Australasian Transport Research Forum 2023 Proceedings
29 November – 1 December, Perth, Australia
Publication website: http://www.atrf.info

Delaunay-triangulation-based algorithm for demand-responsive transport services

Ronny Kutadinata[1], Lele Zhang[2], Aidan Hadwick-Guthrie[3]

[1] National Transport Research Organisation / Australian Road Research Board, Port Melbourne

[2] School of Mathematics & Statistics, The University of Melbourne, Parkville

[3] Mechatronics Engineering, Deakin University, Waurn Ponds

Email for correspondence (presenting author): ronny.kutadinata@arrb.com.au

Abstract

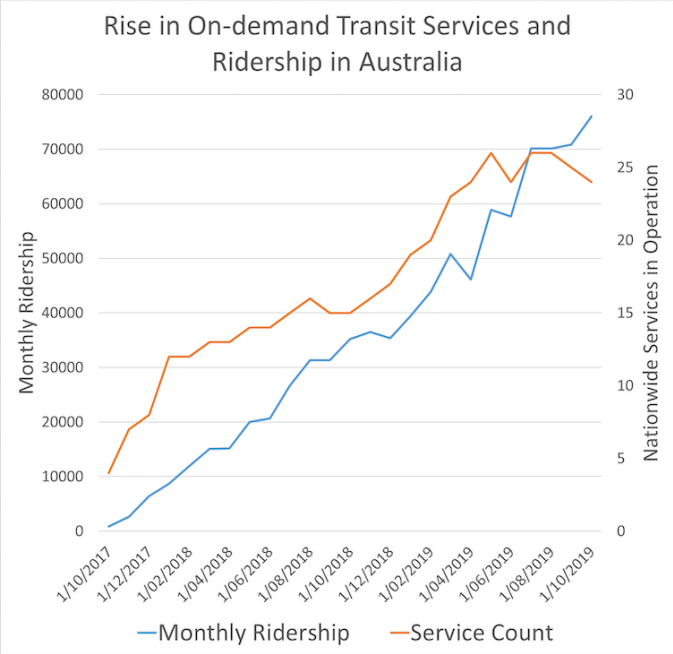
Demand-responsive service is an effective, emerging means to improve mobility levels in urban and rural areas while addressing issues such as traffic congestion, rising transportation costs, and environmental concerns. In this paper, we investigate a dial-a-ride problem (DARP), which deals with planning transit vehicle routes that satisfy passenger transportation requests while minimising the total cost and the customer inconvenience (due to early pick-up or late drop-off). We propose a Delaunay-triangulation-based algorithm, which decomposes the problem into two hierarchical subproblems and solves it iteratively. Numerical experiments on a real road network in Melbourne demonstrate that the proposed algorithm is capable to produce comparable solutions to the established neighbourhood search algorithm and with a significantly lower computational requirement.

1. Introduction

On-demand transit (a.k.a. demand-responsive transit) services have attracted increasing attention from both academia and industry, as a complementary transportation means to traditional scheduled bus/subway services. Passengers can book a trip, specifying the pick-up and drop-off locations and times. The service is more flexible as on-demand vehicles do not have fixed routes or timetables to follow and are scheduled based on requests. Compared to taxi services and driving private cars, on-demand transit services, as an important part of shared economy, specifically, shared mobility, provide great opportunities in addressing transportation challenges, such as rising traffic congestion, emissions, and transportation costs. They are particularly useful in accommodating mobility needs in rural areas, in regions where public transport is underutilised, and for the elderly and disabled.

The rapid development in technologies, such as autonomous vehicles, GPS, and the Internet of Things (IoT) has boosted the trial implementations of on-demand transit services in real life. In Australia, on-demand bus services have been put into trials in Sydney (Kaufman, 2020), Melbourne (PTV, 2023), and Gold Coast (Translink, 2023). In Wales, UK, passengers can "book a shuttle minibus from 'floating bus stops' near their homes directly to their destination" via an app (Kaufman, 2020). Moreover, autonomous buses have been deployed for on-demand services as part of the existing public transport networks in more than 5 countries in Europe (CORDIS, 2020). On-demand public transport has also been trialled in Japan and the US. Based on the data provided by the Queensland Department of Transport and Main Roads, there is a large rise in the on-demand transit services and ridership in Australia (see Figure 1).

Figure 1: Rise in on-demand transit services and ridership in Australia (Kaufman, 2020)



A Dial-a-Ride problem (DARP) is to address an important component of on-demand transportation systems, that is, investigating the planning and optimisation of ride arrangements. As the DARP is motivated from the real world, there are various features to consider, such as origins and destinations of passengers, time windows for pick-up/drop-off, vehicle capacity, fleet type, and passenger special requirements, which make it complex to model and solve. Current pilot projects of DAR are limited to trials in relatively small regions, for example, 11 locations in Wales (Laker, 2022), and several remote suburbs in Melbourne (FlexiRide, 2023). For large implementations, the algorithm for solving the DARP needs to be efficient and scalable in order to deal with a large number of requests and plan routes and stops in real time.

This study aims to develop an efficient algorithm based on Delaunay triangulation to solve the routing problem of a DAR service. Its performance is evaluated via extensive numerical experiments conducted on the Port Melbourne network in Melbourne. The results show that the proposed algorithm is able to achieve, on average, a 63.4% computation time saving while only experiencing a 0.88% decrease in the "optimality score".

This paper is organised as follows. Section 2 briefly reviews the literature on the dial-a-ride problems, the solution approaches, and the Delaunay-triangulation method. Section 3 states the studied problem and the mathematical model, while Section 4 explains the proposed algorithm. Numerical results are discussed in Section 5, followed by a short conclusion in Section 6.

2. Literature review

The first DAR service dates to the 1970s in the US. The emergence of shared mobility and the concern for the development of sustainable transportation systems have provided the impetus for the investigations into the DARPs. In a DARP, passengers (users) book their trips on an app, on the internet or via phone for transport from origins to destinations, which could be physical addresses, bus stops, etc. They may nominate time windows for their pick-up and/or drop-off, and preferences in terms of vehicle type and service type (shared or private). The service provider is responsible for arranging vehicles and planning routes to fulfil all requests. The DARPs have a lot of applications in various areas, including health care, large

transportation terminals and public transport. There are two recent reviews on the DARPs (Ho et al., 2018; Molenbruch et al, 2017a). Ho et al. (2018) reviewed 86 papers published between 2007 and 2017 and presented a detailed taxonomy of the variants of DARPs, based on the planning and scheduling procedures (*static or dynamic*) and uncertainty in the information received (*deterministic or stochastic*). The literature on the DARPs has been growing significantly since 2017, and the topic has attracted at least 200 publications, which further demonstrates the popularity of the DARP studies in the academic community.

