2021 7th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS) | 978-1-7281-8995-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/MT-ITS49943.2021.9529319

An ant colony algorithm with penalties for the dial-a-ride problem with time windows and capacity restriction

1st Qinrui Tang
Institute of Transportation Systems
German Aerospace Center (DLR)
Berlin, Germany
Qinrui.Tang@dlr.de

2nd Maria Giuliana Armellini
Institute of Transportation Systems
German Aerospace Center (DLR)
Berlin, Germany
Maria.Armellini@dlr.de

***Abstract*—This paper proposes a metaheuristic algorithm to solve the dial-a-ride problem (DARP) with time windows and capacity restrictions. The algorithm was developed for the project "HubChain", which proposes an on demand system being integrated with local public transport in the Elde rural region, in northern Germany. Users can request their trips via an online platform by providing the origin and destination as well as the desired arrival or departure time. To solve the problem, ant colony optimization with penalties (ACOP), in which the ants communicate pheromones both locally and globally and meanwhile the constraints are handled by setting penalties, is developed. To validate the results, the proposed algorithm and an exact algorithm were run for multiple test scenarios using the simulation SUMO as a framework. The routes obtained with the proposed algorithm show travel times comparable to the optimal routes, yet obtained in low computation times. This allows therefore the implementation of the proposed ACOP algorithm in a dynamic booking system.**

I. INTRODUCTION

Thanks to technological advances and improvements in digitization in recent years, on-demand mobility services have become a booming market and are the subject of research in different cities worldwide. These shared services are especially being studied as a possible solution for mobility in low demand areas and for the last mile problem. In both cases, public transport cannot provide an adequate service due to low frequency and long and indirect routes, while taxi services can offer a flexible and direct service but with high costs. On-demand systems fill the gap between the latter two, as they try to bundle requests to minimize the number of vehicles and the length of routes without compromising passenger travel times.

Fig. 1 shows the system architecture of an on-demand transport system. A passenger first requests a trip from their origin to destination via passenger interface (e.g. web service/smart phone App) (1). After receiving the trip request, the on-demand disposition module inquires the fleet and journey information according to the request date, the origin and the destination (2). If there is an existing journey, the request could be merged into the journey; if not, a new journey will start. Meanwhile, the on-demand disposition module inquires the travel time between two different locations from the intermodal router module via inputting their coordinate pairs (3). The intermodal router module can provide the calibrated travel time based on the data collected from the real world. Then the on-demand disposition module provides a route with time schedule in reasonable travel costs, and saves it into the database when the route is feasible (4). The route and time schedule are sent to both driver interface and passenger interface (5). The passenger can accept/cancel the request and the decision is also stored in the database (6). The problem solved by the disposition module is known as the dial-a-ride problem (DARP) and more specifically to the dynamic case, as the route is optimized once a new request comes, which means the existing plans could be modified.

The objectives of a DARP could be to minimize the total travel time, the fleet size or the waiting time of passengers. Previous researchers have developed many algorithms to solve a DARP. To obtain the optimal result, exact algorithms, such as Branch-and-Bound, Branch-and-Cut and Branch-and-Price, are frequently used. Cordeau [2] developed a Branch-and-Cut algorithm for a static, multiple vehicles and single depot DARP, which became a solid base for similar problems. Although exact algorithms can find the optimal routes, they

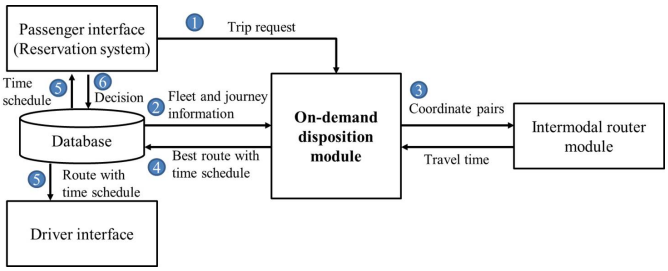

Fig. 1. System architecture of the studied on-demand transportation system

This work has been supported by Federal Ministry for Economic Affairs and Energy, Germany via project "HubChain".

