Computers and Operations Research 120 (2020) 104950

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

Technical Note

Quota travelling salesman problem with passengers, incomplete ride and collection time optimization by ant-based algorithms

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

Universidade Federal do Rio Grande do Norte, Campus Lagoa Nova, RN, Brazil

ARTICLE INFO

Article history:
Received 28 June 2019
Revised 23 January 2020
Accepted 28 March 2020
Available online 2 April 2020

Keywords:
Travelling salesman
Integer programming
Transportation
Meta-heuristics
Shared mobility

ABSTRACT

The Quota Travelling Salesman Problem with Passengers, Incomplete Ride, and Collection Time is a new version of the Quota Travelling Salesman Problem. In this problem, the salesman uses a flexible ridesharing system to minimize travel costs while visiting some vertices to satisfy a pre-established quota. We consider operational constraints regarding vehicle capacity, travel time, passenger limitations, and penalties for rides that do not meet passenger requirements. We present a mathematical formulation and heuristics based on Ant Colony Optimization.

1. Introduction

Collaborative consumption is one of the major paradigms influencing the modern commercial world. According to Heikkila (2014), at least five dynamics contribute to the consolidation of collaborative consumption: sustainability, social changes, economic recession, technological enhancements, and opportunities for new business models. In the transportation segment, this paradigm has made the mobility-on-demand (MoD) systems, one of the most powerful actors in the transformation of urban mobility. Such a transportation model encourages shared mobility rather than traditional transportation services that would be underutilized.

Much of the recent literature for mobility as service consider: planning optimal itineraries to a fixed-size vehicle fleet that satisfies as many travel requests as possible (Ho et al., 2018; Riedler and Raidl, 2018), management of MoD systems based on autonomous vehicles (Fagnant and Kockelman, 2018; Farhan and Chen, 2018; Levin et al., 2017), servicing a given set of travel requests with a single trip (Alonso-Mora et al., 2017), operational considerations to maximize the accomplishment of online and dynamic travel requests (Chen et al., 2018; Dong et al., 2018; Hou et al., 2018; Simonin and O'Sullivan, 2014; Zhang et al., 2018), urban mobility in specific cities (Lokhandwala and Cai, 2018; Masson et al., 2017; Wang et al., 2018), and new flexible ridesharing problem formulations (Armant and Brown, 2020; Stiglic et al., 2018; Zhao et al., 2018). Those works aim at matching drivers and persons demanding rides or maximizing the number of fulfilled requests. Moreover, they focus on drivers dedicated to transport persons or goods. However, there are cases in which professionals that provide services in different locations may offer their vehicles for ridesharing. In these cases, they can adapt their routes to accomplish both tasks.

The problem investigated in this study, called Quota Travelling Salesman Problem with Passengers, Incomplete Ride and Collection Time (QTSP-PIC), models applications in which a professional that provides services in different locations offer his or her vehicle to a mobility system. It also deals with the flexibility of the drop-off point, an important issue to address when trying to reduce total traffic. As pointed out by Zhao et al. (2018), in some cases, the pickup or delivery locations of passengers may be spatially nearby but topologically inaccessible or even temporally unreachable for vehicles. It can cause long-time waiting and impact total traffic. The alternate destination idea suggests that when there is the possibility of sharing a ride, pro-environmental or money-saving concerns can induce persons to agree to fulfill their needs at a similar destination (de Lira et al., 2018).

The QTSP-PIC requires that the driver knows user requests in advance. It is a realistic assumption, since a significant part of the transportation demand of a user may be known in advance. It allows matching preferences of passengers and drivers accordingly. It also helps to reduce undesired effects that may result from unallocated periods, i.e., the period the vehicle is available to be assigned to someone and travels with no passenger. Fiedler et al. (2017) indicated that unallocated traffic in the MoD system would add more

* Corresponding author.
E-mail address: bruno@crateus.ufc.br (B.C.H. Silva).

https://doi.org/10.1016/j.cor.2020.104950
0305-0548/

than 30% to total traffic. The QTSP-PIC contains elements of the Dial-a-Ride Problem (DARP) with ridesharing and flexible locations.

The DARP is a well-known optimization model for MoD systems management. The concept of the DARP came from practical cases in which customers requested transportation services by telephone. The objective is to program routes to transport users who specify pickup and drop-off points (Cordeau and Laporte, 2007). It is a generalization of the Pickup and Delivery Problem (PDP) in which the transportation concerns people, rather than products (Savelsbergh and Sol, 1995). Ho et al. (2018) present an overview of DARP generalizations.

Ridesharing (RP) is another problem related to MoD systems management. It consists in finding the best arrangement of people to share the costs of a single trip in a vehicle, regardless of a prior agreement or a history of cooperation (Nourinejad and Roorda, 2014). Simonin and O'Sullivan (2014) present an overview of the RP, its variants and resolution methods. It is noteworthy that a common factor in the traditional versions of DARP and RP (Ho et al., 2018; Simonin and O'Sullivan, 2014) is to maximize the satisfaction of the person who demands transportation. The problem introduced in this study addresses the viewpoint of the driver.

The QTSP-PIC is a variant of the Quota Travelling Salesman Problem (QTSP)(Awerbuch et al., 1998) contextualized within the shared mobility segment. Let $G = (V, E)$ be a complete graph, where V is the set of vertices and E the set of arcs. A non-negative quota is associated with each vertex $v_i \in V$ and a weight c_{ij} is associated with each arc $(i, j) \in E$. The QTSP involves finding a Hamiltonian cycle on a subset V' of V, which minimizes travel costs and satisfies a collected quota constraint. In the QTSP-PIC, the salesman is the driver of a vehicle and can reduce travel costs by sharing expenses with passengers. There are constraints regarding budget limitation and maximum travel duration related to each person demanding a ride. Each passenger can be transported to the desired destination or an alternate destination. Since rides are prearranged, the driver is not obliged to meet all requests.

From the viewpoint that profit is the excess of returns over expenditure, the model does not allow the driver to profit with rides regarding the route costs. There are countries where the legislation does not allow profit from rides. The objective function of the model proposed in this study takes this assumption. However, it can be easily adapted to situations in which profit is allowed.

Practical applications of the QTSP-PIC model occur in situations where a person, traveling in a private vehicle, uses a ridesharing service to give rides and reduce transportation expenses while visiting some localities to satisfy a personal quota. For instance, consider an independent local courier who must achieve a minimum quota of work. The QTSP-PIC may be applied to assist the courier in scheduling deliveries, minimizing transportation costs as a result of rides. Cases related to sales and tourism are also pertinent ones. The salesman must choose which localities to visit to reach a minimum sales quota and in what order to visit them to take advantage of the best travel requests. In the second case, the driver can be a tourist who must choose the best tourist attractions to visit and uses a ridesharing system to share travel costs. In both cases, the driver may negotiate discounts with passengers who agree to go to alternate destinations.

We present a mathematical programming model for the QTSP-PIC and submit it to an optimization tool. Since it is a new problem, we created 144 instances with the number of vertices ranging from 10 to 500, and the number of potential passengers can reach more than 10000.

We adapted some meta-heuristics from the Ant Colony Optimization (ACO) family of algorithms. We have chosen to implement an ACO algorithm since others from this class were successful when dealing with several routing problems that have similarities with the QTSP-PIC (Herbawi and Weber, 2011; Huang et al., 2018; Liaw et al., 2017; Tripathy et al., 2018; Zufferey et al., 2015). We implemented and compared the results of naive heuristics with those produced by the ACO algorithms. We also present a ride-matching heuristic to support the meta-heuristics and a local search based on multiple neighborhood operators.

Section 2 presents the problem and its formulation. Section 3 presents the solution representation of the QTSP-PIC and three heuristics used in our experiments: the ride-matching heuristic, the naive heuristic, and the local search. Section 4 presents the ACO algorithms adapted to the QTSP-PIC. We discuss our experiments in Section 5. Conclusions and future research directions are outlined in Section 6.

2. Problem definition

The Travelling Salesman Problem (TSP) can be modeled 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 defined such that c_{ij} is the cost associated with arc (i, j). The goal is to determine the shortest Hamiltonian cycle in G. The TSP is NP-hard (Karp, 1975) and also one of the most investigated problems in Combinatorial Optimization.

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. This problem was introduced by Awerbuch and colleagues (Awerbuch et al., 1998), who also presented other prize-collecting oriented routing problems.

The QTSP-PIC is a variant of the QTSP 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, which consists 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 the travel requests. This model combines elements of ridesharing systems (Agatz et al., 2012) with alternative destinations (de Lira et al., 2018), and the selective pickup and delivery problem (Schönberger et al., 2003).

The salesman starts at city $s = 1$ and visits each node of a cycle $\Psi = (N', A')$, $N' \subseteq N$, $A' \subseteq A$, at most once. A bonus, q_i, is associated with each vertex $i \in N$. The bonus collection at vertex i requires g_i time units. It corresponds, for example, to the time spent by the salesman to accomplish a task (provide a service) in that vertex (city). The salesman can choose between accomplish or not the task (and spend or not time in that vertex). It corresponds to collect or not the bonus of a visited vertex. If the salesman chooses not to collect the bonus, the saved time can be enough to meet the requirements of new passengers. The salesman has to collect, at least, K units of bonus.

The cost and time required to traverse arc $(i, j) \in A$ are c_{ij} and t_{ij}, respectively. The vehicle has capacity for, at most, R passengers, besides the salesman. The salesman shares the cost to traverse arc $(i, j) \in A$ with the passengers who also traverse (i, j). The cost is divided equally among the occupants of the vehicle. Let L be the set of travel requests, each request associated to a person. We denote by $L_i \subseteq L$ the subset of travel requests for which $i \in N$ is the origin city. Let l be a person who demands a ride. We denote by $org(l)$ and $dst(l)$, the pickup and drop off points required by l. The maximum fee l agrees to pay for a trip is denoted by w_l, and the maximum duration of a journey is denoted by b_l. The salesman is allowed to take the passenger to a place different from

the demanded destination, i.e. an alternative destination. In this case, there is a penalty to be paid. The penalty to take l to city j, $j \neq dst(l)$, is denoted by h_{lj}.

The QTSP-PIC consists in finding a cycle, Ψ, such that the cost of the route and the penalties are minimized, and the quota constraint is satisfied.

The QTSP is an NP-hard problem (Awerbuch et al., 1998) and a particular case of the QTSP-PIC when 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 is also in the NP-hard class.

An integer non-linear programming model for the QTSP-PIC is presented in formulation (1)–(22). The parameters and variables are listed as follows.

Parameters	
n:	number of vertices
c_{ij}:	cost of arc (i, j)
t_{ij}:	time required to traverse arc (i, j)
q_i:	bonus at vertex i
g_i:	time required to collect the bonus at vertex i
$org(l)$:	origin city of the l-th ride request
$dst(l)$:	destination city of the l-th ride request
b_l:	maximum duration of a journey for the l-th ride request
w_l:	budget for the l-th ride request
h_{lj}:	penalty for taking passenger l to city j
K:	minimum amount to be collected by the travelling salesman
R:	vehicle capacity

Variables	
x_{ij}:	indicates whether the salesman traverses arc (i, j), $x_{ij} = 1$, or not, $x_{ij} = 0$
p_i:	indicates whether the salesman collects the bonus of vertex i, $p_i = 1$, or not, $p_i = 0$
v^l_{ij}:	indicates whether passenger l traverses arc (i, j), $v^l_{ij} = 1$, or not, $v^l_{ij} = 0$
f_{lj}:	indicates whether passenger l leaves the vehicle at city j, $f_{lj} = 1$, or not, $f_{lj} = 0$
d_l:	indicates whether passenger l is picked-up, $d_l = 1$, or not, $d_l = 0$
u_i:	indicates the position of vertex i in the tour. We recall that $u_1 = 1$.

$$\min \sum_{(i,j)\in A} \frac{x_{ij}c_{ij}}{1+\sum_{l\in L} v^l_{ij}} + \sum_{l\in L}\sum_{j=1}^{n} h_{lj}f_{lj} \tag{1}$$

