2018 21st International Conference on Intelligent Transportation Systems (ITSC)
Maui, Hawaii, USA, November 4-7, 2018

Solving Time-dependent Dial-a-ride Problem using Greedy Ant Colony Optimization

Song Guang Ho*, Hong Wei Koh†, Ramesh Ramasamy Pandi‡, Sarat Chandra Nagavarapu§ and Justin Dauwels¶
School of Electrical and Electronic Engineering
Nanyang Technological University, 50 Nanyang Avenue, Singapore, 639798.
Email: *hosongguang@gmail.com, †hkoh005@e.ntu.edu.sg, ‡ramesh006@e.ntu.edu.sg,
§saratchandra.nagavarapu@gmail.com, ¶jdauwels@ntu.edu.sg

***Abstract*—Traffic congestion is a huge problem for transportation service providers, which causes delay and incurs excessive operational cost. Many algorithms exist for the dial-a-ride problem (DARP), which assumes constant travel times throughout the day. This may cause serious constraint violations when they are applied in the real-world, where time-varying travel times are observed. This paper aims to solve time-dependent dial-a-ride problem (TDDARP). A new greedy ant colony optimization (GACO) algorithm is developed, in which two new decision factors (repeat counter and quantity counter) are introduced to efficiently explore the search space. In addition, a method for estimating travel time using staircase regression (SR) of speed data is developed, which in turn is incorporated into GACO. Computational experiments conducted on several DARP benchmark instances in the literature show the significance of incorporating time-dependent travel times into designing high quality solution for TDDARP, while considering peak hour traffic congestion. On average, GACO attains solutions with 88.77% less time window constraint violation and 98.04% less ride time constraint violation after taking time-dependent travel times into account.**

***Index Terms*—Dial-a-ride problem, time-dependent travel times, ant colony optimization.**

I. Introduction

Traffic is worsening in growing metropolitan regions around the world due to considerable growth in urban population and vehicles on road networks. Traffic congestion influences the travel time for road users to reach their destinations, which may incur an additional operational cost for chauffeur and delivery businesses, and even endanger people's lives in the case of emergency transportation services.

There are two major causes for on-road congestion: peak hour traffic and the occurrences of stochastic events. Peak hour traffic occurs every day around commuting hours when crowding of vehicles is at its highest; stochastic events are incidents that occur randomly from time-to-time, such as inclement weather, road work, vehicle breakdown, accident, etc. On average, delays caused by peak hour traffic congestion constitute 70 to 83% of the total traffic delay, whereas delays resulted from the occurrences of stochastic events only constitute 13 to 30% of the total traffic delay [1]. Therefore, in this paper, we focus on the major contributor of this problem, the peak hour traffic congestion.

Vehicle routing problem (VRP) was introduced by Dantzig [2] to model an application of goods delivery service. It is considered as one of the most important combinatorial optimization problems, due to its practicality in the real-world [3]. The dial-a-ride problem (DARP), a sophisticated version of VRP, deals with door-to-door transportation of people while satisfying some predefined constraints to ensure high service quality. In recent times, many algorithms [4] [5] [6] [7] have been developed to solve DARP. However, these existing algorithms assume constant speed throughout the network.

In the literature, time-dependent travel times have been considered while solving vehicle routing problems (VRP). Malandraki and Daskin [8] were the first to introduce the effect of dynamic traffic conditions in VRP and TSP. Moreover, time-dependent VRP with soft time window constraints was solved in [9]. A more complicated version of VRP with a speed distribution model was developed in [10], where multiple ant colony systems were employed to solve the problem.

In the past few years, dial-a-ride problems (DARP) bearing complex constraints were studied in time-dependent network. Xiang presented a scheduling heuristic [11] to tackle stochastic events in a time-dependent dial-a-ride problem (TDDARP). More recently, an exact algorithm based on branch-and-price [12] was proposed for solving TDDARP and tested on a sub road network of Kaohsuing City, Taiwan. Schilde identified that exploiting historical accident information for dynamic DARP with stochastic and time-dependent travel speeds significantly improve the solution quality [13]. Clearly, the variants of DARP gained less attention when compared to other vehicle routing problems.

Traffic congestion is almost certain to worsen in the next few decades due to rising population and wealth [14]. Therefore, it is important to consider time-dependent traffic conditions when performing routing and scheduling for DARP, to make the problem more suitable and applicable in real life scenarios.

In this paper, we propose a new greedy ant colony optimization (GACO) algorithm to solve TDDARP. The contributions of this paper can be summarized as follows:

- A greedy ant colony optimization (GACO) algorithm, with two new decision factors (repeat counter and quantity counter) to efficiently explore the search space.
- A method for travel time estimation using staircase regression (SR).
- Quantitative analysis on the impact of incorporating travel time estimation into GACO to solve TDDARP.

The remainder of this paper is organized as follows. In Section II, we discuss the formulation of TDDARP. In Section III, we introduce the new greedy ant colony optimization (GACO) algorithm. In Section IV, we present a travel time estimation procedure using staircase regression. In Section V, we analyze the performance of the proposed GACO. In Section VI, we conclude the paper and recommend a few possible directions for future research.

II. TDDARP Formulation

Time-dependent dial-a-ride problem (TDDARP) addresses the transportation of people between specified pickup and drop-off locations. TDDARP is solved as a combinatorial optimization problem with the aim of minimizing the total travel time while ensuring no constraint violation. Standard DARP formulation proposed by Cordeau and Laporte [4] is adopted in this paper. Our TDDARP formulation is built on this standard DARP formulation by incorporating time-dependent travel times for vehicles. While standard DARP formulation assumes constant speed throughout the network, our TDDARP formulation assumes all vehicles in the network travel with time varying speed described by a continuous speed model $y(t)$.