There are many DARP variants. In terms of vehicle types, recent studies have investigated using electric vehicles (EVs) (Su et al, 2023), autonomous vehicles (Johnsen and Meisel, 2022; Liang et al., 2020), and heterogeneous fleet (Malheiros et al., 2021). The research of the DAR services has been extended to goods transportation, for example, container transportation (Kuźmicz et al., 2022). New features considered in the problems lead to extra constraints, such as driving ranges and minimum battery levels for EVs (Su et al., 2023), and objectives like environmental cost minimisation and assortment optimisation (Azadeh et al., 2022) in addition to classic profit maximisation (Liang et al., 2020), ride-time minimisation, and customer satisfaction maximisation (Johnsen and Meisel, 2022).

The solution approaches for the DARPs can be generally classified into two categories: exact methods, such as branch and bound (B&B) and its variants (Qu and Bard, 2015; Liu et al., 2015), and (meta-)heuristics, such as Tabu search (Detti et al., 2017), simulating annealing (Braekers et al., 2014), genetic algorithm (Cubillos et al., 2009), and various neighbourhood search algorithms (Su et al, 2023; Molenbruch et al., 2017b). Whilst the exact approaches are limited to small and medium instances, meta-heuristics and hybrid algorithms (Pimenta et al., 2017; Ritzinger et al., 2016) which combine multiple meta-heuristics, are more popular for large instances.

Since the DAR services receive real-time information and require real-time planning and scheduling, the computational efficiency of solution algorithms is crucial. For algorithms that are developed for solving the static, deterministic version of the DARP, that is, decisions are made once off with full knowledge, if they are sufficiently fast, they can be adapted for re-optimisation and be implemented “on-the-go”. Nevertheless, the meta-heuristics, like neighbourhood search algorithms, could be time-consuming, in particular when evaluating the neighbourhood. The current study is motivated by the goal of devising fast (on-line) algorithms.

One of the possible approaches to improve the computation time of DARP algorithms is to reduce the search space by introducing some “filters” or “guidance” scheme. In this study, the focus is to use Delaunay Triangulation (DT) for this purpose. Given a set of nodes, the Delaunay triangles are formed such that the given nodes do not lie within the circumcircle of any of the Delaunay triangles. A Delaunay triangulation is the dual graph of a corresponding Voronoi diagram, where it is formed by partitioning a plane into regions around the given nodes (Delaunay, 1934). As such, the DT edges connect adjacent nodes to each other and can provide useful information to determine the next “closest” stops when being applied to DARPs.

Over the years, DT has been used as the basis of routing algorithms, where the paths are selected to lie on the DT edges (Beasley and Christofides, 1997). Originally, DT is applied to help solve the travelling salesman problem (TSP) (Krasnogor et al., 1995; Lau and Shue, 2001). Although it has been proven that the solution to TSP does not always fully lies on the DT edges (Kantabura, 1983), Krasnogor et al. (1995) pointed out that there is a large portion of the solution that agrees with the DT edges. More recently, DT has been used on multi-depot delivery problems (Tu et al., 2014; Sazonov et al., 2018). In terms of the algorithms, DT is typically used as part of a Tabu neighbourhood search to restrict the search space of the neighbourhood operations (Krasnogor et al., 1995; Lau and Shue, 2001). However, there are

other ways to incorporate the DT into routing algorithms, such as the approach by Tu et al. (2014) (bi-level DT with simulated annealing) and Sazanov et al. (2018) (DT as the communication graph among multi-agents, each performing greedy insertion algorithm).

Therefore, the review has established the need to develop sufficiently fast DARP algorithms for real-life implementation. While DT is a promising approach, previous works have not addressed the method to handle the sequencing requirements of a pick-up and the corresponding drop-off. In light of this, this paper proposes a novel DT-based algorithm that considers the pick-up/drop-off order, as well as the time window by extending the DT into 3D space. Furthermore, the proposed algorithm also eliminates part of the search space by utilising a "non-search" heuristic that results in a significant reduction of the computation time.

3. Problem formulation

3.1. Problem statement

This study considers a fleet (heterogeneous or homogeneous) of capacitated vehicles to service customers, who nominate pick-up and drop-off locations and time windows. Let $\mathcal{P}$ denote the set of customers, that is, $\mathcal{P} = \{1,2,\dots,n\}$. Each customer $i \in \mathcal{P}$ specifies a time window $[E_i, L_i]$ and the customer is ready to be picked up at time E_i and should be dropped off by time L_i. Here, the customer may refer to a group of passengers, and the number of passengers in this customer group is represented by Q_i. The fleet of vehicles is denoted by $\mathcal{K}$, and each vehicle $k \in \mathcal{K}$ has capacity C_k in terms of the maximum number of passengers in the vehicle.

The transport network can be described by a graph $G = (\mathcal{V}, \mathcal{A})$ with a node set $\mathcal{V}$ and an arc set $\mathcal{A} = \{V_iV_j : V_i, V_j \in \mathcal{V}, i \neq j\}$. The set $\mathcal{V}$ contains $2n + 1$ nodes, node V_0 is the depot of vehicles, and nodes V_i and V_{i+n}, respectively, represent the pick-up and drop-off locations of customer $i \in \mathcal{P}$. For simplicity, we shall write node $i \in \mathcal{V}$ rather than $V_i \in \mathcal{V}$.

Following the classification by Ho et al. (2018), the studied problem is a capacitated, static, and deterministic DARP with a heterogeneous fleet, time windows and a single objective.

3.2. Model

The DARP in this study has two classes of decisions to make, the assignment of customers to vehicles and the route design of each vehicle. Our model defines a binary decision variable $x_{i,j}^k$ to represent if vehicle $k \in \mathcal{K}$ traverses the path from node $i \in \mathcal{V}$ to $j \in \mathcal{V}$. If for $i \leq n$, $x_{i,j}^k = 1$, then node j is assigned to be serviced by vehicle k.