Subject to:

$$\sum_{i\in N\setminus\{j\}} x_{ij} \leq 1 \quad \forall j \in N\setminus\{s\} \tag{2}$$

$$\sum_{i\in N\setminus\{j\}} x_{ji} \leq 1 \quad \forall j \in N\setminus\{s\} \tag{3}$$

$$\sum_{i\in N\setminus\{s\}} x_{si} = 1 \tag{4}$$

$$\sum_{i\in N\setminus\{s\}} x_{is} = 1 \tag{5}$$

$$\sum_{i\in N\setminus\{j\}} x_{ij} - \sum_{i\in N\setminus\{j\}} x_{ji} = 0 \quad \forall j \in N\setminus\{s\} \tag{6}$$

$$1 \leq u_i \leq n-1 \quad \forall i \in N \tag{7}$$

$$u_i - u_j + (n-1)x_{ij} \leq n-2 \quad 2 \leq i, j \leq n, i \neq j \tag{8}$$

$$\sum_{(i,j)\in A} p_i q_i x_{ij} \geq K \tag{9}$$

$$\sum_{(i,j)\in A} v^l_{ij}t_{ij} + \sum_{(i,j)\in A} p_i g_i v^l_{ij} \leq b_l \quad \forall l \in L \tag{10}$$

$$\sum_{l\in L} v^l_{ij} \leq R x_{ij} \quad \forall (i, j) \in A \tag{11}$$

$$\sum_{j\in N} f_{lj} \leq 1 \quad \forall l \in L \tag{12}$$

$$\sum_{(i,j)\in A} \frac{v^l_{ij}c_{ij}}{1+\sum_{e\in L} v^e_{ij}} \leq w_l \quad \forall l \in L \tag{13}$$

$$f_{li} = 0 \quad \forall l \in L,\ i = org(l) \tag{14}$$

$$v^l_{ij} \leq d_l \quad \forall l \in L,\ (i, j) \in A \tag{15}$$

$$\sum_{i\in N\setminus\{j\}} v^l_{ij} - \sum_{i\in N\setminus\{j\}} v^l_{ji} = f_{lj} \quad \forall l \in L,\ j \in N\setminus\{org(l)\} \tag{16}$$

$$\sum_{j\in N} f_{lj} = d_l \quad \forall l \in L \tag{17}$$

$$\sum_{j\in N} v^l_{org(l)j} = d_l \quad \forall l \in L \tag{18}$$

$$f_{li} + \sum_{j\in N} v^l_{ij} \leq 1 \quad \forall l \in L, \forall i \in N \tag{19}$$

$$\sum_{i\in N} v^l_{iorg(l)} = 0 \quad \forall l \in L \tag{20}$$

$$\sum_{j\in N\setminus\{s\}} v^l_{js} = f_{ls} \quad \forall l \in L \tag{21}$$

$$\sum_{j\in N} v^l_{dst(l)j} = 0 \quad \forall l \in L \tag{22}$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A \tag{23}$$

$$p_i \in \{0, 1\} \quad \forall i \in N \tag{24}$$

$$v^l_{ij} \in \{0, 1\} \quad \forall l \in L, (i, j) \in A \tag{25}$$

$$f_{lj} \in \{0, 1\} \quad \forall l \in L, \forall j \in N \tag{26}$$

$$d_l \in \{0, 1\} \quad \forall l \in L \tag{27}$$

The objective function, expression (1), aims at minimizing two parcels: the cost of the route, from the viewpoint of the salesman, and the penalties regarding passengers taken to alternative destinations. The salesman shares the cost of the tour with the passengers. Constraints (2) and (3) guarantee that the salesman visits each city of a subset of vertices $N' \subseteq N$ exactly once. Constraints (4) and (5) ensure that the tour starts and ends at vertex $s \in N$. Constraint (6) guarantees the continuity of the tour, implying that if the salesman arrives at a vertex, then he must depart from it. Constraints (7) and (8) forbid sub-tours (Miller et al., 1960), where level variables u_i denote the visiting order of the vertices in the salesman tour. Constraint (8) guarantees that if vertex j is visited immediately after vertex i, the level of i is one unit smaller than the level of j. When $x_{ij} = 0$, the inequality of constraint (8) is trivially satisfied. Constraint (9) guarantees that the salesman collects the minimum quota K. Constraint (10) guarantees that the maximum duration of a journey required by each passenger is satisfied. Constraint (11) guarantees that the capacity of the vehicle is not exceeded. Constraint (12) ensures that no passenger leaves the vehicle in more than one locality. Constraint (13) ensures that the budget limits of all passengers are not exceeded. Constraint (14) ensures that the pickup and drop-off points of each passenger are different. Constraint (15) ensures that passenger $l \in L$ traverses arc (i, j) only if l was picked-up. Constraint (16) guarantees that if passenger l arrives at $j \in N \backslash \{org(l)\}$ and does not continue to another city, l leaves the vehicle at j. This constraint also ensures that if l arrives at j and goes to another city, l does not leave the vehicle at j. Constraints (17) ensures that passenger l leaves the vehicle at city j if and only if l was picked up. Constraint (18) ensures that a passenger picked up by the salesman at vertex i must traverse at least one arc (i, j). If passenger l leaves the vehicle at vertex i, constraint (19) ensures that l does not continue in the vehicle after i. Constraint (20) ensures that passenger l, picked-up at $j = org(l)$, does not traverse any arc (i, j). Constraint (21) ensures that if passenger l arrives at vertex s, l leaves the vehicle at s, since s is the starting and final point for the salesman. Constraint (22) ensures that passenger l does not traverse any arc $(dst(l), j)$, i.e., l leaves the vehicle at vertex $dst(l)$ if l is in the vehicle in some arc $(i, dst(l))$. Finally, constraints (23)-(27) define the scope of the binary variables.

3. Auxiliary heuristics

In this section, we present the solution representation and introduce three methods used in our experiments. Section 3.1 presents the representation of the solution of a QTSP-PIC instance. Section 3.2 presents the *Ride-Matching Heuristic* (RMH) that, given a cycle in G, chooses which persons are picked up. Section 3.3 presents the local search used to refine the solutions found by the methods proposed. Finally, Section 3.4 presents a naive heuristic that provides initial results for the instances of the computational experiments. To improve readability, Table 1 summarizes the parameters and symbols used in the algorithms described in this section.

3.1. Solution representation

A solution for the QTSP-PIC is defined 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 $H'[l] = 0$.

Table 1
Summary of symbols used in the auxiliary heuristics.

Symbol	Description
S	Input solution
S'	Output solution
Ψ	Cycle in S and S'
N'	List of vertices in Ψ
L	Set of ride requests
L'	Set of ride requests assigned to S
R	Vehicle capacity
b	List of the maximum duration of a journey for each person
w	List of the maximum amount each person agrees to pay for a ride
h	List of penalties for drop-offs in alternate destinations
E	Set of ride requests sorted according to w
α	Drop-off vertex computed for ride request $j \in E$
θ	Index of a neighborhood structure
K	Minimum quota to be collected
$INST$	Name of the input instance
ξ	Index of the method applied to build a route
κ	Index of the method applied to assign passengers to a route

The cost of solution S, denoted by $S.cost$, is calculated with Eq. (28).

$$S.cost = \sum_{i,j \in N'} \frac{c_{ij}}{1 + \sum_{l \in L'} v_{ij}^{l}} + \sum_{l \in L'} h_{lH'_l} \tag{28}$$

3.2. Ride-Matching heuristic

Let S be a solution and Ψ the cycle in S. Ψ consists of a list of vertices, denoted by N', and a list of arcs, denoted by A'. The RMH is a greedy method that assigns rides to Ψ. Algorithm 1 presents

Algorithm 1: RMH (S, L, R, b, w, h).

```
1.  S' ← copy(S)
2.  E ← L sorted in non-increasing order of w_l
3.  For each j ∈ E
4.      if org(L[E[j]]) ∈ S.N' ▷ Checks if the pick-up vertex of j is visited
5.
α ← compute_drop(S.N', h, org(L[E[j]]), b_{L[E[j]]}, w_{L[E[j]]}, R)
6.          Assign passenger L[E[j]] to S with drop-off vertex α
7.          S.cost ← evaluate(S)
8.          If S.cost < S'.cost
9.              S' ← S
10.         Else
11.             Remove L[E[j]] from S          ▷ Cancel the ride-matching of j
12. Return S'
```

the pseudo-code of the RMH. There are five input parameters: S, an initial solution with no ride requests assigned to it, i.e. $L'[i] = 0$, $i = 1, \ldots, |L|$; L, the set of ride requests; R, the vehicle capacity; b, the list of the maximum duration of a journey for each person; w, the list of the maximum amount each person agrees to pay for a ride; and h, the list of penalties to take each person to a destination different from the required one. Since, initially, there is no ride request assigned to Ψ, the cost of S is $S.cost = \sum_{(i,j) \in A'} c_{ij}$. The output of the RMH is S', a solution with ride requests assigned to its cycle.

The RMH sorts list L in non increasing order of w_l, the budget of the l-th person. The sorted list is set to E (step 2). The main loop (lines 3–11) checks the possibility of satisfying ride requests. Persons with higher budgets are evaluated before others with lower ones. If the salesman visits the origin city of the $E[j]$-th person, $org(L[E[j]])$, the algorithm computes the drop-off point

to $E[j]$ in the *compute_drop*() procedure and sets it to α (step 5). The *compute_drop*() procedure returns a city from $S.N'$, i.e., a city visited by the salesman. Let v be the city returned by that procedure, v is after $org(L[E[j]])$ in the tour, and the penalty for taking the $E[j]$-th person to v, $h_{L[E[j]]v}$, is minimum concerning the cities in $S.N'$, and the constraints regarding the budget and journey duration for the $E[j]$-th person ($b_{L[E[j]]}$, $w_{L[E[j]]}$) are satisfied. The vehicle capacity, R, is not exceeded. Passenger $L[E[j]]$ is assigned to S with $org(L[E[j]])$ and α as the pickup and drop-off points, respectively (step 6). The cost of solution S is computed by Eq. (28) (step 7). Solution S replaces S', if the cost of the former is lower than the latter (steps 8–9). Otherwise, the algorithm removes the assignment of the $E[j]$-th passenger (setp 11).

3.3. Multi-neighborhood local search

The local search proposed in this study deals with different neighborhood structures. We call it Multi-neighborhood Local Search (MnLS). The neighborhoods deal only with route and bonus collection. The algorithm assigns passengers after obtaining each route. Algorithm 2 presents the MnLS pseudo-code. The input, S,

Algorithm 2: MnLS (S).

```
1.  S' ← copy(S)
2.  θ ← random neighborhood operator
3.  For each i ∈ S'.N' − {v0} do
4.      S'' ← perturbation(S', i, θ)
5.      While S''.quota < K      ▷ Guarantees the minimum quota
6.          S'' ← perturbation(S', i, push)
7.      S'' ← RMH(S'', L)
8.      If S''.cost < S'.cost
9.          S' ← S''
10. Return S'
```

is a QSTP-PIC solution. The output is S', the best solution found by the local search.

After S is copied to S' (step 1), a neighborhood structure is randomly selected and assigned to θ (step 2). There are five neighborhood structures called: *swap, flip, push, pop-vertex* and *pop-quota*.

Let $\Psi = (v_0, \ldots, v_i, \ldots, v_j, \ldots, v_0)$ be the cycle in S. The bonuses are collected in all vertices of Ψ. S satisfies the quota constraint. Every iteration, a vertex from Ψ is processed. Let $v_i \in \Psi$ be the vertex processed in the i-th iteration. The swap neighborhood operator replaces v_i by v_j, $v_j \notin \Psi$, selected at random. The quota associated to vertex v_j is marked as collected.

The flip neighborhood operator reverses a path in Ψ between vertices v_i and v_k, $v_k \in \Psi$, $i < k$. Let $P = (v_i, v_{i+1}, \ldots, v_{k-1}, v_k)$ be a path in Ψ, v_k selected at random. The flip operator reverses P producing $\Psi = (v_0, \ldots, v_k, v_{k-1}, \ldots, v_{i+1}, v_i, \ldots, v_0)$.

The push operator adds an unvisited vertex, chosen at random, to Ψ. The unvisited vertex is inserted after vertex v_i. The pop-vertex operator removes vertex v_i from Ψ. The pop-quota operator deactivates the bonus collection at vertex v_i. Thus, the salesman does not collect the bonus and does not compute the time to collect it at vertex v_i. Then, maybe, the time gained can be enough to include new rides in S.

In the main loop (steps 3–9), the algorithm searches for new solutions that can be generated with neighborhood θ. The flip and push neighborhood operators do not violate the minimum quota constraint K. However, other neighborhood operations may violate it. If the solution violates the minimum quota constraint, the algorithm executes the push operator (step 6) such that the unvisited vertex chosen at random is inserted in the position that results in the lowest increment in the cost of Ψ. So, new vertices are included. The bonus collection is active for those new vertices.

Finally, Algorithm 2 returns the best solution generated (step 10).

3.4. Naive heuristic

Since QTSP-PIC is a new problem, there are no previous results for it. Thus, we implemented a simple heuristic, which we called *Naive* (NH), to provide the first results for the QTSP-PIC. In the computational experiments, we compare those results to the ones produced by the approaches proposed in this study.

Algorithm 3 presents the pseudo-code of the NH heuristic. It

Algorithm 3: NH (*INST*, ξ, κ).

```
1. S ← routing(INST, ξ)
2. S' ← rideMatching(S, κ)
3. return S'
```

consists of two procedures: *routing*() and *rideMatching*(). The input are the QTSP-PIC instance, *INST*, and two parameters ξ, and κ related to procedures *routing*() and *rideMatching*().

The *routing* procedure outputs a cycle that satisfies the quota constraint, i.e., a QTSP solution. Parameter ξ stands for the type of method used inside the procedure: exact or heuristic. If $\xi =$ exact, the instance is submitted to an optimization solver to execute the QTSP mathematical model presented in Awerbuch et al. (1998). Otherwise, the instance is submitted to a Greedy Randomized Adaptive Procedure (GRASP).

The GRASP meta-heuristic has two phases: construction and local search (Feo and Resende, 1989). The first phase constructs a feasible solution. It adds elements, to an initially empty solution, iteratively. The best elements, according to a greedy function, compose a list, namely restrict candidate list (RCL). Every iteration of the construction phase, an element is chosen at random to be added to the solution. In the second phase, the algorithm applies local search to the solution built in the first phase.