A fleet of m vehicles is used to serve n customer requests, where each request i is associated with a passenger-specified time window for either pickup or drop-off. Each vehicle k starts and ends at the same depot. Each passenger requires 10 min service time at both pickup and drop-off vertices. Furthermore, TDDARP is subjected to four key constraints: time window, ride time, load and duration. Following are the details of the occurrences of these constraint violations: a vehicle k fails to meet the time constraint on pickup or drop-off results in the violation of time window constraint $w(x)$; number of customers in a vehicle k exceeding its load limit Q_k causes the violation of load constraint $q(x)$; a vehicle k exceeding its duration limit T_k leads to duration constraint violation $d(x)$; a customer transported for time greater than ride time limit L results in ride time constraint violation $t(x)$. The constraints are defined as follows:

$$q(x) = \sum_{\forall k} \max(q_{k,max} - Q_k, 0) \tag{1}$$

$$d(x) = \sum_{\forall k} \max(d_k - T_k, 0) \tag{2}$$

$$w(x) = \sum_{\forall i} \big[\max(B_i - l_i, 0) + \max(B_{i+n} - l_{i+n}, 0) \big] \tag{3}$$

$$t(x) = \sum_{\forall i} \max(L_i - L, 0). \tag{4}$$

III. Greedy Ant Colony Optimization (GACO)

Greedy ant colony optimization (GACO) is modified from ant colony optimization (ACO) [15], which is inspired by the way ants locate their food source; they react to stimulation based on the performance they achieve. In this paper, four types of natural indirect non-symbolic communication between ants are mimicked: (i) after locating a food source, ants excrete pheromone on their path back to the nest; (ii) while finding food, each ant chooses a search path with more pheromone; (iii) in a place without pheromone, ants move around and explore the area to search for food, without sticking together; (iv) ants can adapt their behaviour to the circumstances, i.e. ants are more attracted to a food source (e.g. carbohydrate, protein, water, etc.) if the particular food source is in low volume (i.e., in high demand).

In this section, we present the new greedy ant colony optimization (GACO) algorithm for TDDARP. There is a major conceptual difference between GACO and other existing ACO algorithms: ants in GACO select a path with the highest decision score, whereas ants in other ACO algorithms select a path based on a probability distribution. Although ants in the real-world behave more randomly than deterministically, our preliminary tests show that GACO is faster in obtaining a reasonably good solution than ACO, despite having a higher probability of getting stuck at local minima.

A. Algorithm

Greedy ant colony optimization algorithm operates within a specific search space $\mathbb{S}$, where all solutions must be feasible without the need to be complete. A solution x is in feasible space $\mathbb{F}$ when all constraints ($q(x)$, $d(x)$, $w(x)$, and $t(x)$) are not violated; a solution x is in complete space $\mathbb{C}$ when all requests are served. Feasible space $\mathbb{F}$, complete space $\mathbb{C}$, and search space $\mathbb{S}$ are defined as follows:

$$\mathbb{F} = \{x : q(x) = d(x) = w(x) = t(x) = 0\} \tag{5}$$

$$\mathbb{C} = \{x : r(x) = n\} \tag{6}$$

$$\mathbb{S} = \{x : x \in \mathbb{F}\}, \tag{7}$$

where $r(x)$ is the number of requests served in a solution. A solution x has a cost $f(x)$ which is equivalent to the total travel time of all vehicles. All solutions are evaluated using three-level neighborhood evaluation method [4], which minimizes all constraint violations while performing scheduling. Here, we assume each vehicle departs immediately after completing a service.

Fundamentally, greedy ant colony optimization is a repetitive process of two basic actions: ant solution construction (i.e., path finding) and program parameters tuning. The optimization process is performed for T total time counts; during each time count t, *nants* number of ants are tasked to construct solution and tune the program parameters accordingly. A flowchart of the algorithm is shown in Fig. 1, and a detailed implementation of GACO is explained in Algorithm 1.

Ant solution construction is influenced by four decision factors: pheromone *Tau*(t), heuristic information *Eta*, repeat counter $R(t)$, and quantity counter $Q(t)$. Pheromone *Tau*(t) contains independent pheromone levels τ_{ij} of each edge (i, j) at a given time count t; heuristic information *Eta* contains the information about the relative possibility η_{ij} of a certain edge (i, j) contributing to higher quality solutions; repeat counter $R(t)$ records the number of occurrences r_{ij} that a certain edge (i, j) has been visited at a given time count t; quantity counter

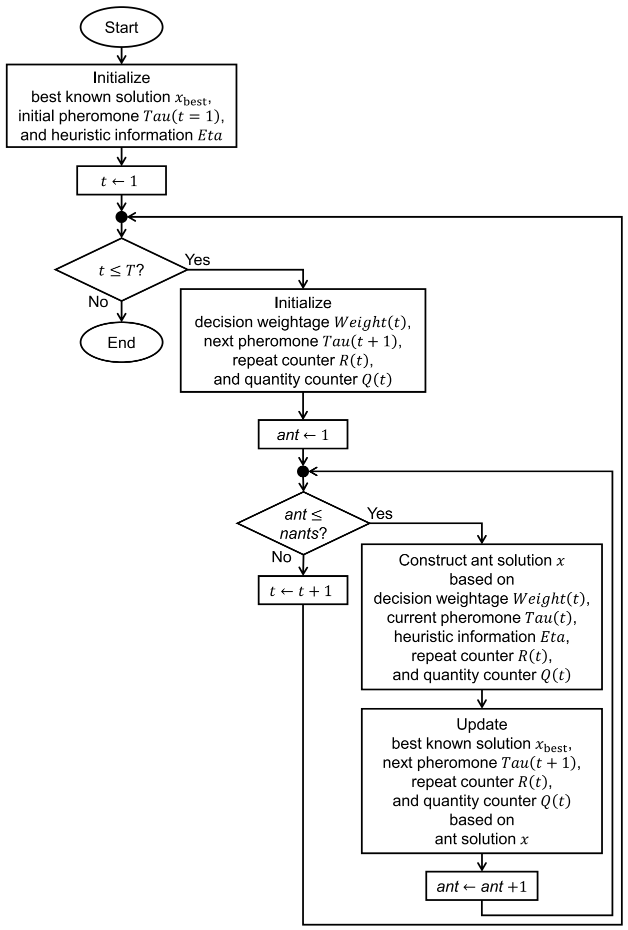
Start
Initialize best known solution x_{best}, initial pheromone $Tau(t=1)$, and heuristic information Eta
$t \leftarrow 1$
$t \leq T$?
Yes
No
End
Initialize decision weightage $Weight(t)$, next pheromone $Tau(t+1)$, repeat counter $R(t)$, and quantity counter $Q(t)$
$ant \leftarrow 1$
$ant \leq nants$?
Yes
No
$t \leftarrow t+1$
Construct ant solution x based on decision weightage $Weight(t)$, current pheromone $Tau(t)$, heuristic information Eta, repeat counter $R(t)$, and quantity counter $Q(t)$
Update best known solution x_{best}, next pheromone $Tau(t+1)$, repeat counter $R(t)$, and quantity counter $Q(t)$ based on ant solution x
$ant \leftarrow ant + 1$

