.18	1943.22	3.91	5%	692.42	1342.78	4.74	5%
B-100-3	1781.00	3115.78	30.96	5%	1803.33	3258.90	15.11	5%
B-100-4	1409.65	2467.02	61.76	5%	1648.58	3360.93	11.00	5%
B-100-5	1361.20	2734.24	39.89	5%	1018.37	1536.34	99.02	5%
B-200-3	3302.83	4840.94	317.37	0%	3016.67	4267.17	56.62	0%
B-200-4	2536.80	3477.13	681.75	0%	2326.97	3139.57	81.34	10%
B-200-5	2127.88	2814.24	906.74	0%	1893.67	2506.64	102.33	10%
B-500-3	6994.84	7203.61	3857.77	0%	6433.92	6475.67	126.51	0%
B-500-4	5419.87	5730.97	24183.87	100%	5191.77	5276.98	267.72	0%
B-500-5	4546.28	4643.03	21118.98	0%	4379.43	4494.11	164.08	33%
C-50-3	1201.92	1801.68	3.12	5%	829.75	1482.86	2.82	5%
C-50-4	937.25	1651.11	7.96	5%	901.40	1605.24	6.16	10%
C-50-5	609.60	1127.99	16.58	5%	766.48	1366.58	7.43	5%
C-100-3	1496.58	2001.76	47.62	0%	1364.00	1819.03	14.21	10%
C-100-4	1352.85	2458.82	32.32	5%	1099.00	1445.29	23.98	0%
C-100-5	1022.70	1466.42	127.11	0%	991.12	1472.23	35.21	5%
C-200-3	2629.00	3197.13	478.90	10%	2510.50	3171.66	65.07	10%
C-200-4	2184.85	2648.38	710.70	10%	2141.40	2741.11	52.59	0%
C-200-5	1881.03	2296.53	486.62	0%	1713.17	2127.84	126.85	10%
C-500-3	6528.08	6618.16	2484.52	0%	6023.42	6087.14	138.12	33%
C-500-4	5139.54	5298.21	8188.46	0%	4942.28	4958.64	65.19	33%
C-500-5	4286.45	4278.49	6061.78	0%	4167.67	4178.10	302.79	0%

Table 7.
Results of the MS-MMAS executions for medium and large instances.

results produced by the experiments were adopted. The first metric, ν, shows the percentages relative to the number of times an ant algorithm found the best-known solution along 20 independent executions. The second metric, Φ, is the relative distance between the cost of the best-known solution χ^* and the best solution χ^{min} of each ant algorithm. The third metric, Ω, is the relative distance between χ^* and the average solution χ^a of each ant algorithm. To calculate Φ, the Eq. (9) was used. Ω is calculated using the formula (10). The average values of ν, Φ and Ω are reported in **Tables 4** and **5**.

$$\Phi = \frac{\chi^{min}}{\chi^*} - 1 \tag{9}$$

$$\Omega = \frac{\chi^a}{\chi^*} - 1 \tag{10}$$

It can be observed from **Tables 4** and **5** that the MS-*MM*AS is the best one concerning the ν and Φ metrics. The MS-ACS is the best algorithm concerning the Ω metric. **Tables 6** and **7** show the data regarding the results of MS-*MM*AS reported in **Tables 4** and **5**. The results of the other ACO variants can be seen in [5].

Tables 8 and **9** present the average processing time (in seconds) spent by each heuristic. Instances are grouped by the number of vertices. From these tables, it can be conclude that the MS-*MM*AS was the ant algorithm that demanded more

n	AS	ACS	MS-ACS	MS-*MM*AS
10	0.05	0.06	0.12	0.15
20	0.10	0.13	0.26	0.46
30	0.20	0.30	1.92	1.37
40	0.34	0.48	2.29	2.29
50	0.41	2.20	5.77	5.92
100	6.68	28.85	32.69	50.75
200	31.81	270.94	409.72	640.89
500	41.72	3477.13	3545.20	9940.87

Table 8.
Average time spent by the ant algorithms for the set of asymmetric instances.

n	AS	ACS	MS-ACS	MS-*MM*AS
10	0.05	0.06	0.12	0.17
20	0.10	0.13	0.27	0.51
30	0.18	0.24	0.49	0.89
40	0.28	0.38	0.73	2.78
50	0.40	0.56	5.41	4.02
100	8.13	13.51	29.17	34.93
200	22.94	51.92	112.58	97.43
500	40.53	75.72	127.83	309.46

Table 9.
Average time spent by the ant algorithms for the set of symmetric instances.

processing time. **Tables 6** and 7 (1) present detailed results concerning the average time required by the MS-*MM*AS. Data regarding the time consumption of the other ACO variants can be seen in [5].