We consider a soft constraint for the service time window $[E_i, L_i]$ for $i \in \mathcal{P}$. Consider that customer i is serviced by vehicle k. If the vehicle arrives at the origin node i earlier than the earliest pick-up time E_i, the vehicle will wait at the node until time E_i and then load the passenger(s). If the vehicle reaches the destination node $n + i$ later than the latest drop-off time L_i, it incurs a penalty, which will be included in the objective function to be minimised. Additionally, the pick-up wait time is limited to δ, beyond which will incur a penalty. For ease of presentation, we extend some attributes of customers to nodes. Specifically, for node $i = 1,2,\dots,2n$, the time window is given by $[e_i, l_i]$ with

$$e_i = \begin{cases} E_i & \text{if } i \leq n, \\ E_{i-n} + s_{i-n} + t_{i,i+n} & \text{otherwise,} \end{cases} \qquad l_i = \begin{cases} E_i + \delta & \text{if } i \leq n, \\ L_{i-n} & \text{otherwise,} \end{cases} \tag{1}$$

where s_i denotes the dwell time (for loading and unloading passengers) and $t_{i,j}$ is the travel time between nodes i and j. Moreover, the number of loading/unloading passengers is

$$q_i = \begin{cases} Q_i & \text{if } i \le n, \\ -Q_i & \text{otherwise.} \end{cases} \tag{2}$$

At the depot node, we have $e_0 = 0$, $l_0 = \infty$ and $q_0 = 0$.

Let u_j^k denote the *effective arrival time* of vehicle k at node $j \in \mathcal{V}$. This variable is constrained by the effective arrival time at the previous node (say $i \in \mathcal{V}$), the dwell time (for loading/unloading passengers) at the previous node, denoted by parameter s_i, the travel time between nodes i and j, denoted by parameter $t_{i,j}$, and the earliest available time of the customer at node j, that is, e_j. The following inequality constraint defines u_i^k.

$$u_j^k \ge \max\{u_i^k + s_i + t_{i,j}, e_j\} x_{i,j}^k \tag{3}$$

If j is a destination node, that is, $j > n$, then u_j^k is simply the arrival time of vehicle k, since $e_j = 0$. Otherwise, if j is an origin node, that is, $1 \le j \le n$, u_j^k is the time when customer j is picked up by the vehicle.

To impose the capacity constraint, we let w_i^k denote the load of vehicle k, that is, the number of passengers that it carries when travelling from node i to j. Given there are q_j passengers boarding/alighting the vehicle at node j, we have

$$w_j^k = (w_i^k + q_j) x_{i,j}^k. \tag{4}$$

The objective function consists of the following components:

- Vehicle cost consists of two parts: (a) $\sum_{i,j \in \mathcal{V}} c_d \, d_{i,j} x_{i,j}^k$, where $d_{i,j}$ is the distance between nodes i and j, c_d is the coefficient that evaluates the vehicle travel cost per km travelled; and (b) $c_v \text{sgn}(u_0^k)$ is the vehicle capital cost, where $\text{sgn}(\cdot)$ is a function that returns 1 if $u_0^k > 0$, and 0 otherwise.
- Penalty for late pick-up and/or drop-off: $c_w \max\{u_i^k - e_i, 0\}^{\alpha_w} + c_l \max\{u_i^k - l_i, 0\}^{\alpha_l}$, where c_w, α_w, c_l, and α_l are coefficients for evaluating the costs for customer $i \in \mathcal{P}$ waiting at the origin node i for pick-up, the detour time, as well as any late time (beyond the time window).

$$\min \sum_{k \in \mathcal{K}} \left(\sum_{i,j \in \mathcal{V}} c_d \, d_{i,j} x_{i,j}^k + c_v \text{sgn}(u_0^k) \right) + \sum_{i \in \mathcal{V}} \big(c_w \max\{u_i^k - e_i, 0\}^{\alpha_w} + c_l \max\{u_i^k - l_i, 0\} \big). \tag{5}$$

s.t. $\sum_{k \in \mathcal{K}} \sum_{j \in \mathcal{V}} x_{i,j}^k = 1, \forall i \in \mathcal{V}$ (6)

$\sum_{j \in \mathcal{V}} x_{i,j}^k = \sum_{j \in \mathcal{V}} x_{n+i,j}^k, \forall i \in \mathcal{V}, k \in \mathcal{K}$ (7)

$\sum_{j \in \mathcal{V}} x_{0,j}^k = \sum_{j \in \mathcal{V}} x_{j,0}^k = 1, \forall k \in \mathcal{K}$ (8)

$\sum_{j \in \mathcal{V}} x_{i,j}^k = \sum_{j \in \mathcal{V}} x_{j,i}^k, \forall i \in \mathcal{V} \cup \{0\}, k \in \mathcal{K}$ (9)

$u_{n+i}^k \ge u_i^k, \forall i \in \mathcal{V}, k \in \mathcal{K}$ (10)

$u_j^k \ge \max\{u_i^k + s_i + t_{i,j}, e_j\} x_{i,j}^k, \forall i, j \in \mathcal{V}, k \in \mathcal{K}$ (11)

$w_j^k = (w_i^k + q_j) x_{i,j}^k, \forall i, j \in \mathcal{P}, k \in \mathcal{K}$ (12)

$w_i^k \le C_k, \forall i \in \mathcal{V}, k \in \mathcal{K}$ (13)

$x_{i,j}^k \in \{0,1\}, u_i^k, w_i^k \ge 0, \forall i, j \in \mathcal{V}, k \in \mathcal{K}$ (14)

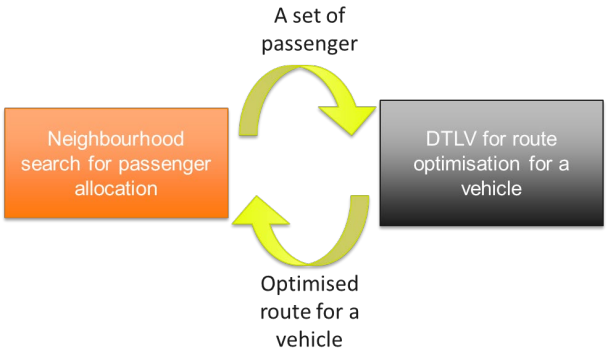
Constraint (6) specifies that each customer is served (picked up) by one vehicle exactly once. Constraint (7) guarantees that if a customer i is (or not) picked up by vehicle k from their origin node i, the vehicle must (or must not) drop them off at their destination node $n + i$. Constraint (8) ensures that every vehicle departs from its depot and returns to it. The flow conservation is guaranteed by (9) together with (8). Constraint (10) ensures that a customer is picked up before getting dropped off, while (11) constrains the effective arrival times of vehicles at nodes. Constraint (12) defines the vehicle load. The vehicle capacity is respected by (13). Constraint (14) defines the domains of the variables.

4. The proposed algorithm

This section outlines the proposed solution for the DARP. The main approach is to decompose the problem into two hierarchical subproblems (as shown in Figure 2):

- the trip allocation problem at the top layer, which is solved by neighbourhood search, and
- multiple instances of the single-vehicle pickup-and-delivery problem with time windows (PDPTW) at the bottom, which is solved by the Delaunay-triangulation based algorithm.