Fig. 1. Greedy ant colony optimization (GACO).

Algorithm 1 Greedy ant colony optimization (GACO)

```
1:  procedure ACO($T, nants, \rho$)
2:      $x_{\text{best}} \leftarrow \varnothing$, where $f(x_{\text{best}}) \to \infty$.
3:      $Tau(t=1) \leftarrow \{0, 0, \ldots, 0\}$.
4:      Initialize $Eta$.                                      ▷ Eq. 13
5:      for $t \leftarrow 1 : T$ do
6:          Initialize $Weight(t)$.                            ▷ Eq. 25
7:          $Tau(t+1) \leftarrow (1-\rho) \times Tau(t)$.
8:          $R(t) \leftarrow \{0, 0, \ldots, 0\}$.
9:          $Q(t) \leftarrow \{1, 1, \ldots, 1\}$.
10:         for $ant \leftarrow 1 : nants$ do
11:             $P \leftarrow Q(t)/ant$.
12:             $x \leftarrow$ GETX($Weight(t), Tau(t), Eta, R(t), P$).
13:             if $x \in \mathbb{C}$ and $f(x) < f(x_{\text{best}})$ then
14:                 $x_{\text{best}} \leftarrow x$.
15:             end if
16:             $Tau(t+1) \leftarrow Tau(t+1) + \Delta Tau(x)$.  ▷ Eq. 22
17:             $R(t) \leftarrow R(t) + \Delta R(x)$.             ▷ Eq. 23
18:             $Q(t) \leftarrow Q(t) + \Delta Q(x)$.             ▷ Eq. 24
19:         end for
20:     end for
21:     return $x_{\text{best}}$.
22: end procedure
```

$Q(t)$ records the number of occurrences q_j that a certain node j has been visited at a given time count t. The four decision factors are described by the equations below:

$$Eta = \{\eta_{ij} : i \in \mathbb{V}, j \in \mathbb{V}, i \neq j\} \tag{8}$$

$$Tau(t) = \{\tau_{ij} : i \in \mathbb{V}, j \in \mathbb{V}, i \neq j\}, \qquad 1 \leq t \leq T \tag{9}$$

$$R(t) = \{r_{ij} : i \in \mathbb{V}, j \in \mathbb{V}, i \neq j\}, \qquad 1 \leq t \leq T \tag{10}$$

$$Q(t) = \{q_j : j \in \mathbb{V}\}, \qquad 1 \leq t \leq T. \tag{11}$$

All of the decision factors are dependent on time count except heuristic information *Eta*. Therefore, heuristic information *Eta* is computed once at the start of the program, whereas other decision factors are updated periodically throughout the execution of the program. Decision weightage $Weight(t) = \{\alpha, \beta, \gamma, \delta\}$ defines the weightage of the four decision factors when selecting a path.

B. Ant Solution Construction

Each ant is responsible for constructing a solution, which is described as follows: (i) create empty routes, (ii) select the next move, (iii) perform the selected move if no constraint is violated, (iv) repeat step ii and iii until no feasible move is available. Here, a move M_j^k is defined as moving a vehicle k from its current location i to next location j.

The next move is selected from a set of allowed moves $\mathbb{A}$, which can be categorized into two types: (i) moving a vehicle k to an unvisited pickup vertex, and (ii) moving a vehicle k to complete a drop-off. An example of allowed moves $\mathbb{A}$ is illustrated in Fig. 2.

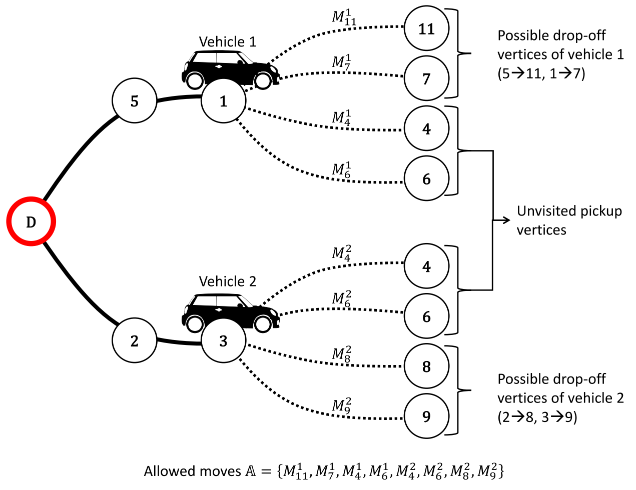
Vehicle 1
Vehicle 2
D
5
1
2
3
M_{11}^1
M_7^1
M_4^1
M_6^1
M_4^2
M_6^2
M_8^2
M_9^2
11
7
4
6
4
6
8
9
Possible drop-off vertices of vehicle 1 (5→11, 1→7)
Unvisited pickup vertices
Possible drop-off vertices of vehicle 2 (2→8, 3→9)
Allowed moves $\mathbb{A} = \{M_{11}^1, M_7^1, M_4^1, M_6^1, M_4^2, M_6^2, M_8^2, M_9^2\}$

Fig. 2. An example of allowed moves $\mathbb{A}$.