5. Discussion

The purpose of this study was to adapt *MM*AS to the QTSP-PIC and compare its performance with the ACO variants proposed in [5]. As expected, MS-*MM*AS proved to be competitive regarding the other ACO variants proposed to solve QTSP-PIC. Similarities and differences that were observed in the results are discussed in section 5.1. The limitations of the study are discussed in Section 5.2.

5.1 Comparison between ACO algorithms

The ACO algorithms proposed by Silva et al. [5] showed to be a viable method for solving QTSP-PIC. Yet, the performance of AS and ACS algorithms, when compared to MS-*MM*AS, was rather poor for the benchmark set studied. MS-*MM*AS improved the results achieved by AS in 134 instances. Compared to ACS, MS-*MM*AS performed better in 136 instances. The results achieved by MS-*MM*AS improved those produced by MS-ACS in 93 instances. It is interesting to note that the MS-ACS algorithm performed slightly better on the large instances than the MS-*MM*AS. The Friedman test and Nemenyi post-hoc ranked these two ACO algorithms with the same scale for the most instance groups, which means that difference between the results achieved by the MS-ACS and MS-*MM*AS was significantly small.

These observations are also supported by the variability results of each ACO algorithm. Metric ν showed that MS-*MM*AS was the algorithm that achieved the best know solutions of the benchmark set in most cases. Metric Φ showed that the MS-*MM*AS performed slightly better overall on the benchmark set than the MS-ACS. A possible explanation for this is that the MS-ACS variation might converge to a local minimum faster than the MS-*MM*AS.

Results presented in this study showed that the MS-*MM*AS algorithm is better suited than the other three ACO variants proposed in [5] to solve QTSP-PIC. This suggests a positive impact of the implementation design proposed in this study and a contribution to the *MAX-MIN* Ant System state of the art.

5.2 Limitations

Due to limited time, parallel computing techniques could not be tested to improve the performance of MS-*MM*AS. A previous study done by Skinderowicz [10] investigated the potential effectiveness of a GPU-based parallel *MAX-MIN* Ant System in solving the TSP. In this study, the most promising *MM*AS variant was able to generate over 1 million candidate solutions per second when solving a large instance of the TSP benchmark set. Other techniques can improve the MS-*MM*AS design and could not be tested due to the lack of time:

- A rank-based pheromone updating rule [25];
- The application of the pseudo-random action choice rule proposed in [21];
- Hybridization with other meta-heuristics [26–28].

6. Conclusions

This work dealt with a recently proposed variant of the Traveling Salesman Problem named The Quota Traveling Salesman Problem with Passengers, Incomplete Ride, and Collection Time. In this problem, the salesman uses a flexible ride-sharing system to minimize travel costs while visiting some vertices to satisfy a pre-established quota. He must respect the budget limitations and the maximum travel time of every passenger. Each passenger can be transported directly to the desired destination or an alternate destination. The alternative destination idea suggests that when sharing a ride, pro-environmental or money-saving concerns can induce persons to agree to fulfill their needs at a similar destination. Operational constraints regarding vehicle capacity and travel time were also considered.

The Multi-Strategy *MAX-MIN* Ant System, a variant from the Ant Colony Optimization (ACO) family of algorithms, was presented. This algorithm uses the MS concept improved with roulette wheel selection and memory-based principles to avoid redundant executions of the local search algorithm. The results of MS-*MM*AS were compared with those produced by the ACO algorithms presented in [5]. To support MS-*MM*AS, the ride-matching heuristic and the local search heuristic based on multiple neighborhood operators proposed by [5] were reused.

The computational experiments reported in this study comprised one hundred forty-four instances. The experimental results show that the proposed ant algorithm variant could update the best-known solutions for this benchmark set according to the statistical results. The comparison results with three other ACO variants proposed in [5] showed that MS-*MM*AS improved the best results of MS-ACS for ninety-three instances, and a significant superiority of MS-*MM*AS over AS and ACS.

The presented work may be extended in multiple directions. First, it would be interesting to investigate if the application of the pseudo-random action choice rule [20] could improve the MS-*MM*AS results. Another further promising idea is the use of pheromone update rule based on ants ranking [25]. Extension of the MS-*MM*AS implementation design with parallel computing techniques [10] and hybridization with other meta-heuristics [26–28] is other interesting opportunity for the future research.