are time-consuming such that it is not possible to be applied in the DARP with a big problem size. The largest problem size being solved by an exact algorithm is 96 requests with eight cars, taking 899 seconds [3], which is unsuitable for the fast-response in practice. Therefore, it is necessary to develop heuristic or meta-heuristic algorithms, such as tabu search [4], large neighborhood search [5] and genetic algorithm [6]. However, as one of meta-heuristic algorithms, the ant colony optimization (ACO) algorithm seems to be paid only little attention.

The ACO algorithm was first developed by Dorigo, Maniezzo and Colorni [7] to solve a traveling salesman problem (TSP). They were inspired by the process of ants searching food. Once an ant finds a shorter route, it will release pheromones to mark the route, so other ants can communicate via pheromones to find the shortest route with high pheromone density. Dorigo, Maniezzo and Colorni [7] simulated the process to find the shortest route among a list of nodes defining the critical rules of selecting the next node (transition probability) and updating the global pheromones. Although the algorithm proved to be closer to the optimal result compared against tabu search and simulated annealing algorithms, it cannot solve big size problems, so Dorigo and Gambardella [8] improved the ACO algorithm. Additional to improving the rules of transition probability and global updating, Dorigo and Gambardella [8] introduced a local updating on pheromones, resulting in finding more near-optimal and stabler best results in shorter time. Later, the ACO algorithms were used to solve a vehicle routing problem which require the vehicle to start and end at the same depot with vehicle capacity constraint [9]–[11] and time windows [12]. More recently, Tripathy et al. [13] and Ho et al. [14] solve a DARP with constraints using ACO algorithms. The constraints could refer to time window constraint requiring vehicle departing and arrival in an interval, vehicle capacity constraint requiring the number of passengers/goods not exceeding its capacity, riding time constraint requiring the time not exceeding a maximum riding time, and so on. The manner to handle constraints of both papers is searching nodes only in the feasible space, but must handle the problem how to insert unvisited nodes into the journey, which consumes computing resources and increases the risks of cannot find feasible solutions even if feasible routes exist.

This paper proposes an ant colony optimization algorithm with penalties (ACOP) to solve the DARP of a real on-demand system in the Elde region, Germany. This on-demand transportation system is funded by the German Federal Ministry for Economic Affairs and Energy through the project "HubChain"[1]. In this system, passengers can book a tour in advance via the passenger interface by given the origin, destination and the desired departure or arrival time. Once a new request comes, the proposed ACOP search for the best routes meeting constraints. The system operates with two vehicles with a capacity of 10 passengers each and a single depot. To reduce the operation cost, if all constraints hold, the requests will be taken over by one vehicle as possible. Only if no feasible route can be found with one vehicle, a second vehicle will be assigned for the new request, which satisfies an global optimization. To analyze the ACOP, test scenarios were running using the microscopic simulation tool Eclipse SUMO (Simulation of Urban Mobility) [15] as framework. The scenarios were run with an exact algorithm developed by Armellini [16] and the optimal routes founded were compare to the results of the ACOP.

The contributions of the paper are (1) developing an ACO for a dynamic DARP with the ability to find comparably near optimal results in acceptable computing time; and (2) a good integration into a practice-oriented on-demand transportation system in Elde region, Germany.

The paper is first describes the problem in detail, and then presents the proposed ACOP, followed by the experimental study. Finally conclusions and perspectives are discussed.