Each ant designs routes for all vehicles, and only one move with the highest decision score is selected at a time; in other words, each ant moves a vehicle at a time, one after another, until a solution is formed. Since each selected move is only performed if it satisfies all constraints, the final solution formed by each ant is always feasible, but not necessarily complete.

Selection of the next move is greedy; the move M_j^k with the highest decision score $S(i,j)$ is selected. Decision score

$S(i, j)$ of the action of moving a vehicle k from vertex i to j is defined as follows:

$$S(i,j) = \frac{\tau_{ij}^{\alpha} \times \eta_{ij}^{\beta}}{p_j^{\gamma} \times r_{ij}^{\delta}}, \tag{12}$$

Algorithm 2 Construct ant solution x

```
1: procedure GETX($Weight(t), Tau(t), Eta, R(t), P$)
2:     $x \leftarrow$ empty routes.
3:     $\mathbb{A} \leftarrow \{M_j^k : k \in \mathbb{K}, j \in \mathbb{P}\}$.
4:     repeat
5:         $x^+_{\text{highest}} \leftarrow \varnothing$.
6:         $S_{\text{highest}} \leftarrow -\infty$.
7:         $[k_{\text{highest}}, j_{\text{highest}}] \leftarrow \varnothing$.
8:         for all $k \in \mathbb{K}$ do
9:             $i \leftarrow$ current location of vehicle $k$.
10:            for all $j \in \mathbb{V}$ do
11:                if $M_j^k \in \mathbb{A}$ then
12:                    if $S(i,j) > S_{\text{highest}}$ then        ▷ Eq. 12
13:                        if $j \in \mathbb{P}$ then
14:                            $x^+ \leftarrow \begin{cases}\text{insert vertices } j\\ \text{and } j+n \text{ into}\\ \text{vehicle } k\\ \text{in solution } x.\end{cases}$
15:                        else if $j \in \mathbb{D}$ then
16:                            $x^+ \leftarrow \begin{cases}\text{insert vertex } j\\ \text{into vehicle } k\\ \text{in solution } x.\end{cases}$
17:                        end if
18:                        if $x^+$ is feasible then
19:                            $x^+_{\text{highest}} \leftarrow x^+$.
20:                            $S_{\text{highest}} \leftarrow S(i,j)$.
21:                            $[k_{\text{highest}}, j_{\text{highest}}] \leftarrow [k, j]$.
22:                        else if $j \in \mathbb{D}$ then
23:                            $x \leftarrow \begin{cases}\text{remove vertex } j-n\\ \text{from solution } x.\end{cases}$
24:                            $\mathbb{A} \leftarrow \mathbb{A} \cup \{M_j^{k'} : k' \in \mathbb{K}, k' \neq k\}$.
25:                        end if
26:                    end if
27:                end if
28:            end for
29:        end for
30:        $x \leftarrow x^+_{\text{highest}}$.
31:        $\mathbb{A} \leftarrow \mathbb{A} \cap \overline{\{M_j^k : k \in \mathbb{K}\}}$.
32:        if $j_{\text{highest}} \in \mathbb{P}$ then
33:            $x \leftarrow \begin{cases}\text{remove vertex } j+n\\ \text{from solution } x.\end{cases}$
34:            $\mathbb{A} \leftarrow \mathbb{A} \cup \{M^{k_{\text{highest}}}_{j_{\text{highest}}+n}\}$.
35:        end if
36:    until no feasible move is available
37:    return $x$.
38: end procedure
```

where $P(t) = \{p_j : j \in \mathbb{V}\}$ is the percentage of succeeded attempts by previous ants to visit a certain vertex j within a given time count t. The value of p_j is calculated as q_j/(*number of attempts made*). An ant is more likely to choose a path if there is higher pheromone or heuristic information; an ant has a higher chance of choosing a path if the path is rarely visited or a node is rarely explored.

The selected move is performed after ensuring that it does not inflict any constraint violation. If the selected move is a path towards a pickup vertex i, both pickup vertex i and drop-off vertex $i + n$ are inserted into the current solution x to create the next solution x^+. Then, if x^+ is a feasible solution, the current solution x is replaced by the next solution x^+. On the other hand, if the selected move is a path towards a drop-off vertex $i + n$, the drop-off vertex $i + n$ is inserted into the current solution x to create the next solution x^+. Then, if the next solution x^+ is feasible, it replaces the current solution x. Otherwise, both pickup vertex i and drop-off vertex $i + n$ are removed from the current vehicle, so that other vehicles have a chance to fulfil the request. Algorithm 2 explains the procedure of constructing an ant solution x.