The first phase of the GRASP developed for the QTSP-PIC builds a route that starts at s. Every iteration, a city is added to the route up to reaching the minimum quota K. The RCL contains the closest cities to the last added one. Let city v_i be the city added to the route at the i-th iteration of the construction phase. In the $i+1$-th iteration, the algorithm selects city v_{i+1} from the RCL. The size of the RCL depends on the distance of the cities not included in the route from v_i. Cities v_j such that $c_{ij} \leq [c_{ij}^{\min}, c_{ij}^{\min} + \alpha(c_{ij}^{\max} - c_{ij}^{\min})]$ compose the RCL, where $\alpha \in [0, 1]$ is a parameter, $c_{ij}^{\min}$ and $c_{ij}^{\max}$ are the distances between the closest and the farthest cities from v_i, respectively. The second GRASP phase applies 2-opt local search (Croes, 1958) to the route built in the first phase. The stopping criterion is a limit for the number of GRASP iterations.

The *rideMatching* procedure receives the κ parameter, and the solution S created by the *routing* procedure. Similarly to the ξ parameter, κ stands for the type of method used to assign passengers to the route: exact or heuristic. If $\kappa =$ exact, the model presented in (1)-(22) is submitted to an optimization solver with variables x_{ij} fixed according to S, i.e., the route created by the *routing* procedure. Otherwise, the instance is submitted to the RMH heuristic.

4. Ant-based approaches

This section presents the Multi-Strategy Ant Colony System applied to the QTSP-PIC. The approach proposed here is an adaptation of the Ant System (AS), proposed by Dorigo et al. (1996), and the Ant Colony System (ACS), introduced by Dorigo and Gambardella (1997). Sections 4.1 and 4.2 present the adaptations of the AS and the ACS, respectively. Section 4.3 presents the Multi-Strategy Ant Colony System.

4.1. Ant system

The AS algorithm was the first ant-based algorithm. It is the baseline of the Ant Colony Optimization and has the following parameters: number of ants, $m \in \mathcal{Z}_{>0}$, pheromone coefficient, $\alpha \in \mathcal{R}_{>0}$, heuristic coefficient, $\beta \in \mathcal{R}_{>0}$, evaporation factor, $\rho \in [0, 1]$ (Dorigo et al., 1996), and initial pheromone in arc (i, j), τ_{ij}^0. α and β are parameters to weight the influence of the pheromone and heuristic information, respectively, during the route construction process executed by each ant.

Algorithm 4 presents the pseudo-code of the AS for the QTSP-PIC. Ants start at vertex s and add vertices to the route up to reach the minimum quota. Every route built by an ant is submitted to the *RMH* algorithm.

Algorithm 4: AS(m, α, β, ρ, τ_{ij}^0).

1. Initialize pheromone trails
2. For $k \leftarrow 1$ to m
3. $W^k[1] \leftarrow s$; $W^k[2] \leftarrow$ random_city($N \setminus \{s\}$)
4. For $k \leftarrow 1$ to m
5. $W^k \leftarrow$ build_route(α, β)
6. $S^k \leftarrow$ assign_passengers(W^k)
7. Pheromone_update(W^k, ρ, τ_{ij}^0)
8. Update(S^*)
9. Return S^*

As proposed by Dorigo and Gambardella (1997), initially, the arcs have the same amount of pheromone, τ_{ij}^0, computed by Eq. (30), where n is the number of vertices and $Cost(D)$ is the value of the TSP tour built by the Nearest Neighbor heuristic presented by Rosenkrantz et al. (1977). Every ant begins at vertex s and the second vertex added to the route is selected at random with uniform distribution (steps 2 and 3).

Ants build their routes in step 5 using Eq. (29) which computes 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. α and β are coefficients that weight the influence of the pheromone and the heuristic information, respectively. They are user-defined parameters. If $\alpha = 0$, the probability computed by Eq. (29) 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.

$$\Upsilon_{ij}^k(t) = \frac{\left[\tau_{ij}(t)\right]^\alpha \cdot \left[\eta_{ij}\right]^\beta}{\sum_{w \in \Lambda^k}\left[\left[\tau_{iw}(t)\right]^\alpha \cdot \left[\eta_{iw}\right]^\beta\right]}, \qquad j \in \Lambda^k \tag{29}$$

Solution S^k, built by the k-th ant, is computed by assigning passengers to W^k with the *RMH* algorithm (step 6). Eqs. (31)-(32) show the formulas used to update pheromone trails (step 7), where $\tau_{ij}(0)$ denotes the initial pheromone in arc (i, j), ρ is the evaporation coefficient, $Cost(W^k)$ is the cost of the route W of the k-th ant, and $\Delta\tau_{ij}^k$, computed by Eq. (33), is the pheromone deposited on arc (i, j) by the k-th ant. The best solution found so far is computed is step 8. Finally, the algorithm returns S^*.

$$\tau_{ij}^0 = (n \times Cost(D))^{-1} \tag{30}$$

$$\tau_{ij} = (1 - \rho) \times \tau_{ij} + \rho \times \Delta\tau_{ij}, \qquad \rho \in [0, 1] \tag{31}$$

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^k \tag{32}$$

$$\Delta\tau_{ij} = \begin{cases} \frac{1}{Cost(W^k),} & \text{if arc } (i, j) \in W^k \\ 0, & \text{otherwise.} \end{cases} \tag{33}$$

The main idea behind Eqs. (31)–(33) is to build a distributed memory mechanism implemented as pheromone where the information is not stored locally in individual ants, but distributed at the arcs of the graph (Dorigo et al., 1996). Since the cost c_{ij} of each arc $(i, j) \in W$ is divided equally among the salesman and the passengers, the amount of pheromone deposited in arc (i, j) is influenced by the travel requests fulfilled in W^k.

4.2. Ant colony system

The ACS also uses Eq. (31) to compute the probability of an ant to move from vertex i to j. However, there is a greedy bias introduced by the coefficient $q0$, a user-defined parameter. Let ω be a random number from range [0, 1]. If $\omega < q0$, the ant chooses to move to j such that $\eta_{ij} = \frac{1}{c_{ij}}$ is maximum. Otherwise, the transition rule established by Eq. (31) is applied. Algorithm 5 shows the pseudo-code of the ACS proposed in this study. Besides m, α, β, ρ, τ_{ij}^0, and $q0$, the ACS has the *maxIter* parameter, the maximum number of iterations, and the ϕ parameter used to update pheromone locally.

Algorithm 5: ACS(*maxIter*, m, α, β, ρ, ϕ, τ_{ij}^0, $q0$).

1. Initialize pheromone trails
2. For $k = 1$ to m
3. $W^k[2] \leftarrow$ random_city($N \setminus \{s\}$)
4. For $i = 1$ to *maxIter*
5. For $k = 1$ to m
6. $W^k \leftarrow$ build_route(α, β, $q0$)
7. $S^k \leftarrow$ assign_passengers(W^k)
8. Local_pheromone_update(W^k, ϕ, τ_{ij}^0)
9. Update(W^*, S^*)
10. $S^* \leftarrow$ MnLS(S^*)
11. Global_pheromone_update(S^*, ρ)
12. Return S^*

The ACS initializes pheromone trails, as described in Section 4.1 (step 1). Since all ants begin at vertex s, the selection of the second vertex of the route built by each ant is at random with uniform distribution (steps 2 and 3). The k-th ant builds the route in step 6. Then, the algorithm uses the *RMH* heuristic to assign passengers to W^k, completing the solution of the k-th ant (step 7). Algorithm 5 updates pheromone locally and globally. Eq. (34) is used for local update (step 8), where ϕ is a decay factor and τ_{ij}^0 is the initial pheromone in arc (i, j). The best route found so far, W^*, is used for global pheromone intensification. It is updated, as well as the best solution, S^*, in step 9. The MnLS algorithm is applied to the best solution found so far (step 10). After local search, the route associated to S^*, W', is used for the global pheromone intensification, with Eq. (31). Finally, the algorithm returns S^*.

$$\tau_{ij} = (1 - \phi) \times \tau_{ij} + \phi \times \tau_{ij}^0, \qquad \phi \in [0, 1] \tag{34}$$

4.3. Multi-strategy ant colony system

There are different types of decisions to be made concerning the QTSP-PIC: satisfaction of the minimum quota, acceptance of

ride requests, and minimization of travel costs under the viewpoint of the salesman. In this context, the idea of the Multi-Strategy Ant Colony System (MS-ACS) is to use different sources of heuristic information for the ants.

In the traditional ACS, all ants use the same heuristic information: the cost of the arcs. In the MS-ACS, there are four sources of heuristic information, which are listed as follows.

- Cost oriented: the heuristic information is the same used in the ACS, i.e., $\eta_{ij} = \frac{1}{c_{ij}}$.
- Time oriented: the heuristic information is $\eta_{ij} = \frac{1}{t_{ij}}$. This strategy aims at saving travel times.
- Quota oriented: the heuristic information is $\eta_{ij} = \frac{q_j}{c_{ij}}$. This strategy aims at choosing vertices with a "good" rate between the bonus of vertex j and the cost to go to j.
- Passenger oriented: the heuristic information is $\eta_{ij} = \frac{|L_j|}{c_{ij}}$.This strategy aims at maximizing the possibility to satisfy as many as possible ride requests. Then, vertices with the best rates between the number of ride requests and the cost to reach them are more likely to come after the current vertex i.

Every ant decides which strategy to use at random with uniform distribution. We designed different heuristic strategies to cover all aspects of the problem and avoid trapping into local optima.

The MS-ACS follows the same structure of the ACS algorithm. However, to build the route (step 6), the ant randomly selects a source of heuristic information.

5. Experiments and results

In this section, we present the methodology used for the experiments and the results. Section 5.1 presents the methodology, including a description of the QTSP-PIC instances. Section 5.2 presents the parameters used in the heuristic algorithms. Section 5.3 presents the results obtained by the optimization tool and Section 5.4 presents the results.

5.1. Methodology

We executed the experiments 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 mathematical formulation, described in Eqs. (1)–(22), was implemented in Gurobi solver(Gurobi, 2018), version 7.5.

We created a set of 144 instances (symmetric and asymmetric) using the method described in Section 5.1.1. 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 the QTSP-PIC repository.

We tuned the parameters of the heuristic algorithms with the IRACE software (López-Ibáñez et al., 2016). The parameter tuning is detailed in Section 5.2. We submitted the QTSP-PIC instances to the solver to conclude about the difficulty to solve our model and to obtain the optimal results. We set 80,000s as the maximum processing time for the solver. Section 5.3 presents the results of this experiment.

We report the best and average results from 20 independent executions of the heuristic algorithms, and the average processing times in seconds. We tested five versions of the naive heuristic. They differ from one another in the implementation of the routing and ride-matching procedures that can be exact or heuristic. In the exact routing procedure, the input for the solver is objective function (35) and constraints (2)–(9) and (23). The solver searches for an optimal solution for the QTSP while the maximum processing time is not reached. In the exact ride-matching, the input for the solver is formulation (1)–(22) with the x_{ij} variables prefixed by the routing procedure. The maximum processing time allowed for the solver was 80,000s.

$$\min \sum_{(i,j)\in A} x_{ij}c_{ij} \tag{35}$$

There are five versions of the naive heuristic, namely *NHx*. The *NH*1 version combines exact routing and ride-matching. *NH*2 combines exact routing and heuristic ride-matching. *NH*3 consists of heuristic routing and exact ride-matching. *NH*4 combines heuristic routing with ride-matching. Finally, the *NH*5 version is composed of the *NH*4 and the *MnLS* applied to the solution produced by the *NH*4. The *NH*1 and *NH*2 methods were executed just once for each instance. Naive heuristics provides initial results for the benchmarking set. So we test the effectiveness of the proposed ant algorithms compared to the initial results. We compared ant algorithms with each other to establish which is the most viable method to solve our problem and to conclude about the influence of specific heuristics behaviors.

We conduct experiments to report the distance between the best known solutions and the best results provided by each ant algorithm. We also calculate the variability in which the ant algorithms achieved the best known solutions stated in the benchmarking set. With these experiments it is possible to conclude if each algorithm was able to find the best known solution of each instance and with what variability this happens.

We applied the Friedman test (Friedman, 1937) with the Nemenyi post-hoc procedure (Nemenyi, 1963), significance level 0.05, to conclude about significant differences among the results of the ant algorithms proposed in this study. We grouped the instances 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.

5.1.1. Instances

The input for the instance generation algorithm is the number of vertices, n, the capacity of the vehicle, R, and the instance type, symmetric or asymmetric. We used the range for the number of vertices, n, and the number of travel requests adopted by Reinelt (1991), Cordeau and Laporte (2003), Uchoa et al. (2017), since these studies address problems similar to the QTSP-PIC. The range of values for R is in accordance with the capacity of private cars and varies from range [3,5]. We created three classes of instances named A, B and C. The parameters used by this algorithm are defined to build instances with varied difficulty and similar features to correlated problems such as the QTSP(Awerbuch et al., 1998), DARP(Cordeau and Laporte, 2007) and RP(Nourinejad and Roorda, 2014).

The elements of the cost matrices (c_{ij}) of the three instance classes were generated uniformly from the range [100,999]. The elements of the time matrices (t_{ij}) of class A instances were generated uniformly from the range [20,99]. The generation of the elements of the time matrices of classes B and C instances depends on their cost matrices. Let av_c be the average of the elements of the cost matrix of a class B or C instance. If $c_{ij} < av\_c$, then t_{ij} is a value uniformly generated from the range [40,99], otherwise the value comes from the range [20,39]. So, lower cost arcs result in higher traffic times. The idea is to simulate urban traffic congestion scenarios where the driver must balance the routing distance with traffic time to fulfill travel requests.