Figure 2: Workflow



The following subsections will detail the algorithm at each layer.

4.1. Neighbourhood search for passenger allocation

At this layer, the algorithm employs the neighbourhood search technique to optimise the trip allocation to vehicles, as detailed in Algorithm 1 below.

Algorithm 1 Trip allocation neighbourhood search

Input:

$\mathcal{R}^k$ ▷ where $\mathcal{R}^k$ is the route of vehicle k

$\Theta^k := \{\tau_i\}$ ▷ where τ_i represents trip i and Θ^k is the set of all trips assigned to vehicle k

Output:

$\mathcal{R}^k_{\text{best}}$

Θ^k_{best}

1. **procedure** TRIPALLOCATIONNEIGHBOURHOODSEARCH$(\mathcal{R}^k, \Theta^k)$
2. $\quad\Theta^k_{\text{best}} \leftarrow \Theta^k,\ \forall k.$
3. $\quad\mathcal{R}^k_{\text{best}} \leftarrow \mathcal{R}^k,\ \forall k.$
4. $\quad$**for** a pre-specified number of operations **do**
5. $\quad\quad\Theta^{\text{neighbour},k}_{\text{best}} \leftarrow \Theta^k,\ \forall k\ .$

6. $\mathcal{R}_{\text{best}}^{\text{neighbour},k} \leftarrow \mathcal{R}^k,\ \forall k$.
7. **for** $\gamma \in \{1,2,\dots,\text{neighbourhood size}\}$ **do**
8. Randomly choose a trip $\tau_i \in \bigcup_k \Theta^k$.
9. $k \leftarrow \eta_1$ ▷ η_1 is the vehicle index of the chosen trip.
10. $\beta \leftarrow \{k \mid \Theta^k \neq \emptyset\} \setminus \{\eta_1\} \cup \min\{k \mid \Theta^k = \emptyset\}$.
11. Reinsert the trip to a random vehicle $k \in \beta$. ▷ greedy insertion
12. $k \leftarrow \eta_2$ ▷ η_2 is the vehicle index being inserted.
13. $\Theta_{\gamma}^{\text{neighbour},k} \leftarrow$ the new trip allocation after reinsertion, $\forall k$.
14. $\mathcal{R}_{\gamma}^{\text{neighbour},k} \leftarrow \mathcal{R}^k,\ \forall k \notin \{\eta_1, \eta_2\}$
15. Re-optimise $\mathcal{R}^{\eta_1}$ and $\mathcal{R}^{\eta_2}$ using Algorithm 2.
16. $\mathcal{R}_{\gamma}^{\text{neighbour},\eta_1} \leftarrow \mathcal{R}^{\eta_1}$
17. $\mathcal{R}_{\gamma}^{\text{neighbour},\eta_2} \leftarrow \mathcal{R}^{\eta_2}$
18. **if** the objective function value (5) of $\mathcal{R}_{\gamma}^{\text{neighbour}}$ is lower than that of $\mathcal{R}_{\text{best}}^{\text{neighbour}}$ **then**
19. $\Theta_{\text{best}}^{\text{neighbour},k} \leftarrow \Theta_{\gamma}^{\text{neighbour},k},\ \forall k$.
20. $\mathcal{R}_{\text{best}}^{\text{neighbour},k} \leftarrow \mathcal{R}_{\gamma}^{\text{neighbour},k},\ \forall k$.
21. **end if**
22. **end for**
23. **if** the objective function value (5) of $\mathcal{R}_{\text{best}}^{\text{neighbour}}$ is lower than that of $\mathcal{R}_{\text{best}}$ **then**
24. $\Theta_{\text{best}}^{k} \leftarrow \Theta_{\text{best}}^{\text{neighbour},k},\ \forall k$
25. $\mathcal{R}_{\text{best}}^{k} \leftarrow \mathcal{R}_{\text{best}}^{\text{neighbour},k},\ \forall k$
26. **end if**
27. $\Theta^k \leftarrow \Theta_{\text{best}}^{\text{neighbour},k},\ \forall k$
28. $\mathcal{R}^k \leftarrow \mathcal{R}_{\text{best}}^{\text{neighbour},k},\ \forall k$
29. **end for**
30. **return** $\Theta_{best}^{k}, \mathcal{R}_{best}^{k}$
31. **end procedure**

The algorithm requires an initial solution estimate as an input, which can be generated by any means, such as greedy insertion. At each iteration (of the for-loop at Line 4), the algorithm will generate and evaluate multiple neighbour solutions by slightly modifying the "base solution" of the current iteration k , i.e., Θ^k and $\mathcal{R}^k$ (Lines 7–22). If the best neighbor solution ($\Theta_{\text{best}}^{\text{neighbour},k}$ and $\mathcal{R}_{\text{best}}^{\text{neighbour},k}$) is better than the current solution estimate (Θ_{best}^{k} and $\mathcal{R}_{\text{best}}^{k}$), the algorithm will update the solution estimate accordingly (Lines 23–26). Finally, the algorithm will update the base solution (Lines 27–28) and move on to the next iteration. The algorithm produces the solution estimate at the end of the iteration as an output (Line 30).

To generate a neighbour solution, the algorithm randomly chooses a trip (Line 8) and moves it to another vehicle randomly selected from a feasible set β (Lines 8–12). Note that the feasible set includes a single empty vehicle to allow the possibility to "procure" more vehicles when needed (Line 10). Then, the routes of the modified vehicles (Vehicles η_1 and η_2) are re-optimised (Line 15) and the generated neighbour solution is stored (Lines 13–14, 16–17).

4.2. Delaunay-triangulation-based algorithm for single-vehicle PDPTW

At the bottom layer, the newly developed Delaunay-triangulation based algorithm is employed. The algorithm solves the single-vehicle PDPTW inherited from Algorithm 1 (Line 15). This algorithm maps the pick-up and drop-off stops into a 3D spatiotemporal space (two

"horizontal" axis for location coordinates and the "vertical" axis for time) and iteratively generates a DT structure created from an appropriate subset of these stops to help generate the route solution.