C. Heuristic Information

Heuristic information is important to equip each ant with a good instinct of direction and the ability to judge a path. For vehicle routing problem (VRP), the conventional method of computing heuristic information is by defining heuristic information η_{ij} of an edge (i, j) as 1/(distance δ_{ij}), so that a shorter path is always preferred. However, DARP is highly complicated, such that travel cost should be minimized while fulfilling constraints that do not exist in VRP. Therefore, we propose a new method of computing heuristic information η_{ij} of an edge (i, j), which is defined as follows:

$$\eta_{ij} = \frac{(pos_{ij})^{k_1}}{\exp(k_2 \times mwt_{ij} + k_3 \times \delta_{ij})}, \tag{13}$$

where k_1, k_2, and k_3 are constants, mwt_{ij} is the minimum waiting time when vertex j is served immediately after vertex i, and pos_{ij} is the possibility of vertex j being served after vertex i. Before computing mwt_{ij} and pos_{ij}, we perform time window adjustment [16] to obtain earliest service time e_i and latest service time l_i for each vertex i. Then, pos_{ij} and mwt_{ij} to travel from vertex i to vertex j are computed as follows:

$$mwt_{ij} = e_j - l_i - d_i \tag{14}$$

$$pos_{ij} = \frac{\int_{e_i}^{l_i} \int_{t_i+d_i}^{\infty} u(t_j - e_j) - u(t_j - l_j)\, \mathrm{d}t_j\, \mathrm{d}t_i}{(l_i - e_i) \times (l_j - e_j)}, \tag{15}$$

where d_i is the service duration at vertex i, and $u(t_j)$ is the unit step function. In this paper, the values of k_1, k_2, and k_3 are set as $\frac{3}{10}$, $\frac{1}{12}$, and 1, respectively. These values are tuned based on our preliminary assessment by changing one-factor-at-a-time (OAT) approach.

D. Update Rules

The most critical component of any ACO algorithms is the ants' ability to react to their environment. Three program parameters are updated after constructing an ant solution, which are pheromone *Tau*$(t+1)$, repeat counter $R(t)$ and quantity counter $Q(t)$. As mentioned in subsection III-A, these parameters are also decision factors that influence the future decision of each ant. The update rules are described as follows:

$$\mathit{Tau}(t+1) \leftarrow \mathit{Tau}(t+1) + \Delta \mathit{Tau}(x) \tag{16}$$

$$R(t) \leftarrow R(t) + \Delta R(x) \tag{17}$$

$$Q(t) \leftarrow Q(t) + \Delta Q(x), \tag{18}$$

where $\Delta\mathit{Tau}(x)$, $\Delta R(x)$, and $\Delta Q(x)$ are values added onto their respective program parameters based on the ant solution x obtained. These added values are defined as follows:

$$\Delta\mathit{Tau}(x) = \{\Delta\tau_{ij}(x) : i \in \mathbb{V}, j \in \mathbb{V}, i \neq j\} \tag{19}$$

$$\Delta R(x) = \{\Delta r_{ij}(x) : i \in \mathbb{V}, j \in \mathbb{V}, i \neq j\} \tag{20}$$

$$\Delta Q(x) = \{\Delta q_j(x) : j \in \mathbb{V}\} \tag{21}$$

$$\Delta\tau_{ij}(x) = \begin{cases} \exp(1)\left[\dfrac{f(x_{\text{best},0})}{f(x)}\right]^{25} \displaystyle\sum_{\forall(i,j)\text{ in }x} p_{ij} & \text{if } x \in \mathbb{C} \\ \exp((\text{perc}(x))^2) \displaystyle\sum_{\forall(i,j)\text{ in }x} p_{ij} & \text{otherwise} \end{cases} \tag{22}$$

$$\Delta r_{ij}(x) = \begin{cases} 1 & \text{if edge } (i,j) \text{ exist in solution } x \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

$$\Delta q_j(x) = \begin{cases} 1 & \text{if node } j \text{ exist in solution } x \\ 0 & \text{otherwise,} \end{cases} \tag{24}$$

where perc(x) is the percentage of requests served in solution x, and $f(x_{\text{best},0})$ is the cost of the first feasible solution obtained in GACO. The pheromone $\Delta\tau_{ij}(x)$ excreted by each ant is influenced by the achievement of that ant. If an ant found an incomplete solution, its achievement is judged according to the percentage of requests *perc*(x) it serves; whereas if an ant found a complete solution, its achievement is judged by the travel cost $f(x)$ it obtains.

E. Decision Weightage

The tuning of decision weightage *Weight*(t) can significantly influence the convergence of GACO. We have incorporated the concept of '*knowledge first, experience later*', by adjusting the decision weightage of heuristic information β. At the start of the program, GACO is designed to heavily rely on heuristic information to explore the search space. As the optimization process continues, the decision making to select the next path relies less on heuristic information, and more on pheromone level. Towards the end of the program, GACO relies solely on pheromone level to exploit the paths with higher pheromone levels. We tune the decision weightage according to the equation below:

$$\beta(t) = \beta_0^{\frac{t}{T}}, \tag{25}$$

where β_0 is set as 0.02 throughout the paper. The other decision weightages α, γ and δ are constants set as 1, 100, and 1, respectively. The values of these decision weightages are tuned using changing one-factor-at-a-time (OAT) approach.

F. Summary

To locate food source, ants rely on a unique way of communication, which is indirect and non-symbolic. Each ant chooses a path according to essential information it obtains from the environment through its senses; each ant also influences the environment through its presence at a certain location, or through the pheromone it excretes.

	Ant Behaviours	GACO Implementation
(i)	Ants excrete pheromone on their path back to the nest after locating a food source	Pheromone $\Delta\tau_{ij}(x)$ is dissipated on a path (i,j) after each ant construct a solution x
(ii)	Ants choose a path with more pheromone while searching for food	Ants select the path with the highest decision score $S(i,j)$; the higher the pheromone τ_{ij}, the higher the value of the decision score $S(i,j)$
(iii)	Ants spread out while searching for food to explore a wider area	Repeat counter r_{ij} records the number of times a path (i,j) was travelled by other ants; ants are less likely to choose a path with a higher repeat counter r_{ij}
(iv)	Ants adapt to the circumstances; they search intensively for a food source if that particular food source is in low quantity	Quantity counter q_j records the number of times a location j was visited by other ants; ants are more likely to choose a path that leads to a location j with a lower quantity counter q_j

TABLE I
GACO IMPLEMENTATIONS OF ANT BEHAVIOURS.