The number of travel requests in each vertex depends on the order of the graph and the car capacity. If $n \leq 20$, the number of travel requests in each vertex is uniformly generated from [0, $3R$];

Table 2
Ranges of values used to set the parameters in the instance generation algorithm.

Parameter	A	B	C
c_{ij}	[100,999]	[100,999]	[100,999]
t_{ij}	[20,99]	[20,39] or [40,99]	[20,39] or [40,99]
w_l	$[0.15\iota, 0.3\iota]$	$[0.15\iota, 0.3\iota]$	$[0.15\iota, 0.3\iota]$
z_{lj}	[0.9,1.5]	[0.9,1.5]	[0.10,0.30]
q_i	[100,299] or [300,700]	[100,299] or [300,700]	[100,299] or [300,700]
g_i	[1,39] or [40,79]	[1,39] or [40,79]	[1,39] or [40,79]
g_i	[1,39] or [40,79]	[1,39] or [40,79]	[1,39] or [40,79]
ϵ	[0.4,0.6]	[0.4,0.6]	[0.4,0.6]

for $20 < n \leq 100$, the range is $[0, 9R]$; and for $n > 100$ the range is $[0, 27R]$.

The destination of the l-th travel request ($dst(l)$) is uniformly generated from $[1, n]$, $dst(l) \neq org(l)$. The maximum fare the l-th person agrees to pay, w_l, is uniformly generated from $[0.15\iota, 0.3\iota]$, where ι is the length of the minimum spanning tree of G. The weights considered for the minimum spanning tree were the costs, c_{ij}. The maximum duration of a journey for the l-th ride request, b_l, is the length of the shortest path between $org(l)$ and $dst(l)$ in G. The weight considered for each arc (i, j) of G to compute the shortest path was t_{ij}.

The penalty to take the l-th person to destination j, h_{lj}, is 0 if $j = org(l)$ or $j = dst(l)$. Otherwise, $h_{lj} = h0_{lj} \times z_{ij}$, where $h0_{lj}$ is the shortest path between $org(l)$ and j, and z_{lj} is uniformly generated from [0.90,1.50] for classes A and B instances and from [0.10,0.30] for class C instances. The penalties of class C instances are moderate in comparison to the other classes. Moderate penalties tend to increase the number of fulfilled ride requests. It makes those instances harder to solve.

Let $e_i = \sum_{j=1}^{n} c_{ij}, \forall i \in N$ and $z_\alpha = \frac{\sum_{i=1}^{n} e_i}{n}$. The q_i value, $1 \leq i \leq n$, is uniformly generated from [100,299] if $e_i \leq z_\alpha$, otherwise it is uniformly generated from [300,700]. The objective is to make high cost vertices, i.e., those for which $e_i > z_\alpha$, more attractive for the salesman than low cost vertices. The time required to collect the prize available at the i-th vertex, g_i, is uniformly generated from [1,39] if $e_i \leq z_\alpha$, otherwise from [40,79]. The quota value is computed by $K = \epsilon(\sum_{i=1}^{n} q_i)$, where ϵ is uniformly generated from [0.4,0.6].

The form of the names of the instances is $X - Y - Z$, where X, Y and Z stand for the class, the number of vertices and the vehicle capacity, respectively. Table 2 summarizes the parameters adopted in the instance generation algorithm.

The parameters reported in Table 2 were defined based on ad-hoc experiments conducted by us to test our operational constraints with hard boundary conditions.

5.2. Parameter tuning

We used the IRACE software, presented by (López-Ibáñez et al., 2016), to tune the parameters of the ant-based algorithms. We submitted 20 symmetric and 20 asymmetric instances to adjust the parameters. Those instances were selected at random. The IRACE uses the *maxExperiments* and *maxTime* parameters as stopping criteria. We set $maxExperiments = 10^3$ and $maxTime = \infty$. Table 3 presents the parameters produced by the IRACE for the ant-based algorithms.

5.3. Results of the optimization solver

Table 4 shows the instances for which the optimization solver was able to produce results within the time limit. It shows the name of the instance (column *Instance*), the type (symmetric or asymmetric), the value of the best solution found (column *UB*), and the percent deviation of the best solution from the lower bound computed by the solver (column *GAP(%)*).

Table 3
Values of the parameters of the ant-based algorithms.

	Asymmetric			Symmetric		
Parameter	AS	ACS	MS-ACS	AS	ACS	MS-ACS
MaxIter	–	20	19	–	13	14
m	344	38	43	409	36	66
α	0.15	6.10	2.39	0.21	9.60	3.16
β	3.31	8.33	7.79	3.11	6.64	9.21
ρ	0.18	0.98	0.86	0.12	0.47	0.30
ϕ	–	0.97	0.59	–	0.77	0.48
$q0$	–	0.82	0.96	–	0.95	0.74

Table 4
Results of the optimization solver for the QTSP-PIC.

	Asymmetric		Symmetric	
Instance	UB	GAP(%)	UB	GAP(%)
A-10-3	431.58	35.00	545.92	0.00
A-10-4	467.60	55.00	460.00	41.00
A-10-5	638.08	83.60	412.73	75.00
A-20-3	5515.00	100.00	6931.00	99.00
A-20-4	5941.00	100.00	3936.00	99.50
A-20-5	*	–	18904.00	100.00
B-10-3	729.50	54.00	834.67	33.80
B-10-4	395.60	81.80	578.73	81.00
B-10-5	431.30	73.70	748.35	71.50
B-20-3	5549.00	100.00	1946.00	100.00
B-20-4	4504.00	100.00	*	–
B-20-5	3733.00	100.00	*	–
C-10-3	356.33	100.00	513.33	35.27
C-10-4	362.80	100.00	412.57	96.17
C-10-5	569.67	95.55	533.80	87.14
C-20-3	*	–	2871.75	100.00
C-20-4	2867.67	100.00	*	–
C-20-5	*	–	2399.17	100.00

Eq. (36) shows the formula to calculate the percent deviation, where *LB* stands for the lower bound. The * symbol means that the solver stopped due to primary memory limitation or got stuck in the linear-programming relaxation phase and did not produce any solution. In these cases, the - symbol implies that there is no GAP to report.

$$GAP = \frac{|(LB - UB)|}{|UB|} \tag{36}$$

The optimization tool was able to find only one optimal result, instance *A-10-3*, and spent 22087s in this case. The solver was able to produce solutions to instances up to 20 vertices from the three classes. Regarding the percent deviation from the lower bound, 10-vertice class C instances were the hardest ones for the solver. It may be explained by the fact that the penalties regarding the drop-offs in alternate destinations were relaxed for the class C instances. So, the algorithm evaluates more ride-matching schemes, and the

Table 5
Distance between the results obtained by the solver and the ant algorithms.

	Asymmetric			Symmetric		
Instance	AS	ACS	MS-ACS	AS	ACS	MS-ACS
A-10-3	0.11	0.11	0.11	0.11	0.11	0.00
A-10-4	0.29	0.12	0.12	0.00	0.36	0.00
A-10-5	-0,15	-0.15	-0.24	0.06	0.05	-0.09
B-10-3	0.07	0.25	0.00	0.00	0.00	0.00
B-10-4	-0.22	0.00	-0.22	-0.14	-0.14	-0.14
B-10-5	0.01	0.30	0.01	0.09	0.08	-0.02
C-10-3	0.01	0.32	0.01	0.09	0.19	0.09
C-10-4	-0.12	-0.15	-0.15	0.05	0.04	-0.01
C-10-5	0.04	0.21	-0.01	-0.17	-0.22	-0.23

Table 6
Comparison between the Naive heuristic and the ant algorithms.

	Asymmetric			Symmetric		
	AS	ACS	MS-ACS	AS	ACS	MS-ACS
NH1	7 x 65	11 x 61	0 x 72	3 x 69	7 x 65	0 x 72
NH2	1 x 71	6 x 66	0 x 72	0 x 72	2 x 70	0 x 72
NH3	0 x 72	1 x 71	0 x 72	1 x 71	1 x 71	1 x 71
NH4	0 x 72	0 x 72	0 x 72	0 x 72	1 x 71	0 x 72
NH5	1 x 71	2 x 70	0 x 72	1 x 71	1 x 71	0 x 72

number of feasible solutions is higher than those of the instances from the other classes.

5.4. Results of the heuristic algorithms

In this section, we report the results of the ant algorithms and compare them to those produced by the optimization solver and the five versions of the naive heuristics. We also compared the variants of the ant algorithm.

Table 5 presents the comparison between the results of the ant algorithms and the solver for 10-vertice instances. The results show the distance, υ, between the best result found by an ant algorithm and the solution obtained by the solver. Eq. (37) shows the formula to compute υ, where χ_{ant} and χ^* denote the best solution found by the ant algorithm and the solution produced by the solver, respectively. A positive value shows that the solution obtained by the solver was better than that obtained by the ant algorithm. A negative value shows the opposite.

Table 5 shows that the results of the ant algorithms are close to those obtained by the solver. The MS-ACS algorithm was the best one for this set of instances. It was able to find the same results found by the solver for 4 instances and improve the solution found by the solver for 9 instances.

$$\upsilon = \frac{\chi_{ant}}{\chi^*} - 1 \tag{37}$$

Table 6 presents a summary of the comparison between the naive heuristics and the ant algorithms. The data presented in Table 6 comprises 72 symmetric and 72 asymmetric instances. We compared the best results obtained by each method. The results are in the $X \times Y$ format, where X and Y stand for the number of instances in which the naive heuristic found the best solution and the number of instances in which the ant algorithm found the best solution, respectively. Tables A.12–A.15 (Appendix A) present detailed results of the naive heuristics. Tables B.16–B.19 (Appendix B) show the results of the ant algorithms.

The *NH* heuristics build solutions by searching for the shortest path to achieving the minimum quota. The ant algorithms behave differently from those heuristics. The ants frequently selected the arcs that allowed satisfying more travel requests. It occurred since the ants update pheromone trails based on the cost of each arc in a route. According to Eq. (28), there is a reduction of the costs of the arcs, since they are divided by the number of passengers in the vehicle.

Table 7
Comparison of the ant algorithms.

	Asymmetric			Symmetric		
	AS	ACS	MS-ACS	AS	ACS	MS-ACS
AS	–	39 x 31	2 x 66	–	28 x 41	4 x 64
ACS	31 x 39	–	0 x 68	41 x 28	–	1 x 69
MS-ACS	66 x 2	68 x 0	–	64 x 4	69 x 1	–

Table 6 shows that the ant algorithms provided results with better quality compared with *NH* heuristics. From this table, we can infer that proposed ant algorithms improved the initial results of the *NH* heuristics and is reasonable to say that the ant algorithms proved to fit the QTSP-PIC.

Table 7 presents the comparison between the ant algorithms. This Table follows the same method of comparison presented in Table 6. Table 7 shows that the MS-ACS was the algorithm that reported the best solution for most instances. The best behavior of the MS-ACS results from the expansion of the decision-making process which includes different heuristic information. It is advantageous for diversifying the search for better solutions.

We can also infer from Table 7 that the AS is provides results with better quality than the ACS in the most asymmetric cases. In the symmetric scenarios, the ACS is superior to the AS in the most cases. Both algorithms uses the pheromone trails as search history to guide the ants. But the ACS concentrate its search around the best solutions found during the search. We observed that in many asymmetric instances, this behaviour lead the ants of the ACS to reuse a set of edges selected in the initial iterations and the search process get stuck in the local minima. These conclusions are also supported by the data from Table 8 that ranks the ant algorithms based on the Friedman test (Friedman, 1937) with the Nemenyi (1963) post-hoc test.

The first column of Table 8 presents the name of the subsets of instances grouped according to their sizes. The other columns of Table 8 present the p-values of the Friedman test and the ranks from the Nemenyi post-hoc test. In the post-hoc test, the order scales from a to c. The a rank indicates that the algorithm achieved the best performance in comparison to the others. The c rank indicates the opposite. If the performances of two or more algorithms are similar, the test assigns the same rank for them.

Regarding a 0.05 significance level, the p-values presented in the Table 8 show that the performance of the ant algorithms was not similar, i.e., the null hypothesis(Nemenyi, 1963) is rejected in all cases. In this Table, we can observe that AS rank higher than ACS for the subsets $g20$, $g30$, $g40$ and $g200$ of asymmetric instances. The ACS rank higher than AS only in the asymmetric subset $g500$.

In the case of symmetric instances, the ACS rank higher than AS for the subsets $g100$, $g200$ and $g500$. The AS rank higher than ACS only in the subset $g20$. In this subset, AS appears with same rank of MS-ACS. The MS-ACS rank higher in the majority of the subsets of this experiment. So we conclude that is the most promising ant algorithm of this study.

To analyze the variability of the results provided by each ant algorithm compared to the best results so far for the benchmarking set, we adopted three metrics regarding the results produced by the experiments. 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 $\chi^{\min}$ of each ant algorithm. The third

Table 8
The p-value from the Friedman's test and the rank of the ant algorithms provided by the Nemenyi post-hoc test.

	Asymmetric				Symmetric			
Subset	p-value	AS	ACS	MS-ACS	p-value	AS	ACS	MS-ACS
g10	0.019034	b	b	a	0.012778	b	b	a
g20	0.000611	b	c	a	0.003096	a	b	a
g30	0.000816	b	c	a	0.004320	b	b	a
g40	0.000585	b	c	a	0.000912	b	b	a
g50	0.004828	b	b	a	0.001139	b	b	a
g100	0.000912	b	b	a	0.000585	c	b	a
g200	0.000300	b	c	a	0.000123	c	b	a
g500	0.000123	c	b	a	0.000300	c	b	a

Table 9
Variability of the ants algorithms.