II. PROBLEM STATEMENT

Passengers send a trip request to the on-demand system. A request must include the origin and destination information (street and number or coordinates) and either the earliest departure time or latest arrival time. In case of the first request, the vehicle that meet the aforementioned constrains is assigned to serve the request. Once a request is accepted and its pick-up and drop-off time are set, new constrains are added regarding the maximum time variation for the pick-up (time window departure) and drop-off (time window arrival). When a new request is made, the on-demand system will first try to serve it with the same vehicle. If this is not possible, the system will try to assign the request to an other vehicle. If no vehicles meet the constrains of the request, it will be fully rejected. The rules of the on-demand transportation system are listed in detail below:

- the tour starts and ends at a depot;
- at each node, there is a non-negative service time for the pickup and drop off of a passenger;
- a vehicle is allow to wait at a node if it is waiting for new requests or if it arrives earlier than the request pick-up time;
- the ride time of each trip can not exceed the maximum acceptable ride time;
- the vehicle only drives in its operating hours (i.e. a duration constraint);
- if a request is a new request, which means the earliest departure time or the latest arrival time is set, the passenger(s) will be picked up or dropped off within T_P time unit.
- if requests have been accepted, a pick-up and drop-off time window are set and the passenger(s) will be picked up or dropped off in these time interval;

The DARP is solved on a complete directed graph $G = (V, E)$ where V is the set of nodes, and E is the set of links. The node set V includes origin set $P = \{1, ..., n\}$, destination set $D = \{1 + n, ..., 2n\}$, origin depot $\{0\}$ and destination

[1]Project website: https://www.hubchain.de/

depot $\{2n+1\}$ where n denotes the number of requests, thus $V = P \cup D \cup \{0, 2n+1\}$.

As mentioned, only if no feasible route can be found for a vehicle, the second vehicle will be assigned. Thus, the following formulation aims to single vehicle. The vehicle has a capacity Q, and the number of passengers upon arrival at node i is $Q_i, \forall i \in V$ and $0 \leq Q_i \leq Q$. For each node $i \in V$, there is q_i passengers being picked up or dropped off. Specifically, $q_0 = q_{2n+1} = 0$ as no passengers are picked up or dropped off at the depot, and $q_i = -q_{i+n}, \forall i \in P$ holds for each request. A time window departure e_i, l_i could be associated with a pick-up node $i \in P$ and a time window arrival e_j, l_j could also be associated with a drop-off node $j \in D$. The earliest departure time e_0 and the latest arrival time l_{2n+1} of the depot are given, so the total tour duration $T_T = l_{2n+1} - e_0$ cannot be exceeded. We also denote service time $t_{S,i}$ and waiting time $t_{W,i}$ for node $i \in V$, and arrival time $t_{A,i}$ and departure time $t_{D,i}$ are also associated with the node. Thus, $t_{D,i} = t_{A,i} + t_{S,i} + t_{W,i}$ holds. The travel time between two nodes is denoted as $t_{i,j}, \forall i, j \in V, i \neq j$. The travel time for each passenger cannot exceed $N_R t_{i,i+n}, i \in P$ where N_R is a factor related to the direct travel time $t_{i,i+n}$.

A decision variable in this DARP is whether a link is taken by the vehicle. Let $x_{i,j}, \forall i, j \in V, i \neq j$ denote this decision variable and $x_{i,j} = 1$ if the vehicle travels from node i to node j; $x_{i,j} = 0$, otherwise. The decision variables $t_{A,i}, t_{D,i}, Q_i \forall i \in V$ are dependent on $x_{i,j}$. Therefore, the DARP is formulated as:

$$\min \sum_{i \in V} \sum_{j \in V, j \neq i} (t_{i,j} + t_{S,i})\, x_{i,j} \tag{1}$$

subject to

$$\sum_{j \in V} x_{j,i} - \sum_{j \in V} x_{i,j} = 0, \forall i \in P \cup D, \tag{2}$$

$$\sum_{j \in V} x_{0,j} = 1, \tag{3}$$

$$\sum_{i \in V} x_{i,2n+1} = 1, \tag{4}$$

$$(t_{A,i} + t_{S,i} + t_{W,i} + t_{i,j}) x_{i,j} \leq t_{A,j}, \forall i, j \in V, i \neq j, \tag{5}$$