We learned how ants behave, and tried our best to mimic and recreate their behaviours in the development of GACO. Table I summarizes how the ant behaviours are mimicked in GACO. In the next section, we explain how to incorporate travel time estimation into GACO.

IV. Travel Time Estimation

To solve time-dependent dial-a-ride problem (TDDARP), time-dependent travel times must be accounted for. In this section, we propose a method of estimating travel time to be incorporated into GACO. Using discrete speed data, we perform staircase regression (SR) to create a speed regression model. Travel time is calculated by executing integration on the speed regression model.

A. Staircase Regression (SR)

Staircase regression is performed to regress discrete speed data into a linear piecewise function. The planning horizon is divided into V number of time segments of equal interval size I. Data points within a particular time segment v are averaged to obtain the mean vehicle speed of that time segment. The resultant speed regression model is a combination of mean vehicle speeds from all time segments.

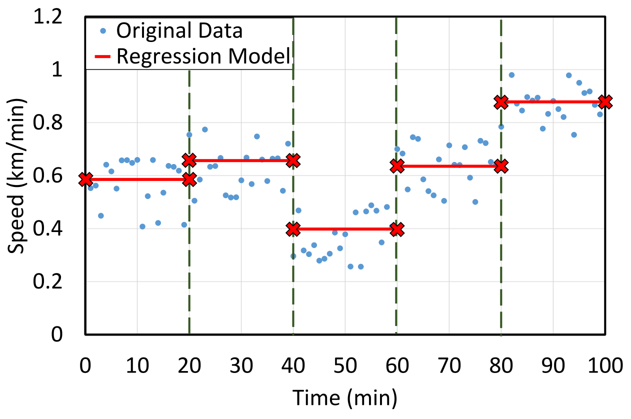

Fig. 3. Staircase regression (SR).

The staircase regression model $Z(t)$ can be formulated as follows:

$$Z(t) = \sum_{v=1}^{V} z_v(t) = z_1(t) + z_2(t) + \cdots + z_V(t) \tag{26}$$

$$z_v(t) = \begin{cases} b_v & \text{if } I(v-1) \le t < Iv \\ 0 & \text{otherwise,} \end{cases} \tag{27}$$

where b_v is the mean speed of a segment v.

Figure 3 shows an example of a discrete speed data within a planning horizon of 100 min. The red lines represent the staircase regression model $Z(t)$.

B. Travel Time Calculation by Speed Regression Model

The area under the speed regression model bounded by departure time D_i from vertex i and arrival time A_j at vertex j is the distance δ_{ij} between the vertices i and j. Assuming that departure time D_i and distance δ_{ij} between two vertices are known, arrival time A_j can be calculated by solving for the upper limit of the definite integral bounded by D_i and A_j.

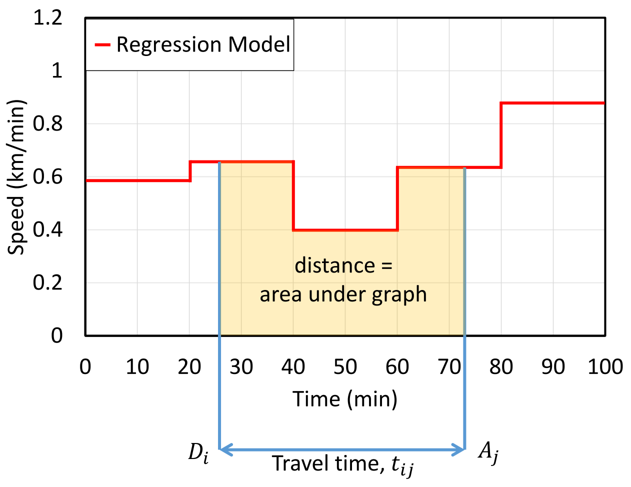

Fig. 4. Travel time calculation using speed regression model.

The distance δ_{ij} and travel time t_{ij} from vertex i to vertex j are described as follows:

$$\delta_{ij} = \int_{D_i}^{A_j} Z(t)\ \mathrm{d}t \tag{28}$$

$$t_{ij} = A_j - D_i. \tag{29}$$

From Fig. 4, the distance δ_{ij} between vertices i and j is equal to the area under the speed regression model $Z(t)$. As the speed regression model $Z(t)$ is a combination of constant functions, the area under the speed regression model $Z(t)$ are made up of several rectangles. Hence, integration of the speed regression model $Z(t)$ can be done in a simpler manner by summing the areas of different rectangles.

V. Simulation Results

The proposed greedy ant colony optimization (GACO) algorithm for TDDARP is implemented in C++. Numerical experiments have been conducted using a computer running 3.6 GHz Intel Core i7 processor with 16 GB RAM. The proposed GACO algorithm has been simulated using the DARP benchmark instances R1a and R1b [4]. For both of these instances, the number of requests is 24 and the number of vehicles is 3. The instances have different demographical distribution of pickup and drop-off locations. The capacity of each vehicle is 6 with a maximum route duration of 480 min; the maximum ride time is 90 min.

The following assumptions have been made for our implementation of GACO algorithm for TDDARP:

- All vehicles in the network move with the same speed at a given time, as generated by a continuous speed model $y(t)$.
- The traffic conditions (i.e., speed of vehicles) are assumed to be perfect and known apriori.

To simulate TDDARP, we sample a continuous speed model $y(t)$ to generate discrete speed data $y[k]$, described as follows:

$$y(t) = 1 - 0.7\exp(-(t-360)^2/(2 \times (80)^2)) \tag{30}$$

$$y[k] = y(k \times T_s), \tag{31}$$

where T_s is the sampling period. Here, we use $T_s = 1$ min.

Fig. 5 illustrates the staircase regression (SR) of the discrete speed data $y[k]$ obtained over a discrete sampling interval of 30 min. The x-axis of the plot shows the time of the day expressed in min and the y-axis represents speed of the vehicle in km/min. The dotted blue line and the solid red line correspond to the discrete speed data $y[k]$ and the staircase regression model respectively.