Algorithm 2 Delaunay-triangulation lowest vertices (DTLV)

Input:
$\mathcal{D} \coloneqq \{(p_i, d_i, t_i)\}$ ▷ where $\mathcal{D}$ is the set of N demand, each with a pick-up node p_i, drop-off node d_i, and request time t_i
$\mathcal{N}$ ▷ where $\mathcal{N}$ is the corresponding set of all nodes (both pick-ups and drop-offs)

Output:
$\mathcal{R} \coloneqq (s_1, s_2, \cdots, s_{2n})$ ▷ where $\mathcal{R}$ is a sequence of stops; s_i representing the generated route for the vehicle serving all trip demands

procedure DTLV($\mathcal{D}, \mathcal{N}$)
  $\alpha \leftarrow \{p_i, 1 \le i \le n\}$ ▷ the set of all available nodes that can be feasibly visited; in this instance consists of all the possible pick-up nodes only
  Denote the index e as the demand with the earliest pick-up request, i.e., $t_e \le t_i, 1 \le i \le n$.
  $s_1 \leftarrow p_e$ ▷ set the earliest pick-up node as the first stop
  $\alpha \leftarrow \alpha \cup \{d_e\}$ ▷ add d_e to α
  $k \leftarrow 1$
  for $2n - 4$ times **do**
    $k \leftarrow k + 1$
    $\gamma \coloneqq \{(x_i, y_i, z_i)\} \leftarrow$ the set of the nodes in α mapped into a 3D space
      ▷ The x and y axes denote the node's locations, while the z-axis indicate the node's service time (i.e. the start of the pick-up/drop-off time window).
    Generate a DT using γ.
    if a DT cannot be generated **then**
      Denote the index e as the node with the lowest z-value, i.e., $z_e < z_i, \forall i \in \gamma \setminus \{e\}$.
      $\epsilon \leftarrow$ find the trip demand index that corresponds to node e.
      $g_\epsilon \leftarrow$ store the stop that corresponds to node e, this can either be p_ϵ or d_ϵ
    else
      $\mathcal{C} \leftarrow$ the set of the N_c nodes that are connected to the current node by DT edges with the lowest z-values; these nodes become the candidate for the next stop
      $\mathcal{S} \leftarrow$ the set of the scores of each candidate evaluated by (15)
      Denote the index e as the node in $\mathcal{C}$ with the best score in $\mathcal{S}$.
      $\epsilon \leftarrow$ find the trip demand index that corresponds to node e.
      $g_\epsilon \leftarrow$ store the stop that corresponds to node e, this can either be p_ϵ or d_ϵ
    end if
    $s_k \leftarrow g_\epsilon$ ▷ select the node g_ϵ as the next stop
    $\alpha \leftarrow \alpha \setminus \{s_{k-1}\}$ ▷ remove the previous stop from the available node set α
    if g_ϵ is a pick-up **then**
      $\alpha \leftarrow \alpha \cup \{d_\epsilon\}$ ▷ add the corresponding drop-off d_ϵ to the available node set
    end if
  end for
  $\sigma_{\text{best}} \leftarrow \infty$
  for each possible permutation of the remaining 3 nodes $(g_1^{\text{final}}, g_2^{\text{final}}, g_3^{\text{final}})$ **do**
    $\sigma \leftarrow$Using this permutation, evaluate the score of the complete route based on (5)
    if $\sigma < \sigma_{\text{best}}$ **then**
      $\left(s_1^{\text{final}}, s_2^{\text{final}}, s_3^{\text{final}}\right) \leftarrow$ the current permutation of $(g_1^{\text{final}}, g_2^{\text{final}}, g_3^{\text{final}})$
    end if
  end for
  $(s_{2n-2}, s_{2n-1}, s_{2n}) \leftarrow \left(s_1^{\text{final}}, s_2^{\text{final}}, s_3^{\text{final}}\right)$
  return $\mathcal{R}$
end procedure

The algorithm selects the earliest stop as a starting point (Line 4) and sequentially builds the route by selecting the next "best" stop (Lines 7–27). At each iteration sequence, the algorithm creates a DT structure from the available stops, i.e., the current stop, the remaining pick-up stops, and the drop-off stops of passengers on board (Lines 9–10). If a DT cannot be created, then the algorithm simply chooses the next earliest node as the next stop (Lines 11–14). On the other hand, if a DT structure is created, the algorithm focuses on analysing several "lowest" vertices (meaning the stops with the closest time deadlines) to be selected as the next stop (Lines 16–20). Once there are only three nodes remaining, the algorithm simply computes the costs of all six possible permutations of the stops and selects the best one (Lines 29–35).

When investigating the next stop candidates (Lines 16–20), the logic of limiting the analysis to only several of the lowest vertices was derived from the authors' observations on the optimal solutions of numerous self-generated sample problems. It was noted that the optimal solution to the single-vehicle PDPTW often selects the lowest vertex as the next stop, and almost never goes beyond the third lowest vertex. Hence, in this algorithm, the analysis is limited to the four lowest vertices.

Denote s_k as the current stop and $\hat{s}_{k+1}$ as a candidate for the next stop. The score of $\hat{s}_{k+1}$ for the purpose of the selection of the vertex as the next stop is governed by the following equation.

$$F_{\text{score}} = F_{\text{grad}} + F_{\text{future}} + R_{\text{nextstop}} + R_{\text{distance}} + R_{\text{horizontal}} + R_{\text{timewindow}} \tag{15}$$

with details as follows.

- $$F_{\text{grad}} = \begin{cases} f_{\text{low}} & \text{if } 0 \le \nabla \le grad_{\text{low}} \\ f_{\text{med}} & \text{if } grad_{\text{low}} < \nabla \le grad_{\text{med}} \\ f_{\text{high}} & \text{if } grad_{\text{high}} < \nabla \le grad_{\text{high}} \\ f_{\text{neg}} & \text{if } \nabla < 0 \end{cases} \tag{16}$$
- ∇ is the gradient between s_k and $\hat{s}_{k+1}$, i.e., $\frac{\Delta z}{\sqrt{\Delta x^2 + \Delta y^2}}$
- F_{future} is computed by scaling the future cost f_{future} of the candidate $\hat{s}_{k+1}$ into a score. The future cost f_{future} is the route cost up to the next $k + 1 + k_{\text{future}}$ stops. The future stops are obtained by assuming the considered candidate $\hat{s}_{k+1}$ is used and a further k_{future} nodes are selected by using the loop in Algorithm 2 (Lines 7–27) but without future cost consideration (to avoid recurrent function calls). Then, the score is computed as follows,

  $$F_{\text{future}} = \left(1 - \frac{f_{\text{future}}}{f_{\text{future}}^{max} - f_{\text{future}}^{min}}\right) \cdot \text{future}_{max} \tag{17}$$

  where $f_{\text{future}}^{\max}$ and $f_{\text{future}}^{\min}$ are the maximum (worst) and minimum (best) future cost, respectively, among the candidate next stops $\mathcal{C}$. Furthermore, Equation (17) takes into account the "gap" between the best and worst future costs. If the gap is small, F_{future} dominates (15) and the future cost becomes the main factor determining the selection of the next stop.
- R_{variable} is a function that returns an integer value between r_{variable} and 0 depending on the (lowest) ranking of the indicated variable among all the N_c candidates. The candidate with the lowest variable value will obtain the maximum score r_{variable} and the score will incrementally be deducted by one for each next ranked candidate, with a lower bound of zero score.
- R_{nextstop} ranks the preliminary cost of inserting the candidate $\hat{s}_{k+1}$ into the (incomplete) route, as evaluated by (5). So, the cost is only computed until $\hat{s}_{k+1}$.