Abbreviations

MaaS	Mobility as a Service
QTSP-PIC	Quota Traveling Salesman Problem with Passengers, Incomplete Ride and Collection Time
ACO	Ant Colony Optimization
RMH	Ride-Matching Heuristic
MnLS	Multi-neighborhood Local Search
MS-ACS	Multi-Strategy Ant Colony System
*MM*AS	*MAX-MIN* Ant System
MS-*MM*AS	Multi-Strategy *MAX-MIN* Ant System
MS	Multi-Strategy
TSP	Traveling Salesman Problem
QTSP	Quota Traveling Salesman Problem
AS	Ant System
ACS	Ant Colony System

Author details

Bruno C.H. Silva[1,2*], Islame F.C. Fernandes[2], Marco C. Goldbarg[2] and Elizabeth F.G. Goldbarg[2]

1 Federal University of Ceará No. 1, Fortaleza, Brazil

2 Federal University of Rio Grande do Norte No. 2, Natal, Brazil

*Address all correspondence to: bruno@crateus.ufc.br

IntechOpen

References

[1] Mobile Tech Spurs Financial Inclusion in Developing Nations. shorturl.at/bjyOZ [Accessed in: 2020-08-08]

[2] Hensher D., Ho C., Mulley C., Nelson J., Smith G., Wong Y. (2020). Understanding Mobility as a Service (MaaS): Past, Present and Future. h ttps://doi.org/10.1016/C2019-0-00508-0 [Accessed in: 2020-08-08]

[3] Sin C. Ho, W.Y. Szeto, Yong-Hong Kuo, Janny M.Y. Leung, Matthew Petering, Terence W.H. Tou, A survey of dial-a-ride problems: Literature review and recent developments, Transportation Research Part B: Methodological, Volume 111, 2018, Pages 395–421, ISSN 0191–2615.https://doi.org/10.1016/j.trb.2018.02.001 [Accessed in: 2020-08-02]

[4] de Lira, V. M., Perego, R., Renso, C., Rinzivillo, S., Times, V. C. (2018). Boosting ride sharing with alternative destinations. IEEE Transactions on Intelligent Transportation Systems, 19 (7), 2290–2300. https://doi.org/10.1109/TITS.2018.2836395 [Accessed in: 2020-08-02]

[5] Bruno C.H. Silva, Islame F.C. Fernandes, Marco C. Goldbarg, Elizabeth F.G. Goldbarg. Quota travelling salesman problem with passengers, incomplete ride and collection time optimization by ant-based algorithms. Computers & Operations Research, Volume 120, 2020, 104950, ISSN 0305–0548. https://doi.org/10.1016/j.cor.2020.104950. [Accessed in: 2020-08-02]

[6] M. Dorigo, M. Birattari and T. Stutzle, "Ant colony optimization," in IEEE Computational Intelligence Magazine, vol. 1, no. 4, pp. 28–39, Nov. 2006, https://doi.org/10.1109/MCI.2006.329691. [Accessed in: 2020-08-04]

[7] J. Pedro Schmitt, F. Baldo and R. Stubs Parpinelli. A MAX-MIN Ant System with Short-Term Memory Applied to the Dynamic and Asymmetric Traveling Salesman Problem. 2018 7th Brazilian Conference on Intelligent Systems (BRACIS), Sao Paulo, 2018, pp. 1–6, https://doi.org/10.1109/BRACIS.2018.00009. [Accessed in: 2020-08-04]

[8] J. Pedro Schmitt, R. Stubs Parpinelli and F. Baldo. (2019). Analysis of Max-Min Ant System with Local Search Applied to the Asymmetric and Dynamic Travelling Salesman Problem with Moving Vehicle. In International Symposium on Experimental Algorithms (pp. 202–218). Springer, Cham.https://doi.org/10.1007/978-3-030-34029-2$_$14[Accessed in: 2020-08-04]

[9] Yang, K., You, X., Liu, S. et al. A novel ant colony optimization based on game for traveling salesman problem. Appl Intell (2020). https://doi.org/10.1007/s10489-020-01799-w [Accessed in: 2020-08-04]

[10] R. Skinderowicz. Implementing a GPU-based parallel MAX–MIN Ant System, Future Generation Computer Systems,Volume 106, 2020, Pages 277–295, ISSN 0167-739X. https://doi.org/10.1016/j.future.2020.01.011 [Accessed in: 2020-08-04]

[11] S.L. Martins, P.M. Pardalos, M.G.C. Resende, and C.C. Ribeiro. Greedy randomized adaptive search procedures for the steiner problem in graphs. In P. M. Pardalos, S. Rajasekaran, and J. Rolim, editors, Randomization methods in algorithmic design, volume 43 of DIMACS Series on Discrete Mathematics and Theoretical Computer Science, pages 133–145. American Mathematical Society, 1999. https://bookstore.ams.org/dimacs-43. [Accessed in: 2020-08-04]

[12] K. Ilavarasi and K. S. Joseph. Variants of travelling salesman problem: A survey. International Conference on Information Communication and Embedded Systems (ICICES2014), Chennai, 2014, pp. 1–7. https://doi.org/10.1109/ICICES.2014.7033850. [Accessed in: 2020-08-04]

[13] Awerbuch, B., Azar, Y., Blum, A., Vempala, S. New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen. SIAM Journal on computing, 28(1), 254–262, 1998. https://epubs.siam.org/doi/abs/10.1137/S009753979528826X. [Accessed in: 2020-08-04]