Five independent simulations are done using each method on each instance; all reported results are the mean value over these simulations. We analysed the impact of incorporating travel time estimation into GACO by using six quantifiers: (i) difference in total travel time, (ii) error in travel time estimation, (iii) late departure from pickup vertex, (iv) late arrival at drop-off vertex, (v) time window constraint violation, and (vi) ride time constraint violation.

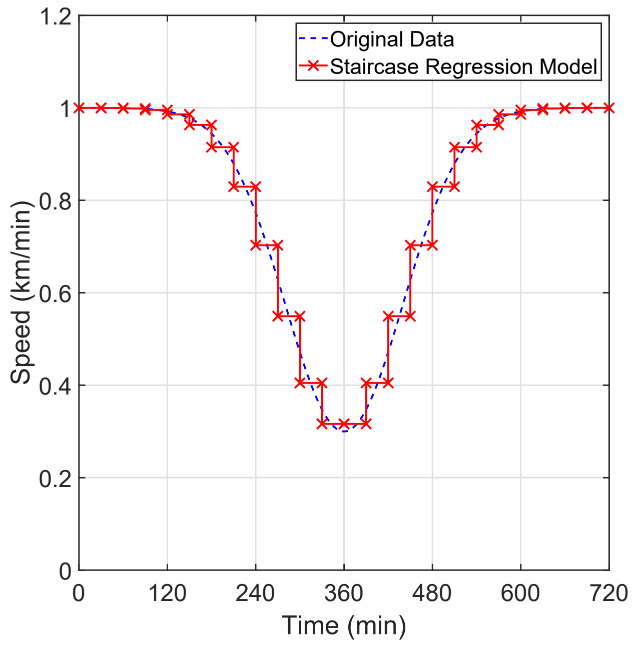

Fig. 5. Staircase regression of discrete speed data $y[k]$.

The estimated total travel time is the total travel time obtained for the planned routes; actual total travel time is the total travel time obtained from the simulated routes using the continuous speed model; difference in total travel time is the difference between these two travel time values; error in the travel time estimation is the sum of all absolute travel time errors computed for all the vehicle traversals; late departure from pickup vertex and late arrival at drop-off vertex are the amount of delays incurred at the respective vertices when vehicles arrive late when following a planned route; time window constraint violation indicates the total excess travel time incurred at the pickup and drop-off locations and ride time constraint violation represents the total excess ride time for all the passengers during routing.

	GACO	GACO + SR
Estimated total travel time (min)	209.50	358.85
Actual total travel time (min)	341.00	361.17
Difference in total travel time (min)	131.50	**4.14**
Error in travel time estimation (min)	131.50	**11.89**
Late departure from pickup vertex (min)	97.12	**15.95**
Late arrival at drop-off vertex (min)	193.45	**17.35**
Time window constraint violation (min)	66.18	**12.02**
Ride time constraint violation (min)	**0.00**	0.06

TABLE II
COMPARISON OF GACO AND GACO+SR ON R1A INSTANCE [4].

In Table II and III, we compare the quality of solutions obtained using GACO and GACO+SR on R1a and R1b benchmark instances. Here, GACO assumes constant vehicle speed of 1 km/min throughout the planning horizon, whereas GACO+SR assumes vehicle speed is time-dependent, and uses the staircase regression (SR) model $Z(t)$ to estimate the travel time between any two locations.

	GACO	GACO + SR
Estimated total travel time (min)	200.80	323.76
Actual total travel time (min)	347.73	334.16
Difference in total travel time (min)	146.93	**10.43**
Error in travel time estimation (min)	146.93	**22.41**
Late departure from pickup vertex (min)	151.82	**63.23**
Late arrival at drop-off vertex (min)	305.12	**65.14**
Time window constraint violation (min)	84.99	**4.95**
Ride time constraint violation (min)	3.16	**0.00**

TABLE III
COMPARISON OF GACO AND GACO+SR ON R1B INSTANCE [4].

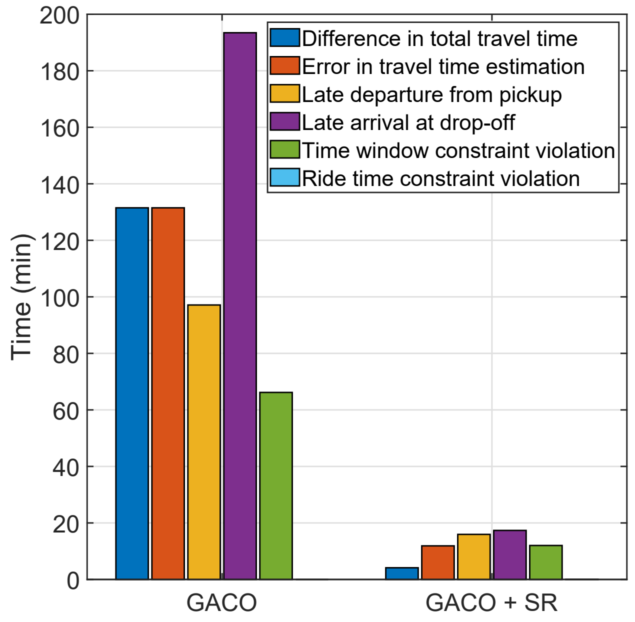

Fig. 6. Comparison of GACO and GACO+SR on R1a instance [4].