	Asymmetric			Symmetric		
Metric	*AS*	*ACS*	*MS − ACS*	*AS*	*ACS*	*MS − ACS*
ν	2.7%	1.04%	19.30%	4%	3%	20%
Φ	0.148644	0.134575	0.000320	0.177144	0.11579	0.001828
Ω	0.197718	0.205496	0.062074	0.230225	0.195750	0.066787

Table 10
Average time spent by the algorithms for the set of symmetric instances.

n	AS	ACS	MS-ACS	NH1	NH2	NH3	NH4	NH5
10	0.05	0.06	0.12	5.62	0.37	5.07	0.01	0.01
20	0.10	0.13	0.27	45.95	1.49	42.42	0.02	0.02
30	0.18	0.24	0.49	141.59	3.75	138.94	0.03	0.038
40	0.28	0.38	0.73	306.63	8.49	293.99	0.09	0.07
50	0.40	0.56	5.41	299.49	24.47	294.02	0.12	0.13
100	8.13	13.51	29.17	*	415.16	*	0.18	0.23
200	22.94	51.92	112.58	*	*	*	0.66	1.01
500	40.53	75.72	127.83	*	*	*	4.42	22.07

Table 11
Average time spent by the algorithms for the set of asymmetric instances.

n	AS	ACS	MS-ACS	NH1	NH2	NH3	NH4	NH5
10	0.05	0.06	0.12	6.26	0.29	5.97	0.01	0.01
20	0.10	0.13	0.26	42.64	1.03	41.75	0.02	0.02
30	0.20	0.30	1.9	250.65	3.19	231.31	0.032	0.04
40	0.34	0.48	2.29	240.99	7.60	223.86	0.05	0.092
50	0.41	2.20	5.77	388.36	17.99	372.86	0.11	0.13
100	6.68	28.85	32.69	*	388.04	*	0.21	0.48
200	31.81	270.94	409.72	*	*	*	1.03	6.11
500	41.72	3477.13	3545.20	*	*	*	12.21	371.62

metric, Ω, is the relative distance between χ^* and the average solution χ^a of each ant algorithm. To calculate Φ, we used Eq (38). Ω is calculated using the formula (39). We report the average values of ν, Φ and Ω in Table 9.

$$\Phi = \frac{\chi^{min}}{\chi^*} - 1 \tag{38}$$

$$\Omega = \frac{\chi^{a}}{\chi^*} - 1 \tag{39}$$

From Table 9, we observes that the *MS-ACS* is the best one concerning this three metrics. In this experiment, we observe that AS is better than ACS in metric ν and ACS is better than AS in metric Φ. In the asymmetric instances, AS is better than ACS in metric Ω. In this same metric, ACS achieved a best result compared to AS in symmetric scenarios. Tables B.16–B.19 (Appendix B) and Tables B.20–B.21 (Appendix B) show the data regarding the results reported in Table 9.

Tables 10 and 11 present the average time (in seconds) spent by each heuristic. Instances are grouped by number of vertices. In these tables, the * symbol denotes that the solver finished its execution due to excessive consume of primary memory. The results show that the naive heuristics that use exact procedures were not viable to solve large instances.

From Tables 10 and 11, we can also conclude that the AS was the fastest ant algorithm and the NH4 was the fastest naive heuristic. Tables A.12–A.15 (Appendix A) and Tables B.16–B.19 (Appendix B) present detailed results concerning the average time required by these heuristics.

6. Conclusions

The problem presented in this study, named QTSP-PIC, has a significant social and environmental impact and contributes to propose improvements to life quality regarding sustainable means. It is a variant of the QTSP contextualized within the shared mobility segment.

We presented the general problem and its mathematical formulation. Since the QTSP-PIC is a new problem, we created a set of instances for the experiments. The number of vertices of the graphs

that represent QTSP-PIC instances ranges from 10 to 500. We submitted the mathematical model of those instances to an optimization tool and limited the processing time to 80,000 seconds for each case. The optimization tool managed to solve only one instance with ten vertices.

We proposed naive heuristics and ant algorithms for the QTSP-PIC. We implemented three ant algorithms, namely AS, ACS, and MS-ACS.

The AS follows the basic scheme presented by Dorigo and Stützle (2003). A ride-matching heuristic proposed in this study assigns passengers to the route built by the ant algorithm. The ACS is an AS variant with a local search. We introduced a local search with multiple neighborhood operators. The central idea of the MS-ACS is to use different sources of heuristic information for the ants. We designed different heuristic strategies to cover all aspects of the problem and avoid trapping into local optima. There are four heuristic information sources. Every ant decides which information source to use at random.

The ant algorithms allowed, through pheromone, considering information about the search history concerning the route and the advantage of the ridesharing simultaneously. So, it was not necessary to design more complex mechanisms within the algorithms to take into account those information types.

We tested the performances of the ant algorithms on 144 instances up to 500 vertices. The MS-ACS provided the best results.

In future works, we will investigate novel mathematical formulations for the QTSP-PIC. We will also investigate learning techniques to be used by the ants of the MS-ACS algorithm to choose the source of information regarding the instance type and the search space processed. Other meta-heuristic techniques can also be applied to the problem addressed.

Appendix A. Results of the naive heuristics

Table A.12
Results of the NH heuristics for small symmetric instances.

	Symmetric									
Instance	*NH1*	*Time*	*NH2*	*Time*	*NH3*	*Time*	*NH4*	*Time*	*NH5*	*Time*
A-10-3	1172.00	4.04	1172.00	0.38	1172.00	3.00	742.00	0.01	700.75	0.01
A-10-4	971.83	4.35	971.83	0.41	849.00	3.96	618.17	0.01	618.17	0.01
A-10-5	1018.75	9.32	1018.75	0.45	738.17	8.91	738.17	0.01	515.80	0.01
A-20-3	1226.50	19.97	1426.00	1.54	1109.00	18.69	1872.00	0.01	1509.00	0.01
A-20-4	479.33	48.66	479.33	1.42	913.33	47.45	708.40	0.02	708.40	0.02
A-20-5	692.00	77.99	692.00	2.03	812.67	73.07	781.88	0.01	543.08	0.01
A-30-3	830.75	64.37	892.25	3.22	1103.83	61.43	1909.00	0.04	1550.75	0.04
A-30-4	1100.50	86.50	1154.00	4.45	1512.37	80.86	2237.00	0.02	1738.58	0.03
A-30-5	808.50	219.31	819.83	5.11	1021.47	215.69	1108.42	0.03	839.57	0.05
A-40-3	1152.83	231.00	1233.67	13.00	1625.83	219.00	2614.50	0.05	1575.58	0.07
A-40-4	882.83	655.00	1277.83	6.60	1259.00	611.00	1830.60	0.05	1590.00	0.06
A-40-5	626.80	529.00	951.00	9.24	1123.72	533.00	1341.38	0.40	1039.55	0.05
B-10-3	1105.00	4.84	1105.00	0.47	834.67	3.67	834.67	0.02	834.67	0.02
B-10-4	978.25	7.81	1011.25	0.40	649.00	7.77	649.00	0.03	493.58	0.03
B-10-5	1208.50	4.85	1208.50	0.31	1208.50	4.45	1428.00	0.03	851.90	0.03
B-20-3	1313.00	29.16	1313.00	1.17	1449.00	27.85	1371.00	0.05	1351.83	0.05
B-20-4	1452.00	47.58	1485.50	2.22	1184.10	46.05	1452.33	0.02	1396.33	0.02
B-20-5	1246.50	47.57	1246.50	1.62	1095.75	45.77	1192.67	0.02	1003.37	0.02
B-30-3	1104.17	119.83	1291.67	3.83	1704.75	116.12	1740.67	0.04	1313.83	0.04
B-30-4	891.17	137.15	1195.00	3.34	1501.50	134.40	1980.50	0.03	1780.67	0.03
B-30-5	694.50	268.47	1019.17	2.07	1203.00	267.12	1460.42	0.04	1060.22	0.06
B-40-3	1118.33	252.78	1395.83	7.26	1662.00	237.60	1835.75	0.04	1496.50	0.04
B-40-4	1431.08	409.86	1574.08	10.56	1514.00	402.60	2177.00	0.10	1743.50	0.20
B-40-5	1067.58	506.88	1110.08	9.90	1395.58	462.00	1668.00	0.10	1539.62	0.10
C-10-3	636.67	4.76	612.50	0.31	558.08	4.22	612.50	0.01	648.75	0.01
C-10-4	530.83	5.99	571.25	0.31	530.83	5.05	496.25	0.01	496.25	0.01
C-10-5	544.80	4.67	504.33	0.33	682.92	4.62	563.03	0.01	563.03	0.01
C-20-3	1169.83	35.93	1067.92	1.15	906.67	34.20	972.00	0.02	911.75	0.02
C-20-4	786.08	74.78	872.40	1.21	933.08	57.89	1225.62	0.02	1028.80	0.03
C-20-5	973.33	31.96	953.10	1.12	1255.67	30.83	1129.00	0.01	1108.10	0.02
C-30-3	1148.58	116.41	1429.75	4.80	1428.75	110.34	1626.42	0.03	1163.08	0.03
C-30-4	832.38	85.14	1059.65	3.12	1278.92	81.78	1387.67	0.03	1498.45	0.03
C-30-5	719.58	177.16	965.80	3.86	764.68	182.73	1044.07	0.04	702.83	0.04
C-40-3	*	*	1674.25	7.19	*	*	1214.92	0.04	1116.08	0.05
C-40-4	*	*	1181.40	8.87	*	*	1437.80	0.06	1370.60	0.06
C-40-5	*	*	960.50	9.56	*	*	1314.67	0.04	1138.83	0.07

Table A.13
Results of the NH heuristics for medium and large symmetric instances.

	Symmetric									
Instance	*NH*1	*Time*	*NH*2	*Time*	*NH*3	*Time*	*NH*4	*Time*	*NH*5	*Time*
A-50-3	1258.25	462.00	1812.00	24.42	2054.17	457.00	2747.92	0.10	2203.83	0.10
A-50-4	1213.60	762.96	1222.00	22.44	1883.58	735.00	2868.50	0.08	1702.83	0.09
A-50-5	*	*	1967.33	18.48	*	*	1871.95	0.08	1093.43	0.09
A-100-3	*	*	3360.67	461.03	*	*	3739.83	0.19	2898.50	0.22
A-100-4	*	*	2149.47	398.55	*	*	2247.72	0.19	1848.58	0.23
A-100-5	*	*	2236.08	273.25	*	*	2184.95	0.19	1634.38	0.26
A-200-3	*	*	*	*	*	*	4402.00	0.67	4084.58	0.85
A-200-4	*	*	*	*	*	*	3837.03	0.66	3062.80	1.08
A-200-5	*	*	*	*	*	*	4390.17	0.64	3582.15	0.74
A-500-3	*	*	*	*	*	*	7994.17	4.12	7771.58	15.28
A-500-4	*	*	*	*	*	*	6680.80	4.52	6090.40	45.86
A-500-5	*	*	*	*	*	*	6680.80	4.52	6090.40	45.86
B-50-3	1626.50	523.38	2160.33	21.78	2020.50	498.96	2278.33	0.07	2184.42	0.07
B-50-4	*	*	1644.50	29.70	*	*	1813.82	0.30	1930.07	0.30
B-50-5	982.40	952.08	1320.75	15.18	1857.50	960.30	2902.17	0.30	1758.38	0.30
B-100-3	*	*	2878.67	272.48	*	*	3948.67	0.18	3548.17	0.20
B-100-4	*	*	3522.67	361.55	*	*	4590.67	0.18	3413.75	0.19
B-100-5	*	*	2645.75	775.25	*	*	2425.72	0.19	1778.53	0.29
B-200-3	*	*	*	*	*	*	5224.00	0.65	4098.00	0.86
B-200-4	*	*	*	*	*	*	3715.62	0.68	3465.78	1.28
B-200-5	*	*	*	*	*	*	5799.10	0.69	2935.52	1.12
B-500-3	*	*	*	*	*	*	8827.75	4.21	7915.25	11.84
B-500-4	*	*	*	*	*	*	6921.38	4.25	6414.02	15.43
B-500-5	*	*	*	*	*	*	6802.53	4.63	5546.78	14.51
C-50-3	*	*	1468.75	26.11	*	*	1953.75	0.09	1623.92	0.10
C-50-4	*	*	1440.65	44.02	*	*	2100.25	0.07	1727.30	0.08
C-50-5	*	*	1517.10	18.17	*	*	2046.18	0.06	2074.90	0.12
C-100-3	*	*	2400.83	418.50	*	*	2887.92	0.20	2300.25	0.22
C-100-4	*	*	1591.60	357.69	*	*	2298.80	0.19	1574.00	0.25
C-100-5	*	*	1416.92	418.22	*	*	2063.50	0.18	1651.98	0.21
C-200-3	*	*	*	*	*	*	4237.25	0.64	3673.00	0.96
C-200-4	*	*	*	*	*	*	3819.45	0.64	3145.20	0.72
C-200-5	*	*	*	*	*	*	2372.00	0.67	2330.17	1.44
C-500-3	*	*	*	*	*	*	8071.00	4.15	7524.58	11.31
C-500-4	*	*	*	*	*	*	6377.48	3.91	6167.82	6.98
C-500-5	*	*	*	*	*	*	5793.27	5.48	5219.33	31.64

Table A.14
Results of the NH heuristics for small asymmetric instances.