- R_{distance} ranks the 3D Euclidean distance between s_k and $\hat{s}_{k+1}$.
- $R_{\text{horizontal}}$ ranks the distance between s_k and $\hat{s}_{k+1}$ projected into the xy-plane.
- $R_{\text{timewindow}}$ ranks the candidate based on the earliest time window.

5. Numerical study

This section presents the results of the application of the proposed algorithm to solve PDPTW.

5.1. Methodology and setup

5.1.1. Case study area

The numerical study simulates the Port Melbourne area in Melbourne, Australia (Figure 3). Multiple sets of trip requests were randomly generated throughout the area, with each demand consists of a pick-up location, a drop-off location, and a trip start time. Figure 3 shows an illustration of the demand location with the circle size indicating the "popularity" of a location. As can be noted, many of the trips are originating or going towards a node at the top right-hand corner of the map, which is the Southern Cross train station. The Southern Cross train station is a main transit hub, especially for trip to/from Port Melbourne area.

For this study, a total of 22 different demand datasets were created, consisting of either 50 (3 datasets), 100 (6 datasets), 150 (6 datasets), 200 (6 datasets), or 500 (1 dataset) number of demands (i.e., trip requests). For simplicity of the result analysis, each trip request consists of one passenger, although technically the algorithm is able to handle group bookings. Furthermore, the request time is normally distributed with a mean at 5:30 PM and standard deviation of 1 hour, but is limited to between 4–7 PM. Moreover, in order to generate the locations, a metric that compares the public transport and car travel (Kutadinata et al. 2021) is used as a prioritisation consideration, leading to more trips being generated to/from areas where car travels are more beneficial to the travellers.

Figure 3: Port Melbourne map showing the demand distribution

5.1.2. Implementation details

For this numerical study, the parameters used for the algorithms are shown in Table 1.

Table 1: Parameters used for the algorithms

Description	Notation	Values
Neighbourhood size	-	30
No. of iterations for Algorithm 1	-	300
No. of next stop candidates	N_c	4
Gradient scores for DTLV	$f_{\text{low}}, f_{\text{med}}, f_{\text{high}}, f_{\text{neg}}$	1.5, 3.5, -1, 4
Gradient thresholds for DTLV	$grad_{\text{low}}, grad_{\text{med}}, grad_{\text{high}}$	0.5, 0.8, 99
Future cost parameters	$\text{future}_{max}, k_{\text{future}}$	100, 1
Maximum ranking scores for R	$r_{\text{nextstop}}, r_{\text{distance}}, r_{\text{horizontal}}, r_{\text{timewindow}}$	4.5, 2, 2.5, 3
Vehicle capacity	C_k	9
Vehicle distance cost rate	c_d	1 /km
Vehicle capital cost	c_v	2000 or 20,000
Wait penalty	c_w, α_w	1, 1
Late penalty	c_l, α_l	10, 2

The algorithm was implemented in MATLAB and the scripts were run at Deakin University's remote desktop facility. The computation utilised MATLAB parallel computing with four workers.

As a benchmark, the performance of the proposed algorithm will be compared against a neighbourhood-search algorithm as described by Czioska et al. (2019) and Kutadinata et al. (2019). Essentially, the benchmark neighbourhood-search algorithm follows the same two-level approach and utilises Algorithm 1. However, it utilises another neighbourhood-search algorithm for the bottom layer (i.e., the single vehicle case) instead of Algorithm 2. For the rest of the paper, the benchmark neighbourhood-search algorithm will simply be referred to as the NS algorithm, whereas the proposed algorithm will be referred to as the DTLV algorithm.

5.2. Results

In this subsection, the simulation results of using DTLV and NS are compared. The comparison includes the algorithm runtime, the final optimisation score (as computed with (5)), as well as the detailed statistics of the customer's experience (i.e. the level of service) and the vehicle operations. Finally, the effect of the vehicle capital cost c_v on the solutions is investigated by scrutinising the changes in the number of used vehicles and the level of service.

Firstly, Figure 4 shows the average runtime reduction and the optimisation score decrease when utilising DTLV. It can be seen that DTLV achieves significant computation saving with an almost negligible score decrease. The runtime reduction is at least 57% and up to 71%, whereas the score decrease is only up to 1.6%. Furthermore, it can be seen from Figure 5 that, even though the percentage of runtime reduction on the higher number of demands is smaller, the actual time savings are more prominent on the higher number of demands, growing from 530 seconds of time savings with 50 demands to 917 seconds of time savings for 500 demands.

Figure 4: Results of runtime reduction and score decrease

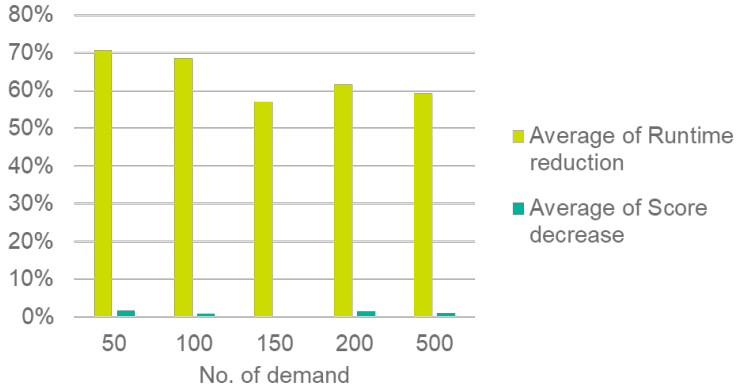
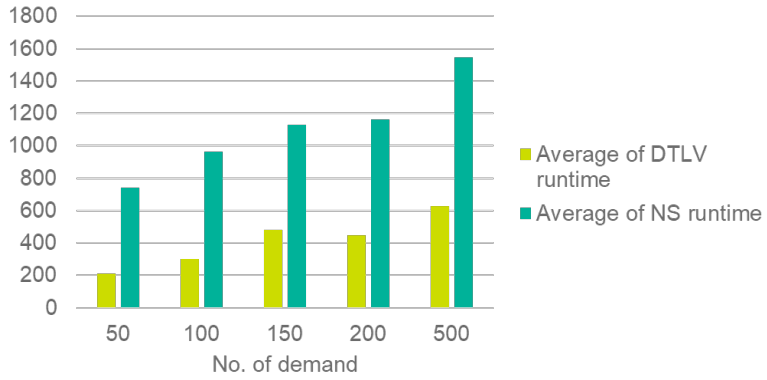