$$t_{i,i+n} \leq t_{A,i+n} - t_{D,i} \leq N_R t_{i,i+n}, \forall i \in P, \tag{6}$$

$$(Q_i + q_i) x_{i,j} \leq \min\{Q_j, Q\}, \forall i, j \in V, i \neq j, \tag{7}$$

$$t_{A,2n+1} - t_{D,0} \leq l_{2n+1} - e_0. \tag{8}$$

$$t_{D,i} \leq e_i + T_P, \forall i \in P', \tag{9}$$

$$l_i - T_P \leq t_{A,i}, \forall i \in D', \tag{10}$$

$$e_i \leq t_{D,i} \leq l_i, \forall i \in P'', \tag{11}$$

$$e_i \leq t_{A,i} \leq l_i, \forall i \in D''. \tag{12}$$

where P', P'', D', D'' represent the origins of the requests having the earliest departure time, the time window departure, the destinations of the requests having the latest arrival time and the time window arrival, respectively.

The objective (1) is to minimize the total travel time of the tour. Constraint (2) requires that each node is visited from exactly one other node, and each node can only connect to exactly one other node. Constraint (3) and constraint (4) require the vehicle to start and end at the depot. Constraint (5) represents the rules of time sequence. Constraint (6) ensures that the ride time of a request does not exceed the maximum acceptable ride time and that the request is picked up before dropped off (arrival time at destination later than the departure time from origin). Constraint (7) displays the number of passengers on the vehicle at each node, which cannot exceed the vehicle capacity. Constraint (8) ensures that the vehicle only drives during the operation time. Finally, if the time constraints are set, the constraints (9) and (10) are applied on the new request setting the earliest departure time and latest arrival time respectively. If a request is accepted, the arrival and departure time windows must be satisfied for future routes (constraint (11) and (12) respectively).

III. ANT COLONY ALGORITHM WITH PENALTIES

An ACO algorithm simulates the behavior of ants to search for food using pheromones. Thus, the rules of communication using pheromone is the key in this algorithm. The ACOP proposed in this paper is based on the ant colony system developed by Dorigo and Gambardella [8]. As mentioned, the algorithm was introduced a local updating rule, whereas can only handle the constraints (2) - (5) in this paper. We thereby develop penalty functions to handle the constraints (6) - (12). The idea behind this is that infeasible routes cause higher travel costs due to penalty, so the pheromone level of these edges is low and they are selected with lower probabilities.

The functions $f(x_{i,j}) \leq 0$ represents the constraints (6) - (12). Then the objective function is formulated as

$$\min \sum_{i \in V} \sum_{j \in V, j \neq i} \left((t_{i,j} + t_{S,i})\, x_{i,j} + \sum_{\mathscr{P}} \lambda \max(0, f(x_{i,j})) \right), \tag{13}$$

where $\mathscr{P}$ is the penalty set referring to constraints (6) - (12), and λ are the penalty parameters.

The state transition rule, the global updating rule and the local updating rule, are explained below.

A. State transition rule

The state transition rule defines the way to select the next node. Usually the probability of a node being selected is

$$p_k(i,j) = \begin{cases} \frac{\tau(i,j) \cdot \eta(i,j)^\beta}{\sum_{u \in J(i)} \tau(i,u) \cdot \eta(i,u)^\beta}, & \text{if } j \in J_k(i) \\ 0, & \text{otherwise} \end{cases}. \tag{14}$$

where $k = 1, ..., N_A$ is an ant and N_A is the number of ants, $\eta(i,j)$ is the pheromone level of the edge, $J_k(i)$ is the set of end nodes of edge (i,j), β is a parameter, and $\tau_{i,j}$ is the edge travel cost defined as:

$$\tau_{i,j} = t_{i,j} + t_{S,i} + \sum_{\mathscr{P}} \lambda \max(0, f(x_{i,j})), \forall j \in J_k(i). \tag{15}$$

To make the edges with fewer costs more dominant, the following rule is applied:

$$j = \begin{cases} \arg\max_{u \in J_k} \left\{ \tau(i,j) \cdot \eta(i,j)^{\beta} \right\}, & \text{if } p' \leq p'_0 \\ J \text{ by Eq. (14)}, & \text{otherwise} \end{cases}. \tag{16}$$

where p' is a random number in the interval $[0, 1]$ and p'_0 is a parameter also in the interval $[0, 1]$, and J is the next node decided by Eq. (14).