	Asymmetric									
Instance	*NH*1	*Time*	*NH*2	*Time*	*NH*3	*Time*	*NH*4	*Time*	*NH*5	*Time*
A-10-3	823.00	2.83	823.00	0.30	478.42	2.58	823.00	0.02	823.00	0.02
A-10-4	959.25	3.71	1044.50	0.32	959.25	3.37	1044.50	0.01	523.57	0.01
A-10-5	934.50	8.55	934.50	0.28	934.50	8.34	934.50	0.01	720.50	0.01
A-20-3	901.17	33.31	930.17	1.02	1164.50	32.24	837.67	0.03	771.25	0.04
A-20-4	955.00	33.35	955.00	0.93	901.67	32.31	1352.00	0.02	878.83	0.02
A-20-5	589.42	50.63	618.92	1.04	810.33	49.88	1113.00	0.03	889.27	0.04
A-30-3	1101.17	89.92	1095.33	2.64	1700.17	87.59	1299.67	0.03	1585.83	0.03
A-30-4	502.98	212.46	792.80	3.14	739.55	210.02	1179.03	0.02	1143.90	0.03
A-30-5	745.63	145.30	1243.67	3.57	1087.20	142.37	1516.00	0.02	1089.55	0.03
A-40-3	781.92	238.26	1258.58	4.62	1573.75	226.90	2126.83	0.06	1539.75	0.08
A-40-4	863.87	442.86	1283.33	5.28	1833.57	474.17	1494.40	0.04	1266.55	0.05
A-40-5	1017.50	544.50	1212.50	7.26	1024.92	501.60	1166.97	0.05	806.78	0.07
B-10-3	947.00	4.54	947.00	0.34	947.00	4.21	947.00	0.01	901.83	0.02
B-10-4	814.00	7.25	814.00	0.28	395.67	6.96	814.00	0.01	547.67	0.01
B-10-5	786.33	9.31	786.33	0.27	786.33	9.06	786.33	0.01	681.67	0.01
B-20-3	1177.50	38.58	1177.50	0.85	1321.25	41.60	1376.33	0.03	1342.33	0.03
B-20-4	1390.00	27.68	1390.00	0.97	1544.50	26.34	1579.00	0.02	1415.67	0.02
B-20-5	1361.00	33.95	1361.00	1.00	1354.83	32.67	1559.50	0.01	1365.50	0.01
B-30-3	1496.00	177.76	1627.17	3.46	1756.17	172.15	2258.50	0.04	1945.67	0.05
B-30-4	1225.00	173.10	1491.00	3.00	1999.83	171.57	2299.50	0.02	1840.17	0.03
B-30-5	977.50	236.71	1274.50	2.62	1333.02	235.63	2322.42	0.04	1403.27	0.05
B-40-3	1575.00	228.36	1654.00	5.28	2411.00	206.58	3061.00	0.03	2031.00	0.05
B-40-4	1618.00	217.00	1634.00	9.90	3304.00	207.90	3505.00	0.10	2772.00	0.20
B-40-5	1506.00	500.94	1918.50	11.22	2638.03	400.62	2799.83	0.10	1718.00	0.10
C-10-3	484.83	6.28	546.33	0.38	494.17	5.86	485.42	0.01	485.42	0.02
C-10-4	402.50	9.27	463.20	0.16	402.50	9.11	463.20	0.01	450.30	0.01
C-10-5	578.00	4.65	595.25	0.34	623.83	4.24	710.75	0.01	710.75	0.01
C-20-3	705.08	50.95	877.33	1.12	941.33	49.32	1228.17	0.02	982.33	0.02
C-20-4	1001.25	56.69	999.40	1.22	1022.33	54.81	1205.10	0.03	1019.98	0.04
C-20-5	757.33	58.62	920.22	1.16	1169.33	56.64	1130.45	0.04	1130.45	0.04
C-30-3	995.17	87.94	1035.00	2.87	1436.75	91.59	1949.25	0.04	1621.92	0.04
C-30-4	1062.17	219.19	1188.23	3.89	1330.78	669.20	1489.62	0.04	1429.30	0.06
C-30-5	869.32	913.52	968.23	3.60	998.33	301.68	1268.72	0.04	1090.68	0.08
C-40-3	*	*	1230.42	8.20	*	*	2164.25	0.04	1781.83	0.05
C-40-4	*	*	1395.55	8.34	*	*	2006.23	0.06	1676.87	0.10
C-40-5	*	*	1127.72	8.38	*	*	2202.97	0.05	1243.75	0.13

Table A.15
Results of the NH heuristics for medium and large asymmetric instances.

	Asymmetric									
Instance	*NH1*	*Time*	*NH2*	*Time*	*NH3*	*Time*	*NH4*	*Time*	*NH5*	*Time*
A-50-3	1499.50	191.00	2072.17	11.88	2087.00	171.60	3683.00	0.06	4034.75	0.08
A-50-4	887.65	614.46	1725.50	19.14	1948.83	608.52	2763.73	0.07	1800.93	0.09
A-50-5	1120.50	705.54	1693.33	18.57	1631.65	663.30	2101.65	0.05	2030.98	0.08
A-100-3	*	*	2878.67	272.48	*	*	5140.50	0.21	3185.00	0.50
A-100-4	*	*	3522.67	361.55	*	*	4142.23	0.22	3320.90	0.30
A-100-5	*	*	2645.75	775.25	*	*	3222.02	0.21	2502.80	0.46
A-200-3	*	*	*	*	*	*	6279.83	1.07	6121.92	4.40
A-200-4	*	*	*	*	*	*	6710.75	1.01	5252.67	3.35
A-200-5	*	*	*	*	*	*	3988.88	1.02	3775.77	11.71
A-500-3	*	*	*	*	*	*	16613.40	11.39	15037.70	53.12
A-500-4	*	*	*	*	*	*	12934.70	11.43	12480.80	126.93
A-500-5	*	*	*	*	*	*	9392.30	14.94	9011.07	361.66
B-50-3	2044.00	273.24	2381.00	15.18	3449.00	269.94	4973.00	0.05	3274.00	0.07
B-50-4	1695.00	946.44	2081.00	19.80	3114.00	902.22	3655.50	0.30	2798.00	0.30
B-50-5	1998.00	767.58	2565.00	21.78	2753.00	743.16	2900.00	0.30	2234.00	0.30
B-100-3	*	*	3455.67	359.02	*	*	5144.42	0.21	4267.92	0.41
B-100-4	*	*	2831.58	323.68	*	*	3602.68	0.21	3315.77	0.55
B-100-5	*	*	3157.20	251.68	*	*	3761.15	0.21	3761.15	0.45
B-200-3	*	*	*	*	*	*	9448.75	1.03	7999.42	3.51
B-200-4	*	*	*	*	*	*	7759.43	1.02	6004.88	4.67
B-200-5	*	*	*	*	*	*	4927.37	1.03	4413.57	7.65
B-500-3	*	*	*	*	*	*	18131.80	11.49	15587.80	189.71
B-500-4	*	*	*	*	*	*	12636.60	11.43	12055.00	613.32
B-500-5	*	*	*	*	*	*	10181.80	12.35	9840.28	676.18
C-50-3	*	*	1528.67	19.00	*	*	2184.25	0.06	2115.17	0.08
C-50-4	*	*	1764.33	18.28	*	*	3018.60	0.06	2582.60	0.11
C-50-5	*	*	723.90	18.28	*	*	2105.90	0.05	1760.50	0.12
C-100-3	*	*	1933.83	251.07	*	*	4255.17	0.21	3346.75	0.47
C-100-4	*	*	2712.70	274.45	*	*	4434.55	0.21	3883.15	0.43
C-100-5	*	*	2332.27	623.26	*	*	3061.67	0.21	2138.00	0.79
C-200-3	*	*	*	*	*	*	6553.75	1.03	5938.83	5.26
C-200-4	*	*	*	*	*	*	6458.45	1.03	5037.18	6.49
C-200-5	*	*	*	*	*	*	4484.33	1.03	4045.73	7.99
C-500-3	*	*	*	*	*	*	14455.50	11.56	13580.30	176.89
C-500-4	*	*	*	*	*	*	11749.50	11.52	11099.10	781.76
C-500-5	*	*	*	*	*	*	9921.12	13.79	9406.67	365.03

Appendix B. Results of ant-based algorithms

Table B.16
Results of the ant-based algorithms for small symmetric instances.

	AS			ACS			MS-ACS		
Instance	*Best*	*AVG*	*Time*	*Best*	*AVG*	*Time*	*Best*	*AVG*	*Time*
A-10-3	606.50	670.65	0.05	606.50	668.00	0.07	545.92	587.46	0.12
A-10-4	460.00	553.69	0.05	625.73	647.35	0.06	460.00	492.50	0.13
A-10-5	435.93	474.52	0.06	434.05	453.65	0.09	371.93	398.47	0.16
A-20-3	679.75	848.52	0.09	955.83	1092.48	0.11	786.33	958.83	0.22
A-20-4	346.30	430.65	0.12	500.90	645.50	0.13	348.48	394.34	0.33
A-20-5	413.97	451.31	0.11	473.70	583.70	0.13	432.40	468.85	0.29
A-30-3	652.00	810.03	0.14	915.42	1125.70	0.16	695.17	798.10	0.35
A-30-4	915.25	970.92	0.13	1044.00	1220.60	0.17	912.25	970.23	0.35
A-30-5	572.93	605.48	0.18	579.30	685.76	0.30	541.48	587.16	0.56
A-40-3	922.92	1024.33	0.23	1158.17	1257.32	0.31	906.00	1000.70	0.59
A-40-4	694.68	713.83	0.32	776.87	842.69	0.48	608.75	681.51	0.85
A-40-5	560.92	589.58	0.29	656.62	775.62	0.42	557.92	564.02	0.79
B-10-3	834.67	868.47	0.05	834.67	869.57	0.05	834.67	854.80	0.11
B-10-4	493.58	556.78	0.05	493.58	539.09	0.07	493.58	511.03	0.13
B-10-5	812.97	820.37	0.06	811.67	891.88	0.09	726.35	842.76	0.16
B-20-3	1291.83	1304.67	0.09	1015.42	1361.05	0.11	953.83	1090.60	0.22
B-20-4	1016.67	1065.65	0.09	1016.75	1170.27	0.11	822.82	1016.75	0.23
B-20-5	849.08	915.03	0.10	913.07	981.02	0.11	687.13	738.57	0.24
B-30-3	950.33	1030.30	0.17	875.17	957.02	0.21	792.75	918.03	0.43
B-30-4	948.05	997.14	0.21	855.02	936.53	0.23	715.48	884.37	0.53
B-30-5	602.88	671.92	0.21	655.38	656.60	0.34	510.13	612.13	0.56
B-40-3	1142.25	1226.18	0.27	1137.00	1332.55	0.34	975.50	1010.37	0.66
B-40-4	1021.00	1087.74	0.26	1069.63	1245.72	0.38	714.05	901.46	0.70
B-40-5	994.58	1044.01	0.28	837.85	1065.05	0.39	830.03	902.11	0.74
C-10-3	558.67	589.53	0.04	612.50	633.15	0.05	558.67	613.95	0.10
C-10-4	434.60	434.60	0.05	428.80	428.80	0.06	408.45	424.73	0.11
C-10-5	437.73	507.29	0.05	411.07	454.00	0.07	409.60	453.44	0.14
C-20-3	666.08	734.35	0.13	768.58	902.85	0.14	613.83	740.92	0.30
C-20-4	481.85	659.41	0.14	787.18	872.42	0.21	441.65	596.91	0.39
C-20-5	742.57	831.79	0.10	776.38	968.34	0.14	713.55	815.86	0.27
C-30-3	1021.83	1061.35	0.18	1125.25	1230.07	0.23	923.42	1006.23	0.46
C-30-4	859.35	952.65	0.19	776.50	952.89	0.22	770.15	860.68	0.46
C-30-5	708.17	756.34	0.29	656.67	773.15	0.37	518.33	671.35	0.71
C-40-3	777.75	809.07	0.32	761.75	855.95	0.41	629.00	770.50	0.82
C-40-4	722.60	812.02	0.34	820.55	1036.37	0.36	686.60	733.48	0.71
C-40-5	654.33	681.19	0.50	721.67	808.52	0.87	475.50	638.95	15.89

Table B.17
Results of the ant-based algorithms for medium and large symmetric instances.