Figure 5: Comparison of runtime



Secondly, Tables 2 and 3 show the comparison of the detailed statistics between the DTLV and NS. These tables show the differences in the operation resources and the service level, with a positive number indicating that the number obtained from the DTLV is higher, and vice versa. It can be observed that DTLV generally leads to higher use of the operational resources, indicated by the higher number of vehicles used, longer total operating times, higher VKT, as well as higher inefficiency of vehicle utilisation (dead running and idling). However, these are compensated by a lower wait time. Furthermore, the detour time difference seems to increase as the number of demands increases, with DTLV yielding lower detour time at a smaller number of demands.

Table 2: Statistics totalled from all vehicles

	No. of demands				
Average of difference in...	**50**	**100**	**150**	**200**	**500**
No. of vehicles used	0	0.5	0.33	0.33	-1
Total operating time (min)	18	75	60	-20	96
Total VKT (km)	2.38	3.97	1.50	6.60	26.88
Total idle time (min)	12	62	54	-36	28
Total empty km/dead run (km)	1.04	0.97	-3.80	5.53	11.72
Total empty idle (min)	7	57	61	-36	19

Table 3: Statistics totalled from all demands/trips

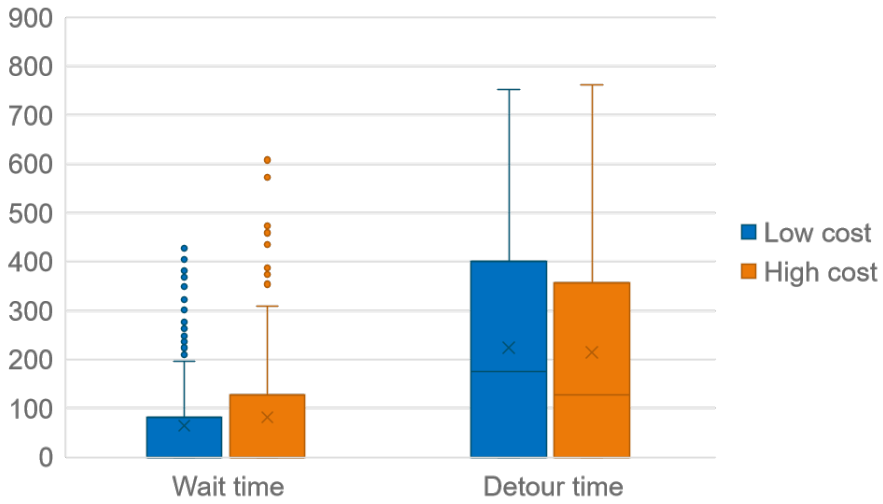
Average of difference in...	No. of demands 50	100	150	200	500
Total late pick-up time (min)	0	0	0	0	0
Total late drop-off time (min)	0	0	0	0	-0.5
Total wait time (min)	-9.5	-8.0	-25.1	-10.3	16.8
Total detour time (min)	-19.6	-7.0	8.5	2.8	53.7

Having said that, it should be noted that the differences are arguably insignificant when considering each individual vehicle or customer, even in the 500-demand case that has the highest magnitude differences. For instance, the total difference of 26.88 km VKT was aggregated from a total of 42 (DTLV) and 43 (NS) vehicles, meaning each vehicle's VKT only differs by approximately 600 metres. As another example, the total detour time of 53.7 minutes is attributed to 500 customers, resulting in only 6 seconds of difference per person between DTLV and NS.

Finally, the simulations were re-run with the vehicle capital cost c_v changed from 2,000 (low cost) to 20,000 (high cost). The results showed that most of the routing solutions being produced are utilising 1 or 2 fewer vehicles. For NS, the solutions for 11 out of the 22 datasets are using 1 fewer vehicle, another dataset is using 2 fewer vehicles, 9 datasets have no change, while there is an outlier where one dataset ends up using one more vehicle. For DTLV, the increased vehicle capital cost results in 15 datasets using 1 fewer vehicle, and another one using 2 fewer vehicles.

Figure 6 illustrates the typical impact of the increased capital cost on the wait time and detour time. The data is taken from one of the datasets with 200 demands using DTLV. Note that there is no late penalty being incurred in this case, as well as in most cases. As can be observed, the wait time distribution is worse (mean difference of 18 seconds), whereas the detour time has surprisingly improved (mean difference of 9 seconds). Coupled with the fact that the wait time is doubly penalised by the objective function (5), there is sufficient evidence to conclude that the service level has decreased for the cases with the high vehicle capital cost.

Figure 6: Results of wait and detour times with changed vehicle capital cost c_v



6. Conclusion

In conclusion, the proposed algorithm shows the capability to match the optimality of an established method of neighbourhood search algorithm with significantly faster computing time. This is mainly attributed to the fact that DTLV is not a search heuristic like NS. It was then shown in a case study area of Port Melbourne that the DTLV was able to optimise the simulated PDPTW problems, and how the formulation was able to be implemented and tailored to achieve a desired goal. These outcomes are highly relevant when considering the emerging ubiquity of ICT and the popularity of a shared and flexible economy in transport. For potential future works, several improvements have been identified: utilising deep learning techniques for the next stop selection in DTLV, developing a DT-based passenger allocation algorithm, and developing a framework to handle real-time requests.

References

Beasley, JE & Christofides, N 1997, 'Vehicle routing with a sparse feasibility graph', *Eur. J. Oper. Res.*, vol. 98, pp. 499-511.

Braekers, K, Caris, A & Janssens, GK 2014, 'Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots', *Transp. Res. Part B: Methodol.*, vol. 67, pp. 166-186.

CORDIS 2020, *EU Research results, Autonomous on-demand buses underway in the streets of Europe*, viewed 15 April 2023, https://cordis.europa.eu/article/id/415431-autonomous-on-demand-buses-underway-in-the-streets-of-europe.

Cubillos, C, Urra, E & Rodríguez, N, 'Application of genetic algorithms for the DARPTW problem', *Int. J. Comput. Commun. Control*, vol. 4, no. 2, pp. 127-136.