Eq. (16) means that, for each time to select the next node with a given parameter p'_0, if the random number p' is less than or equal to p'_0, the node on the edge with the shortest travel cost is selected. But if the random number p' is grater than p'_0, the next node is selected based on its probability.

B. Global updating rule

At the end of each iteration, each ant has found a tour with a different travel cost. Then the ants can communicate by exchanging pheromones. The pheromone level is globally updated with the rule:

$$\tau(i,j) \leftarrow (1-\alpha) \cdot \tau(i,j) + \alpha \cdot \Delta\tau(i,j), \tag{17}$$

where

$$\Delta\tau(i,j) = \begin{cases} \frac{1}{L_{best}}, & \text{if } (i,j) \in \text{global} - \text{best} - \text{tour} \\ 0, & \text{otherwise} \end{cases}. \tag{18}$$

Notation $\alpha \in [0, 1]$ is a pheromone decay parameter, and L_{best} is the tour with the lowest cost found so far. Eq. (17) ensures that only the edges of the best global tour are considered.

C. Local updating rule

A local updating is applied every time when the next node is selected. Once an ant selects the next node in one iteration, it marks the edge linking the current node and the next node with pheromones. The other ants in the same iteration can select the next node based on the pheromones left by previous ants. As the global tour length has not been obtained at this stage, it is call local updating. The pheromone level is locally updated with the rule:

$$\tau(i,j) \leftarrow (1-\rho) \cdot \tau(i,j) + \rho \cdot \Delta\tau(i,j), \tag{19}$$

where $\rho \in [o, 1]$ is a parameter, $\Delta\tau(i,j)$ is set as τ_0, and τ_0 is the initial pheromone level.

Finally, the pseudo-codes of the algorithm can be seen in Algorithm 1.

Algorithm 1 ACOP

Require: $\tau_0, N_A, N_{I,max}, p'_0, \alpha, \beta, \rho, \tau_0, \lambda$ ▷ $N_{I,max}$ is the maximum iteration
Ensure: $bestTour, L_{best}$
function ACO
  for $i, j \in V$ **do** ▷ initialization
    $\tau(i,j) \leftarrow \tau_0$
  end for
  for $k = 1, ..., N_A$ **do**
    $i \leftarrow 0$ ▷ set depot as the first node
    $J_k(i) \leftarrow P$ ▷ set the set of origins as the set of yet to be visited node
    $Tour_k \leftarrow Tour_k + i$
  end for
  $bestTourCost \leftarrow$ a large constant
  for $iter = 1, ..., N_{I,max}$ **do** ▷ starting to solve
    for $w = 1, ..., 2n+1$ **do**
      if $w < 2n+1$ **then**
        for $k = 1, ..., N_A$ **do**
          $\tau_k(i,j) \leftarrow$ Eq. (15)
            ▷ calculate the travel costs with penalty
          $j \leftarrow$ Eq. (16) ▷ select the next node
          if $i \in P$ **then**
            $J_k(i) \leftarrow J_k(i) - i$
            $J_k(i) \leftarrow J_k(i) + (i+n)$
          end if
        end for
      else ▷ select the depot as the last node
        for $k = 1, ..., N_A$ **do**
          $j \leftarrow 2n+1$
        end for
      end if
      $t_{D,i}, t_{A,j} \leftarrow$ Eq. (5)
      $Tour_k \leftarrow Tour_k + j$
      for $k = 1, ..., N_A$ **do**
        $\tau_k(i,j) \leftarrow$ Eq. (19) ▷ local updating
        $i \leftarrow j$ ▷ move to the next node
      end for
    end for
    for $k = 1, ..., N_A$ **do**
      if $L_{best} < TourCost_k$ **then**
        $L_{best} \leftarrow TourCost_k$
        $bestTour \leftarrow Tour_k$
      end if
      for $i \in bestTour - (2n+1), j \in bestTour - 0$ **do**
        $\tau(i,j) \leftarrow$ Eq.(17) ▷ global updating
      end for
    end for
  end for
  return $bestTour, L_{best}$
end function