	AS			ACS			MS-ACS		
Instance	*Best*	*AVG*	*Time*	*Best*	*AVG*	*Time*	*Best*	*AVG*	*Time*
A-50-3	1186.58	1382.03	0.31	1301.75	1523.38	0.43	1041.25	1190.12	0.85
A-50-4	1062.80	1106.44	0.36	1014.62	1053.05	0.45	862.03	998.79	0.96
A-50-5	694.97	735.78	0.48	680.90	707.84	0.63	553.85	682.10	1.31
A-100-3	1827.33	1947.92	0.99	1860.33	2018.90	1.23	1576.08	1699.58	24.95
A-100-4	1454.30	1492.00	12.89	1403.27	1454.37	21.11	1269.28	1328.74	34.44
A-100-5	1167.05	1193.84	13.63	1063.95	1159.77	19.03	999.42	1046.91	37.00
A-200-3	3276.75	3464.38	35.07	3083.67	3281.62	54.29	2731.17	2932.77	87.59
A-200-4	2489.87	2548.28	4.73	2346.55	2454.79	73.55	2265.92	2311.55	126.35
A-200-5	2657.80	2733.24	32.19	2458.17	2621.34	43.36	2113.38	2240.55	79.78
A-500-3	7158.33	7229.30	41.23	6760.42	6847.89	81.63	6445.75	6532.27	108.38
A-500-4	5605.08	5783.13	91.21	5035.35	5139.81	133.43	5007.38	5191.13	328.15
A-500-5	5618.95	5819.34	20.24	4751.40	4823.16	28.84	4651.00	4793.28	57.18
B-50-3	1245.67	1317.23	0.36	1306.67	1384.48	0.57	1068.92	1195.90	1.04
B-50-4	1051.73	1120.98	0.38	1085.18	1196.22	0.62	931.52	1040.11	10.46
B-50-5	956.65	1019.23	0.39	817.88	940.56	0.53	748.58	854.38	1.07
B-100-3	2576.17	2795.90	0.96	2264.08	2491.07	11.48	1920.75	2042.80	23.29
B-100-4	2632.20	2813.67	0.89	2102.77	2652.72	1.06	2063.63	2221.03	2.19
B-100-5	1312.12	1341.51	17.23	1270.80	1365.09	29.59	1043.22	1142.69	48.26
B-200-3	3625.67	3676.45	34.59	2963.00	3179.53	48.48	2913.67	3025.05	90.87
B-200-4	2748.80	2817.57	4.46	2532.42	2619.81	69.69	2397.13	2455.75	122.99
B-200-5	2218.13	2341.24	51.94	2034.30	2095.53	9.42	1912.87	1972.98	150.29
B-500-3	7541.25	7633.09	30.89	6641.42	6783.36	96.86	4415.60	4537.48	94.18
B-500-4	5818.20	5937.41	45.38	5229.83	5344.13	110.64	5139.48	5224.54	131.72
B-500-5	5089.38	5233.11	31.94	4594.72	4783.45	55.40	4415.60	4537.48	94.18
C-50-3	1122.42	1216.93	0.36	1001.50	1129.42	0.45	896.75	1017.65	0.93
C-50-4	1243.78	1296.77	0.42	1349.03	1482.22	0.53	954.85	1168.31	10.73
C-50-5	880.37	1028.34	0.60	1057.67	1232.51	0.84	840.23	957.49	21.36
C-100-3	1656.50	1694.68	1.09	1537.00	1734.67	1.27	1376.33	1452.47	25.79
C-100-4	1262.00	1304.66	14.14	1193.80	1340.64	21.93	1067.92	1151.15	37.67
C-100-5	1158.95	1259.97	11.42	1241.28	1373.09	14.91	1029.47	1156.45	28.94
C-200-3	2742.50	2915.02	3.51	2675.75	2720.22	4.50	2564.58	2603.33	87.84
C-200-4	2466.38	2549.03	33.51	2361.07	2415.72	45.32	2106.93	2211.53	85.47
C-200-5	1963.67	1995.41	6.46	1888.83	1928.63	118.68	1721.17	1790.24	182.05
C-500-3	6886.36	7017.39	31.84	6161.09	6308.32	62.41	6190.75	6248.55	91.50
C-500-4	5490.10	5549.37	18.29	4985.18	5059.24	21.78	4976.39	5097.15	43.28
C-500-5	4623.17	4732.39	53.75	4138.84	4264.46	90.52	4029.83	4157.45	201.98

Table B.18
Results of the ant-based algorithms for small asymmetric instances.

	AS			ACS			MS-ACS		
Instance	*Best*	*AVG*	*Time*	*Best*	*AVG*	*Time*	*Best*	*AVG*	*Time*
A-10-3	478.42	478.42	0.05	478.42	616.27	0.06	478.42	478.42	0.13
A-10-4	601.08	601.08	0.05	523.57	523.57	0.07	523.57	523.57	0.12
A-10-5	542.00	545.14	0.05	542.00	644.18	0.07	482.60	552.28	0.11
A-20-3	624.58	664.62	0.09	773.42	858.08	0.12	545.50	716.42	0.24
A-20-4	674.33	778.77	0.09	795.63	984.28	0.11	439.67	588.52	0.23
A-20-5	483.17	506.80	0.09	696.25	765.45	0.13	420.78	576.98	0.26
A-30-3	866.25	976.02	0.14	1065.08	1128.60	0.19	760.00	827.45	0.35
A-30-4	451.97	524.57	0.18	606.67	661.52	0.27	489.85	515.22	0.48
A-30-5	576.30	644.73	0.15	683.58	858.84	0.22	539.62	608.37	0.40
A-40-3	898.33	925.63	0.26	912.33	1146.00	0.45	764.75	803.00	0.76
A-40-4	835.42	905.52	0.27	907.38	1071.28	0.39	753.15	845.69	0.79
A-40-5	583.17	622.87	0.29	572.53	714.56	0.53	482.98	582.09	0.83
B-10-3	778.83	842.57	0.04	913.00	913.00	0.05	729.50	865.70	0.09
B-10-4	306.90	377.91	0.04	395.67	395.67	0.05	306.90	395.67	0.10
B-10-5	434.75	545.34	0.05	560.67	633.51	0.07	434.75	479.17	0.12
B-20-3	937.25	976.37	0.09	893.25	1069.87	0.11	893.25	997.62	0.22
B-20-4	1029.83	1079.67	0.08	1305.00	1337.37	0.10	895.25	1114.57	0.22
B-20-5	897.83	946.77	0.08	1128.83	1188.57	0.10	897.00	975.33	0.20
B-30-3	944.17	1004.62	0.15	1021.75	1153.87	0.21	861.08	973.22	0.42
B-30-4	966.42	1039.24	0.16	996.38	1094.91	0.23	771.73	937.93	0.46
B-30-5	841.92	971.15	0.18	1044.67	1189.71	0.24	702.00	899.36	0.46
B-40-3	1164.83	1299.57	0.25	1359.25	1498.23	0.36	951.00	1149.05	0.68
B-40-4	1727.75	1781.68	0.22	1969.50	2025.20	0.31	1403.00	1637.12	0.56
B-40-5	947.03	1073.87	0.27	1215.62	1371.71	0.38	932.48	1097.92	0.76
C-10-3	359.25	359.25	0.05	469.67	481.13	0.06	359.25	438.42	0.12
C-10-4	318.40	344.39	0.07	307.10	383.17	0.08	307.10	393.40	0.17
C-10-5	595.25	601.02	0.05	689.87	698.22	0.06	566.58	585.25	0.12
C-20-3	766.83	769.30	0.12	769.92	811.12	0.16	650.25	745.98	0.32
C-20-4	688.17	741.88	0.13	747.05	966.93	0.20	613.83	764.22	0.33
C-20-5	887.50	905.85	0.14	950.27	1050.01	0.19	693.05	765.28	0.36
C-30-3	925.00	954.65	0.19	1013.08	1101.75	0.27	869.75	936.75	0.47
C-30-4	891.50	937.38	0.26	873.37	967.62	0.36	796.37	937.02	0.67
C-30-5	678.17	704.85	0.47	708.97	879.73	0.71	613.10	751.52	13.39
C-40-3	1195.17	1282.12	0.30	1407.33	1472.42	0.42	1089.83	1137.03	0.85
C-40-4	844.15	907.41	0.50	1092.20	1160.85	0.74	787.30	901.87	14.17
C-40-5	800.67	879.33	0.71	723.90	744.04	0.76	674.90	676.68	1.29

Table B.19
Results of the ant-based algorithms for medium and large asymmetric instances.

	AS			ACS			MS-ACS		
Instance	*Best*	*AVG*	*Time*	*Best*	*AVG*	*Time*	*Best*	*AVG*	*Time*
A-50-3	1541.58	1642.07	0.29	1421.25	1661.65	0.44	1259.42	1410.55	0.80
A-50-4	939.83	1024.66	0.33	1142.40	1324.53	0.55	867.35	926.20	0.93
A-50-5	937.92	998.11	0.35	876.00	1022.22	0.60	728.85	923.74	10.69
A-100-3	1678.33	1764.65	11.38	1750.50	1777.30	25.60	1488.33	1526.65	41.51
A-100-4	1955.88	2045.87	0.97	2111.87	2211.30	23.14	1531.23	1700.35	3.23
A-100-5	1392.50	1491.58	11.06	1456.48	1650.13	27.44	1172.48	1262.39	41.00
A-200-3	3099.08	3218.08	38.57	3160.00	3239.28	406.13	2710.25	2799.92	425.12
A-200-4	2742.58	2809.49	3.63	2678.35	2814.21	329.02	2324.87	2448.07	323.95
A-200-5	2022.92	2117.25	64.16	1936.20	1956.10	111.30	1782.52	1820.53	102.28
A-500-3	8366.25	8432.40	17.31	7547.50	7603.90	656.10	7048.50	7104.10	465.00
A-500-4	6101.20	6239.04	19.49	5412.28	5608.40	1363.30	5238.50	5316.89	1460.10
A-500-5	4969.50	5110.50	34.72	4606.50	4734.33	3230.39	4440.37	4478.59	3501.97
B-50-3	1881.75	1924.23	0.30	1715.08	1970.27	0.44	1536.50	1696.75	0.84
B-50-4	1439.13	1468.57	0.37	1328.25	1553.53	0.68	1084.82	1320.08	10.90
B-50-5	1461.37	1596.70	0.34	1507.12	1576.20	0.59	1178.52	1334.71	0.93
B-100-3	2344.67	2459.05	1.02	2164.92	2255.85	19.64	2015.08	2209.33	37.90
B-100-4	1786.82	1918.52	1.23	1753.35	1889.29	4.05	1504.97	1561.61	5.27
B-100-5	2050.45	2264.07	10.22	2056.47	2390.37	27.34	1504.07	1616.79	42.01
B-200-3	3900.25	4246.38	3.35	3636.08	3913.93	23.26	3302.42	3412.52	322.30
B-200-4	3024.77	3130.56	39.48	2731.17	2836.57	29.86	2532.07	2707.25	495.66
B-200-5	2367.53	2508.99	52.07	2281.67	2372.81	78.99	2097.10	2160.59	893.19
B-500-3	8148.00	8320.15	20.95	7216.43	7395.19	1169.30	6886.08	6731.50	2106.44
B-500-4	5983.57	6062.66	47.45	5815.18	5972.26	7310.88	5517.34	5639.27	7628.10
B-500-5	4951.02	5137.85	45.89	4672.04	4794.44	5928.32	4541.48	4631.19	5614.49
C-50-3	1500.25	1542.87	0.36	1477.58	1625.90	0.48	1288.92	1452.02	0.93
C-50-4	1227.32	1303.70	0.56	1361.88	1539.36	0.88	953.35	1171.58	1.76
C-50-5	664.58	841.59	0.86	904.62	1029.37	15.15	694.10	797.29	24.15
C-100-3	1622.75	1747.95	11.12	1769.58	1853.85	25.64	1452.75	1530.63	40.07
C-100-4	1977.80	2062.60	11.75	1970.30	2216.51	28.06	1496.57	1704.36	4.32
C-100-5	1150.90	1213.31	1.42	1250.50	1281.35	78.80	987.00	1092.96	78.92
C-200-3	2951.50	2986.25	36.04	2693.75	2773.15	272.38	2670.00	2693.32	328.37
C-200-4	2472.20	2510.93	3.91	2319.40	2390.87	597.19	2186.08	2234.64	314.12
C-200-5	2047.00	2137.73	45.13	1962.33	2066.18	590.39	1878.00	1909.56	482.50
C-500-3	6978.56	7028.78	20.37	6557.42	6694.16	1826.19	6470.34	6528.72	1356.29
C-500-4	5640.84	5782.28	46.23	5236.85	5368.92	6894.38	5078.54	5128.40	6901.70
C-500-5	4773.83	4842.15	26.78	4381.17	4425.46	2915.37	4193.38	4257.33	2872.73

Table B.20
Percentage of best solutions found regarding small instances.

	Asymmetric			Symmetric		
Instance	*AS*	*ACS*	*MS − ACS*	*AS*	*ACS*	*MS − ACS*
A-10-3	100%	100%	100%	0%	0%	30%
A-10-4	0%	25%	20%	40%	0%	35%
A-10-5	0%	0%	15%	0%	0%	50%
A-20-3	0%	0%	10%	25%	0%	0%
A-20-4	0%	0%	20%	15%	0%	0%
A-20-5	0%	0%	10%	20%	0%	0%
A-30-3	0%	0%	20%	20%	0%	0%
A-30-4	10%	0%	0%	0%	0%	30%
A-30-5	0%	0%	20%	0%	0%	20%
A-40-3	0%	0%	40%	0%	0%	15%
A-40-4	0%	0%	20%	0%	0%	20%
A-40-5	0%	0%	20%	0%	0%	20%
B-10-3	0%	0%	40%	40%	15%	40%
B-10-4	20%	0%	15%	20%	35%	100%
B-10-5	80%	0%	10%	0%	0%	40%
B-20-3	0%	40%	40%	0%	0%	10%
B-20-4	0%	0%	20%	0%	0%	40%
B-20-5	0%	0%	25%	0%	0%	5%
B-30-3	0%	0%	20%	0%	0%	20%
B-30-4	0%	0%	10%	0%	0%	60%
B-30-5	0%	0%	20%	0%	0%	40%
B-40-3	0%	0%	15%	0%	0%	20%
B-40-4	0%	0%	5%	0%	0%	20%
B-40-5	0%	0%	20%	0%	0%	10%
C-10-3	100%	0%	35%	20%	0%	25%
C-10-4	0%	60%	55%	0%	0%	10%
C-10-5	0%	0%	15%	0%	0%	40%
C-20-3	0%	0%	10%	0%	0%	10%
C-20-4	0%	0%	5%	0%	0%	5%
C-20-5	0%	0%	15%	0%	0%	15%
C-30-3	0%	0%	5%	0%	0%	20%
C-30-4	0%	0%	20%	0%	0%	10%
C-30-5	0%	0%	5%	0%	0%	40%
C-40-3	0%	0%	25%	0%	0%	20%
C-40-4	0%	0%	5%	0%	0%	20%
C-40-5	0%	0%	100%	0%	0%	40%

Table B.21
Percentage of best solutions found regarding medium and large instances.