[14] Niels Agatz, Alan Erera, Martin Savelsbergh, Xing Wang. Optimization for dynamic ride-sharing: A review, European Journal of Operational Research, Volume 223, Issue 2, 2012, Pages 295–303. https://doi.org/10.1016/j.ejor.2012.05.028 [Accessed in: 2020-08-04]

[15] Schonberger, J., Kopfer, H., Mattfeld, D. C. A combined approach to solve the pickup and delivery selection problem. In Operations Research Proceedings 2003 (pp. 150–155). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-55537-4$_$24 [Accessed in: 2020-08-04]

[16] Deneubourg, J. L., Aron, S., Goss, S., Pasteels, J. M. The self-organizing exploratory pattern of the argentine ant. Journal of insect behavior, 3(2), 159–168, 1990. https://doi.org/10.1007/BF01417909 [Accessed in: 2020-08-04]

[17] Hart, Adam; Jackson, Duncan E. U-turns on ant pheromone trails. Current Biology, v. 16, n. 2, p. R42-R43, 2006. https://doi.org/10.1016/j.cub.2006.01.015 [Accessed in: 2020-08-04]

[18] Marco Dorigo, Eric Bonabeau, Guy Theraulaz, Ant algorithms and stigmergy, Future Generation Computer Systems, Volume 16, Issue 8, 2000, Pages 851–871, ISSN 0167-739X. https://doi.org/10.1016/S0167-739X(00)00042-X [Accessed in: 2020-08-04]

[19] Skinderowicz R. Ant Colony System with Selective Pheromone Memory for TSP. Lecture Notes in Computer Science, vol 7654. Springer, Berlin, Heidelberg. 2012. https://doi.org/10.1007/978-3-642-34707-8_49 [Accessed in: 2020-08-04]

[20] M. Dorigo and L. M. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. in IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 53–66, April 1997. https://doi.org/doi: 10.1109/4235.585892 [Accessed in: 2020-08-04]

[21] Thomas Stutzle, Holger H. Hoos. MAX–MIN Ant System. Future Generation Computer Systems. Volume 16, Issue 8, 2000, Pages 889–914, ISSN 0167-739X. https://doi.org/10.1016/S0167-739X(00)00043-1 [Accessed in: 2020-08-04]

[22] Manuel Lopez-Ibanez, Jeremie Dubois-Lacoste, Leslie Perez Caceres, Mauro Birattari, Thomas Stutzle. The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives, Volume 3, 2016, Pages 43–58, ISSN 2214–7160. https://doi.org/10.1016/j.orp.2016.09.002 [Accessed in: 2020-08-04]

[23] O'Gorman, Thomas. A comparison of the F-test, Friedman's test, and several aligned rank tests for the analysis of randomized complete blocks. Journal of Agriculture Biology and Environmental Statistics. 6. 367–378, 2001. https://doi.org/10.1198/108571101317096578 [Accessed in: 2020-08-04]

[24] Alan C. Elliott, Linda S. Hynan. A SAS macro implementation of a multiple comparison post hoc test for a Kruskal–Wallis analysis, Computer

Methods and Programs in Biomedicine, Volume 102, Issue 1, 2011, Pages 75–80, ISSN 0169–2607. https://doi.org/10.1016/j.cmpb.2010.11.002 [Accessed in: 2020-08-04]

[25] Q. Song, Q. Zhao, S. Wang, Q. Liu and X. Chen. Dynamic Path Planning for Unmanned Vehicles Based on Fuzzy Logic and Improved Ant Colony Optimization. IEEE Access, vol. 8, pp. 62107–62115, 2020. https://doi.org/10.1109/ACCESS.2020.2984695 [Accessed in: 2020-08-10]

[26] M. Xia. An ant colony algorithm Hybridized with Iterated Local Search for the QAP. 2009 Asia-Pacific Conference on Computational Intelligence and Industrial Applications (PACIIA), Wuhan, pp. 80–83, 2009. https://doi.org/10.1109/PACIIA.2009.5406542 [Accessed in: 2020-08-27]

[27] Li, X., Tian, P., Leung, S. An ant colony optimization metaheuristic hybridized with tabu search for open vehicle routing problems. J Oper Res Soc 60, 1012–1025 (2009). https://doi.org/10.1057/palgrave.jors.2602644 [Accessed in: 2020-08-27]

[28] P.S. Shelokar, Patrick Siarry, V.K. Jayaraman, B.D. Kulkarni. Particle swarm and ant colony algorithms hybridized for improved continuous optimization. Applied Mathematics and Computation, Volume 188, Issue 1, P. 129–142, 2007. https://doi.org/10.1016/j.amc.2006.09.098 [Accessed in: 2020-08-27]