IV. EXPERIMENTAL STUDY

A. Configurations

Table I lists the values of the adopted parameters and constrains for the testing scenarios. For the ACOP parameters the following values are chosen: the number of ants (N_A) is the same as the number of requests, the maximum iteration number is 200, $p'_0 = 0.9$, $\alpha = 0.1$, $\beta = 2$,$\rho = 0.1$ and $\tau_0 = 0.002$. The penalty parameter for vehicle capacity is set to 1000, and for the rest of the penalty parameters 100 is taken.

B. Testing scenarios

The ACOP algorithm is tested in the Elde region in the north of Germany. The road network is imported to SUMO from OpenStreetMap. A depot (in blue) and 22 nodes (in red) are randomly generated in this road network (see Fig. 2). The nodes are numbered from 100 to 122, being node 100 the depot. The travel times between nodes are calculated using SUMO. For the test scenarios, 10 requests are generated by randomly choosing a start and end node, as well as an earliest departure or a latest arrival time. The basic information of the requests are summarized in Table II.

TABLE I
PARAMETERS FOR CONSTRAINTS

Parameter	Notation	Value
Vehicle capacity	Q	10 seats
Maximum ride time	N_R	4 times
Start of operation time	e_0	07:00:00
End of operation time	e_{2n+1}	11:30:00
Time window departure	$l_i - e_i, \forall i \in P$	30 min
Time window arrival	$l_j - e_j, \forall j \in D$	30 min
Maximum time variation in planned stop	T_P	30 min
Service time at each stop	$t_{S,i}, i \in P \cup D$	1 min
Maximum waiting time at each stop	$t_{W,i}, i \in P \cup D$	180 min

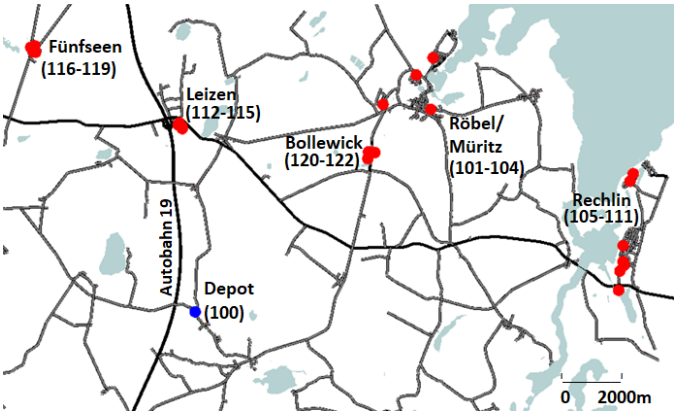

Fig. 2. Test scenario of the Elde region, Germany

TABLE II
TESTING REQUESTS IN SIMULATION STUDY

N°.	Number of passengers	Start node	End node	Earliest departure	Latest arrival
1	2	109	106	07:30:00	
2	2	102	113	08:00:00	
3	2	115	105		11:00:00
4	2	111	121	09:00:00	
5	1	112	103		09:30:00
6	2	114	117		11:00:00
7	1	120	116		08:00:00
8	1	119	108	07:40:00	
9	2	110	118		10:00:00
10	1	104	107	10:20:00	

C. Results

Table III lists the results of the proposed algorithm after each request arrives. The total travel time includes the service time at each stop. The request 1 to 4 can all be served by the first vehicle. When request 3 comes, the total waiting time of the vehicle rises to 7251 seconds. This is due to the difference between the earliest departure time of request 1 and 2 and the latest arrival time of request 3. Thus, the first vehicle has to wait at a stop until the passengers of request 3 can be picked up. When request 5 comes, the first vehicle cannot feasible handle this request, thereby the second vehicle is assigned to this request. The request 5 to 9 are also assigned to the second vehicle. When request 10 is received, the total travel time would be 7741 seconds with the first vehicle and 9691 seconds with the second vehicle, so the first vehicle is assigned to this request.