	Asymmetric			Symmetric		
Instance	*AS*	*ACS*	*MS − ACS*	*AS*	*ACS*	*MS − ACS*
A-50-3	0%	0%	15%	0%	0%	20%
A-50-4	0%	0%	20%	0%	0%	35%
A-50-5	0%	0%	20%	0%	0%	20%
A-100-3	0%	0%	20%	0%	0%	5%
A-100-4	0%	0%	5%	0%	0%	15%
A-100-5	0%	0%	15%	0%	0%	5%
A-200-3	0%	0%	10%	0%	0%	25%
A-200-4	0%	0%	20%	0%	0%	20%
A-200-5	0%	0%	10%	0%	0%	5%
A-500-3	0%	0%	5%	0%	0%	15%
A-500-4	0%	0%	10%	0%	0%	15%
A-500-5	0%	0%	100%	0%	0%	5%
B-50-3	0%	0%	5%	0%	0%	20%
B-50-4	0%	0%	10%	0%	0%	10%
B-50-5	0%	0%	10%	0%	0%	10%
B-100-3	0%	0%	10%	0%	0%	20%
B-100-4	0%	0%	15%	0%	0%	20%
B-100-5	0%	0%	5%	0%	0%	25%
B-200-3	0%	0%	10%	0%	0%	5%
B-200-4	0%	0%	10%	0%	0%	10%
B-200-5	0%	0%	5%	0%	0%	5%
B-500-3	0%	0%	5%	0%	0%	20%
B-500-4	0%	0%	5%	0%	0%	5%
B-500-5	0%	0%	10%	0%	0%	20%
C-50-3	0%	0%	100%	0%	0%	20%
C-50-4	0%	0%	40%	0%	0%	20%
C-50-5	0%	0%	20%	0%	0%	15%
C-100-3	0%	0%	5%	0%	0%	20%
C-100-4	0%	0%	20%	0%	0%	10%
C-100-5	0%	0%	15%	0%	0%	5%
C-200-3	0%	0%	25%	0%	0%	15%
C-200-4	0%	0%	20%	0%	0%	20%
C-200-5	0%	0%	5%	0%	0%	10%
C-500-3	0%	0%	10%	0%	25%	0%
C-500-4	0%	0%	10%	0%	0%	5%
C-500-5	0%	0%	5%	0%	0%	15%

CRediT authorship contribution statement

Bruno C.H. Silva: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - original draft, Writing - review & editing. **Islame F.C. Fernandes:** Methodology, Data curation, Writing - original draft. **Marco C. Goldbarg:** Conceptualization, Supervision, Validation, Writing - review & editing. **Elizabeth F.G. Goldbarg:** Supervision, Validation, Writing - review & editing.

References

Agatz, N., Erera, A., Savelsbergh, M., Wang, X., 2012. Optimization for dynamic ride-sharing: a review. Eur. J. Oper. Res. 223 (2), 295–303. doi:10.1016/j.ejor.2012.05.028.

Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., Rus, D., 2017. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. Proc. Natl. Acad. Sci. 114 (3), 462–467. doi:10.1073/pnas.1611675114.

Armant, V., Brown, K.N., 2020. Fast optimised ridesharing: objectives, reformulations and driver flexibility. Expert Syst. Appl. 141, 112914. doi:10.1016/j.eswa.2019.112914.

Awerbuch, B., Azar, Y., Blum, A., Vempala, S., 1998. New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen. Ann. Phys. (N Y) 28, 254–262. doi:10.1137/S009753979528826X.

Chen, Z., Liu, X.C., Wei, R., 2018. Agent-based approach to analyzing the effects of dynamic ridesharing in a multimodal network. Comput. Environ. Urban Syst. doi:10.1016/j.compenvurbsys.2018.10.004.

Cordeau, J.F., Laporte, G., 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transp. Res. Part B Methodol. 37, 579–594. doi:10.1016/S0191-2615(02)00045-0.

Cordeau, J.F., Laporte, G., 2007. The dial-a-ride problem: models and algorithms. Ann. Oper. Res. 153, 29–46. doi:10.1007/s10479-007-0170-8.

Croes, G.A., 1958. A method for solving traveling salesman problems. Oper. Res. 6, 791–812. doi:10.1287/opre.6.6.791.

Dong, Y., Wang, S., Li, L., Zhang, Z., 2018. An empirical study on travel patterns of internet based ride-sharing. Transp. Res. Part C Emerg. Technol. 86, 1–22. doi:10.1016/j.trc.2017.10.022.

Dorigo, M., Gambardella, L.M., 1997. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. 1, 53–66. doi:10.1109/4235.585892.

Dorigo, M., Maniezzo, V., Colorni, A., 1996. Ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. Part B (Cybernetics) 26 (1), 29–41. doi:10.1109/3477.484436.

Dorigo, M., Stützle, T., 2003. The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances. In: Handbook of Metaheuristics. Springer US, Boston, MA, pp. 250–285. doi:10.1007/0-306-48056-5_9.

Fagnant, D.J., Kockelman, K.M., 2018. Dynamic ride-sharing and fleet sizing for a system of shared autonomous vehicles in austin, texas. Transportation (Amst) 45 (1), 143–158. doi:10.1007/s11116-016-9729-z.

Farhan, J., Chen, T.D., 2018. Impact of ridesharing on operational efficiency of shared autonomous electric vehicle fleet. Transp. Res. Part C Emerg. Technol. 93, 310–321. doi:10.1016/j.trc.2018.04.022.

Feo, T.A., Resende, M.G., 1989. A probabilistic heuristic for a computationally difficult set covering problem. Oper. Res. Lett. 8 (2), 67–71. doi:10.1016/0167-6377(89)90002-3.

Fiedler, D., Čáp, M., Čertický, M., 2017. Impact of mobility-on-demand on traffic congestion: Simulation-based study. In: Proceedings of the IEEE 20th International Conference on Intelligent Transportation Systems(ITSC), pp. 1–6. doi:10.1109/ITSC.2017.8317830.

Friedman, M., 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J. Am. Stat. Assoc. 32, 675–701. doi:10.1080/01621459.1937.10503522.

Gurobi, 2018. Gurobi optimizer. http://www.gurobi.com/products/gurobi-optimizer. Accessed in: 2018-03-12.

Heikkila, S., 2014. Mobility as a Service: A Proposal for Action for the Public Administration, Case Helsinki. Aalto University.

Herbawi, W., Weber, M., 2011. 2011 ieee 23rd international conference on tools with artificial intelligence, pp. 282–288. doi:10.1109/ICTAI.2011.50.

Ho, S.C., Szeto, W.Y., Kuo, Y., Leung, J.M.Y., Petering, M., Tou, T.W.H., 2018. A survey of dial-a-ride problems: literature review and recent developments. Transp. Res. Part B: Methodol. 111, 395–421. doi:10.1016/j.trb.2018.02.001.

Hou, L., Li, D., Zhang, D., 2018. Ride-matching and routing optimisation: models and a large neighbourhood search heuristic. Transp. Res. Part E: Logist. Transp. Rev. 118, 143–162. doi:10.1016/j.tre.2018.07.003.

Huang, S.C., Jiau, M.K., Liu, Y.P., 2018. An ant path-oriented carpooling allocation approach to optimize the carpool service problem with time windows. IEEE Syst. J. 1–12. doi:10.1109/JSYST.2018.2795255.

Karp, H.M., 1975. On the computational complexity. Networks 5, 45–68. doi:10.1002/net.1975.5.1.45.

Levin, M.W., Kockelman, K.M., Boyles, S.D., Li, T., 2017. A general framework for modeling shared autonomous vehicles with dynamic network-loading and dynamic ride-sharing application. Comput. Environ. Urban Syst. 64, 373–383. doi:10.1016/j.compenvurbsys.2017.04.006.

Liaw, R., Chang, Y., Ting, C., 2017. Advances in Swarm Intelligence. Springer International Publishing, pp. 293–300. doi:10.1007/978-3-319-61824-1_32.

de Lira, V.M., Perego, R., Renso, C., Rinzivillo, S., Times, V.C., 2018. Boosting ride sharing with alternative destinations. IEEE Trans. Intell. Transp. Syst. 1–11. doi:10.1109/TITS.2018.2836395.

Lokhandwala, M., Cai, H., 2018. Dynamic ride sharing using traditional taxis and shared autonomous taxis: a case study of nyc. Transp. Res. Part C Emerg. Technol. 97, 45–60. doi:10.1016/j.trc.2018.10.007.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T., 2016. The irace package: iterated racing for automatic algorithm configuration. Oper. Res. Perspect. 3, 43–58. doi:10.1016/j.orp.2016.09.002.

Masson, R., Trentin, A., Lehuédé, F., Malhéné, N., Péton, O., Tlahig, H., 2017. Optimization of a city logistics transportation system with mixed passengers and goods. EURO Journal on Transportation and Logistics 6 (1), 81–109. doi:10.1007/s13676-015-0085-5.

Miller, C.E., Tucker, A.W., Zemlin, R.A., 1960. Integer programming formulation of traveling salesman problems. J. ACM 7 (4), 326–329. doi:10.1145/321043.321046.

Nemenyi, P.B., 1963. Distribution-free multiple comparisons. Princeton University.

Nourinejad, M., Roorda, M., 2014. A dynamic carsharing decision support system. Transp. Res. 66, 36–50. doi:10.1016/j.tre.2014.03.003.

Reinelt, G., 1991. Tsplib - a traveling salesman problem library. ORSA J. Comput. 3 (4), 376–384. doi:10.1287/ijoc.3.4.376.

Riedler, M., Raidl, G., 2018. Solving a selective dial-a-ride problem with logic-based benders decomposition. Comput. Oper. Res. 96, 30–54. doi:10.1016/j.cor.2018.03.008.

Rosenkrantz, D., Stearns, R., P. Lewis, I., 1977. An analysis of several heuristics for the traveling salesman problem. SIAM J. Comput. 6 (3), 563–581. doi:10.1137/0206041.

Savelsbergh, M., Sol, M., 1995. The general pickup and delivery problem. Transp. Sci. 29, 17–29. doi:10.1287/trsc.29.1.17.

Schönberger, J., Kopfer, H., Mattfeld, D.C., 2003. A combined approach to solve the pickup and delivery selection problem. In: Operations Research Proceedings 2002. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 150–155. doi:10.1007/978-3-642-55537-4_24.

Simonin, G., O'Sullivan, B., 2014. Optimisation for the ride-sharing problem: A complexity-based approach. In: Proceedings of the Twenty-first European Conference on Artificial Intelligence. IOS Press, pp. 831–836. doi:10.3233/978-1-61499-419-0-831.

Stiglic, M., Agatz, N., Savelsbergh, M., Gradisar, M., 2018. Enhancing urban mobility: integrating ride-sharing and public transit. Comput. Oper. Res. 90, 12–21. doi:10.1016/j.cor.2017.08.016.

Tripathy, T., Nagavarapu, S.C., Azizian, K., Pandi, R.R., Dauwels, J., 2018. Advances in Computational Intelligence Systems. Springer International Publishing, pp. 325–336. doi:10.1007/978-3-319-66939-7_28.

Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., Subramanian, A., 2017. New benchmark instances for the capacitated vehicle routing problem. Eur. J. Oper. Res. 257 (3), 845–858. doi:10.1016/j.ejor.2016.08.012.

Wang, Y., Zheng, B., Lim, E., 2018. Understanding the effects of taxi ride-sharing a case study of singapore. Comput. Environ. Urban Syst. 69, 124–132. doi:10.1016/j.compenvurbsys.2018.01.006.

Zhang, C., Xie, J., Wu, F., Gao, X., Chen, G., 2018. Algorithm designs for dynamic ridesharing system. In: Tang, S., Du, D., Woodruff, D., Butenko, S. (Eds.), Algorithmic Aspects in Information and Management. Springer International Publishing, Cham, pp. 209–220. doi:10.1007/978-3-030-04618-7_17.

Zhao, M., Yin, J., An, S., Wang, J., Feng, D., 2018. Ridesharing problem with flexible pickup and delivery locations for app-based transportation service: mathematical modeling and decomposition methods. J. Adv. Transp. 2018, 1–21. doi:10.1155/2018/643095.

Zufferey, N., Farres, J., Glardon, R., 2015. Ant metaheuristic with adapted personalities for the vehicle routing problem. In: Computational Logistics. Springer International Publishing, Cham, pp. 3–15. doi:10.1007/978-3-319-24264-4_1.