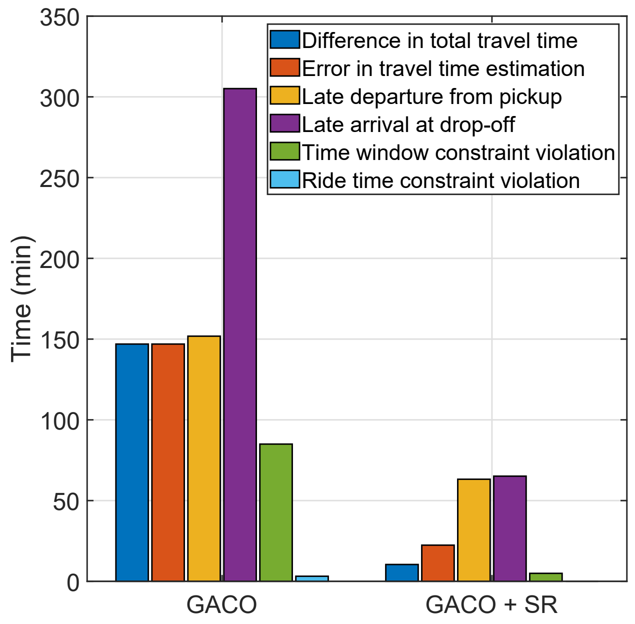

Fig. 7. Comparison of GACO and GACO+SR on R1b instance [4].

	Reduction (%)
Difference in total travel time	94.77
Error in travel time estimation	87.68
Late departure from pickup vertex	68.19
Late arrival at drop-off vertex	83.46
Time window constraint violation	88.77
Ride time constraint violation	98.04

TABLE IV
TOTAL REDUCTION IN PERFORMANCE QUANTIFIERS.

A lower magnitude of each quantifier indicates higher solution quality. Figs. 6 and 7 show that GACO+SR clearly outperforms GACO in designing routes and schedules in the event of peak hour traffic congestion. GACO+SR contributes to the reduction in the late arrival of the vehicles at the drop-off vertices remarkably. The six quantifiers obtained from R1a and R1b instances are summed and compared. The percentage of reduction in these quantifiers are shown in Table IV. It is evident that the inclusion of staircase regression model for travel time estimation for GACO significantly reduces the constraint violations and the other error values.

VI. CONCLUSION

In this paper, we have proposed a greedy ant colony optimization (GACO) to solve time-dependent dial-a-ride problem (TDDARP). Existing algorithms in the literature does not consider time-dependent travel times when solving dial-a-ride problem. This may cause serious delays and constraint violations when they are applied to solve real life problems. We developed a complete methodology for estimating travel time using staircase regression (SR) of speed data, and incorporated it into GACO to come up with GACO+SR.

This paper focuses on traffic congestion caused by peak hour traffic, which leads to delay and excessive operational cost for transportation service providers. We use an inverted Gaussian distribution function to model vehicle speed during peak hour. Computational experiments conducted using standard DARP instances [4] show that GACO+SR generates higher quality solutions for TDDARP than GACO.

Some possible directions for the future work include: (i) explore other regression methods, (ii) perform experiments with other traffic condition and on larger scale problems, and (iii) apply GACO to solve other more sophisticated versions of TDDARP (e.g. multi-depot, heterogeneous fleet and dynamic requests).

ACKNOWLEDGMENT

The research was partially supported by the ST Engineering-NTU Robotics Corporate Laboratory through the National Research Foundation (NRF) corporate lab@university scheme.

REFERENCES

[1] A. Skabardonis, P. Varaiya, and K. Petty, "Measuring recurrent and non-recurrent traffic congestion," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1856, pp. 118–124, 2003.

[2] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management science*, vol. 6, no. 1, pp. 80–91, 1959.

[3] P. Toth and D. Vigo, *The vehicle routing problem*. SIAM, 2002.

[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] S. N. Parragh, K. F. Doerner, and R. F. Hartl, "Variable neighborhood search for the dial-a-ride problem," *Computers & Operations Research*, vol. 37, no. 6, pp. 1129–1138, 2010.

[6] K. Braekers, A. Caris, and G. K. Janssens, "Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots," *Transportation Research Part B: Methodological*, vol. 67, pp. 166–186, 2014.

[7] S. G. Ho, R. R. Pandi, S. C. Nagavarapu, and J. Dauwels, "Multi-atomic annealing heuristic for the dial-a-ride problem," in *Proceedings of the 12th IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, 2018, pp. 272–277.

[8] C. Malandraki and M. S. Daskin, "Time dependent vehicle routing problems: formulations, properties and heuristic algorithms," *Transportation science*, vol. 26, no. 3, pp. 185–200, 1992.

[9] S. Ichoua, M. Gendreau, and J.-Y. Potvin, "Vehicle dispatching with time-dependent travel times," *European journal of operational research*, vol. 144, no. 2, pp. 379–396, 2003.

[10] A. V. Donati, R. Montemanni, N. Casagrande, A. E. Rizzoli, and L. M. Gambardella, "Time dependent vehicle routing problem with a multi ant colony system," *European journal of operational research*, vol. 185, no. 3, pp. 1174–1191, 2008.

[11] Z. Xiang, C. Chu, and H. Chen, "The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments," *European Journal of Operational Research*, vol. 185, no. 2, pp. 534–551, 2008.

[12] T.-Y. Hu and C.-P. Chang, "Exact algorithm for dial-a-ride problems with time-dependent travel cost," *Journal of the Eastern Asia Society for Transportation Studies*, vol. 10, pp. 916–933, 2013.

[13] M. Schilde, K. F. Doerner, and R. F. Hartl, "Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem," *European journal of operational research*, vol. 238, no. 1, pp. 18–30, 2014.

[14] A. Downs, "Traffic: Why it's getting worse, what government can do," Brookings Institution, Tech. Rep., 2004.

[15] M. Dorigo and M. Birattari, "Ant colony optimization," in *Encyclopedia of machine learning*. Springer, 2011, pp. 36–39.

[16] S. G. Ho, S. C. Nagavarapu, R. R. Pandi, and J. Dauwels, "Improved tabu search heuristics for static dial-a-ride problem: Faster and better convergence," *arXiv preprint arXiv:1801.09547*, 2018.