To validate these results, the same requests are again simulated in SUMO but this time using an exact algorithm developed by Armellini [16] to define the optimal routes. Different scheduling methods could influence the departure and arrival time of each request, influencing also the possible routes. To avoid this, the mentioned algorithm was modified to have the same scheduling methods, constrains and objective as the proposed ACOP, but still finding the optimal solution. It is found that the results can well match except when the request 9 arrives. The optimal route handling request 9 is "100 → 120 → 119 → 116 → 110 → 108 → 112 → 103 → 118 → 114 → 117 → 100" by the second vehicle with a total travel time of 7091 seconds. The result found with ACOP is 7096 seconds, which is only 0.07% longer than the optimal route. Thus, we can conclude that the ACOP was able to find very good results in the test scenarios and thereby meet the requirements of the project "HubChain".

Computing efficiency of ACOP is displayed in Table IV. The results were running in a laptop with Intel(R) Core(TM) i7-HQ CPU 2.70GHz and 16.0 Gb memory. The first column represents the scenario. The second and the third column summarize the number of nodes in each route which clarifies the problem size. The forth column lists the total computing time of both vehicles in millisecond. As the problem size increases, the computing time increases almost linearly. Generally the results can be found in acceptable time and thus, the proposed ACOP algorithm is suitable for applying in practice.

V. CONCLUSIONS

This paper proposes an ant colony algorithm with penalties to solve a dynamic DARP with time windows and capacity constraints. This algorithm is the core of a disposition module which that belongs to an on-demand transport system implemented in the Elde region, Germany. To validate the algorithm, test scenarios were simulated in SUMO and the results were compared with the optimal routes found by an exact algorithm [16]. The proposed ACOP algorithm was able to nearly find the same routes as the optimal ones, but with acceptable computing time for practical application. Thus, the proposed ACOP proves to be suitable for the project "HubChain", as well as for other on-demand systems with similar characteristics.

Despite these positive results, there is still room for improvement. The algorithm only takes vehicle travel time as an objective, causing in some cases that passengers have to wait a long time inside the vehicle for the next passenger to be picked up. Although the time window and maximum travel time restrictions are still met, passengers could not be satisfied with long waiting times. Another aspect to be evaluated in future work is the management of the fleet. Although the system has two vehicles, one vehicle is preferably used unless a request has to be rejected. A more flexible use of the fleet could significantly improve passenger travel times at the

TABLE III
RESULTS OF TEST SCENARIOS WITH ACOP ALGORITHM

Request N°.	Vehicle N°.	Total travel time (s)	Waiting Time(s)	Route
1	1	2242	0	100→109→106→100
2	1	3311	0	100→109→106→102→113→100
3	1	4620	7251	100→109→ 106→102→113→115→105→100
4	1	7119	4752	100→109→106→102→113→111→ 121 →115→105→ 100
5	1	7119	4752	100→109→106→102→113→111→ 121 →115→105→ 100
	2	1619	0	100→112→103→100
6	1	7119	4752	100→109→106→102→113→111→ 121 →115→105→ 100
	2	2837	2576	100→112→103→114→117→100
7	1	7119	4752	100→109→106→102→113→111→ 121 →115→105→ 100
	2	4289	7009	100→112→103→114→117→100
8	1	7119	4752	100→109→106→102→113→111→ 121 →115→105→ 100
	2	6062	5902	100→120→119→116→108→112→103→114→117→100
9	1	7119	4752	100→109→106→102→113→111→ 121 →115→105→ 100
	2	7096	4202	100→120→119→116→108→112→103→114→117→100
10	1	7741	4469	100→109→106→102→113→111→121→115→104→105→107→ 100
	2	7096	4202	100→120→119→116→108→112→103→114→117→100