Czioska, P, Kutadinata, R, Trifunović, A, Winter, A, Sester, M & Friedrich, B 2019, 'Real-world meeting points for shared demand-responsive transportation systems', *Public Transport*, vol. 11, pp. 341-377.

Delaunay, B 1934, 'Sur la sphère vide. A la mémoire de Georges Voronoï', *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, vol. 6, pp. 793-800.

Detti, P, Papalini, F & de Lara, GZM 2017, 'A multi-depot dial-a-ride problem with heterogeneous vehicles and compatibility constraints in healthcare', *Omega*, vol. 70, pp. 1-14.

Ho, SC, Szeto, WY, Kuo, YH, Leung, JM, Petering, M & Tou, TW 2018, 'A survey of dial-a-ride problems: Literature review and recent developments', *Transportation Research Part B: Methodological*, vol. 111, pp. 395–421, doi: 10.1016/j.trb.2018.02.001.

Johnsen, LC & Meisel, F 2022, 'Interrelated trips in the rural dial-a-ride problem with autonomous vehicles', *European Journal of Operational Research*, vol. 303, no. 1, pp. 201-219, doi: 10.1016/j.ejor.2022.02.021.

Kantabura, V 1983, 'Traveling salesman cycles are not always subgraphs of Voronoi duals', *Inform. Process. Lett.*, vol. 16, pp. 11-12.

Kaufman, B 2020, '1 million rides and counting: on-demand services bring public transport to the suburbs', *The Conversation*, viewed 15 April 2023, https://theconversation.com/1-million-rides-and-counting-on-demand-services-bring-public-transport-to-the-suburbs-132355.

Krasnogor, N, Moscato, P & Norman, M 1995, 'A new hybrid heuristic for large geometric traveling salesman problems based on the Delaunay triangulation', paper presented at the *Anales del XXVII simposio Brasileiro de Pesquisa Operacional*, Vitoria, Brazil.

Kutadinata, R, Thompson, R & Winter, S 2019, 'Passenger-Freight Demand Responsive Transport Services: A Dynamic Optimisation Approach', *26th ITS World Congress*, Singapore.

Kutadinata, R, Dey, S & Leow, D 2021, 'Relative mobility analysis of a public transport network in comparison with car travel', *Transp. Res. Record*, vol. 2675, no. 11, pp. 214-225.

Kuźmicz, K, Ryciuk, U, Glińska, E, Kiryluk, H & Rollnik-Sadowska, E 2022, 'Perspectives of mobility development in remote areas attractive to tourists', *Ekonomia i Środowisko - Economics and Environment*, vol. 80, no. 1, pp. 150-188.

Laker, L 2022, 'All aboard! How on-demand public transport is getting back on the road', *The Guardian*, viewed 15 April 2023, https://www.theguardian.com/technology/2022/aug/11/ all-aboard-how-on-demand-public-transport-is-getting-back-on-the-road.

Lau, SK & Shue, LY 2001, 'Solving travelling salesman problems with an intelligent search approach', *APAC J Oper. Res.*, vol. 18, pp. 77-87.

Liang, X, de Almeida Correia, GH, An, K & van Arem, B 2020, 'Automated taxis' dial-a-ride problem with ride-sharing considering congestion-based dynamic travel times', *Transportation Research Part C: Emerging Technologies*, vol. 112, pp. 260-281.

Liu, M, Luo, Z & Lim, A 2015, 'A branch-and-cut algorithm for a realistic dial-a-ride problem', *Transp. Res. Part B: Methodol.*, vol. 81, pp. 267-288.

Malheiros, I, Ramalho, R, Passeti, B, Bulhões, T & Subramanian, A 2021, 'A hybrid algorithm for the multi-depot heterogeneous dial-a-ride problem', *Computers & Operations Research*, vol. 129, no. 105196, doi: 10.1016/j.cor.2020.105196.

Molenbruch, Y, Braekers, K & Caris, A 2017a, 'Typology and literature review for dial-a-ride problems', *Ann Oper Res*, vol. 259, pp. 295–325, doi: 10.1007/s10479-017-2525-0.

Molenbruch, Y, Braekers, K & Caris, A 2017b, 'Benefits of horizontal cooperation in dial-a-ride services', *Transp. Res. Part E: Logist. Transp. Rev.*, vol. 107, pp. 97-119.

Pimenta, V, Quilliot, A, Toussaint, H & Vigo, D 2017, 'Models and algorithms for reliability-oriented dial-a-ride with autonomous electric vehicles', *Eur. J. Oper. Res.*, vol., 257, no. 2, pp. 601-613.

PTV 2023, 'FlexiRide, Public Transport Victoria', *PTV website*, viewed 15 April 2023, https://www.ptv.vic.gov.au/more/travelling-on-the-network/flexiride/.

Qu, Y & Bard, JF 2015, 'A branch-and-price-and-cut algorithm for heterogeneous pickup and delivery problems with configurable vehicle capacity', *Transp. Sci.*, vol. 49, no. 2, pp. 254-270.

Ritzinger, U, Puchinger, J & Hartl, RF 2016, ‚Dynamic programming based metaheuristics for the dial-a-ride problem', *Ann. Oper. Res.*, vol. 236, no. 2, pp. 341-358.

Sazonov, VV, Skobelev, PO, Lada, AN & Mayorov, IV 2018, 'Application of multiagent technologies to multiple depot vehicle routing problem with time windows', *Automation and Remote Control*, vol. 79, no. 6, pp. 1139-1147.

Sharif Azadeh, S, Atasoy, B, Ben-Akiva, ME, Bierlaire, M & Maknoon, MY 2022, 'Choice-driven dial-a-ride problem for demand responsive mobility service', *Transportation Research Part B: Methodological*, vol. 161, pp. 128-149, doi: 10.1016/j.trb.2022.04.008.

Su, Y, Dupin, N & Puchinger, J 2023, 'A deterministic annealing local search for the electric autonomous dial-a-ride problem', *European Journal of Operational Research*, vol. 309, no. 3, pp. 1091-1111, doi: 10.1016/j.ejor.2023.02.012.

Translink 2023, 'On demand Gold Coast', *Translink website*, viewed 15 April 2023, https://translink.com.au/travel-with-us/on-demand/gold-coast.

Tu, W, Fang, Z, Li, Q, Shaw, S & Chen, B 2014, 'A bi-level Voronoi diagram-based metaheuristic for a large-scale multi-depot vehicle routing problem', *Transp. Res. Part E*, vol. 61, pp. 84-97.