TABLE IV
COMPUTING EFFICIENCY OF ACOP ALGORITHM

Scenario No.	Number of nodes (vehicle 1)	Number of nodes (vehicle 2)	Computing time (ms)
1	2	0	391
2	4	0	440
3	6	0	604
4	8	0	692
5	10	2	859
6	10	4	967
7	10	6	1093
8	10	8	1437
9	10	10	1405
10	12	12	1406

expense of slightly higher mileage. The authors also notice the influences of request ordering on solution being worth to exploring. Finally, although the algorithm was developed for the project "HubChain", it is intended to be evaluated in scenarios with more complex networks and higher demand.

REFERENCES

[1] J.-F. Cordeau and G. Laporte, "The dial-a-ride problem (darp): Models and algorithms," *Annals OR*, vol. 153, pp. 29–46, 06 2007.

[2] J.-F. Cordeau, "A branch-and-cut algorithm for the dial-a-ride problem," *Operations Research*, vol. 54, no. 3, pp. 573–586, 2006.

[3] S. C. Ho, W. Szeto, Y.-H. Kuo, J. M. Leung, M. Petering, and T. W. Tou, "A survey of dial-a-ride problems: Literature review and recent developments," *Transportation Research Part B: Methodological*, vol. 111, pp. 395–421, 2018.

[4] J.-F. Cordeau and G. Laporte, "A tabu search heuristic for the static multi-vehicle dial-a-ride problem," *Transportation Research Part B: Methodological*, vol. 37, no. 6, pp. 579–594, 2003.

[5] T. Gschwind and M. Drexl, "Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem," *Transportation Science*, vol. 53, no. 2, pp. 480–491, 2019.

[6] R. M. Jorgensen, J. Larsen, and K. B. Bergvinsdottir, "Solving the dial-a-ride problem using genetic algorithms," *Journal of the operational research society*, vol. 58, no. 10, pp. 1321–1331, 2007.

[7] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[8] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on evolutionary computation*, vol. 1, no. 1, pp. 53–66, 1997.

[9] J. E. Bell and P. R. McMullen, "Ant colony optimization techniques for the vehicle routing problem," *Advanced engineering informatics*, vol. 18, no. 1, pp. 41–48, 2004.

[10] S. Mazzeo and I. Loiseau, "An ant colony algorithm for the capacitated vehicle routing," *Electronic Notes in Discrete Mathematics*, vol. 18, pp. 181–186, 2004.

[11] B. Yu, Z.-Z. Yang, and B. Yao, "An improved ant colony optimization for vehicle routing problem," *European journal of operational research*, vol. 196, no. 1, pp. 171–176, 2009.

[12] S. Balseiro, I. Loiseau, and J. Ramonet, "An ant colony algorithm hybridized with insertion heuristics for the time dependent vehicle routing problem with time windows," *Computers & Operations Research*, vol. 38, no. 6, pp. 954–966, 2011.

[13] T. Tripathy, S. C. Nagavarapu, K. Azizian, R. R. Pandi, and J. Dauwels, "Solving dial-a-ride problems using multiple ant colony system with fleet size minimisation," in *UK Workshop on Computational Intelligence*. Springer, 2017, pp. 325–336.

[14] S. G. Ho, H. W. Koh, R. R. Pandi, S. C. Nagavarapu, and J. Dauwels, "Solving time-dependent dial-a-ride problem using greedy ant colony optimization," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 778–785.

[15] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. [Online]. Available: https://elib.dlr.de/124092/

[16] M. G. Armellini, "Optimierung der buslinie 450 in braunschweig durch on-demand-zubringer," Master's thesis, Fachhochschule Münster, 2019.