New Feasibility Test Driven Adaptive Large Neighborhood Search for Time-Dependent Dial-a-Ride Problems

Wenjie Hu[a], Yang Wang[a], Jin-Kao Hao[b], Wei-Neng Chen[c], Jiliu Li[a,*], Xin Jin[d]

[a]*School of Management, Northwestern Polytechnical University, Xi'an 710072, China*
[b]*Department of Computer Science, Université d'Angers, Angers 49045, France*
[c]*School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China*
[d]*Cloud Reliability Lab, Huawei Cloud Computing Technologies, Beijing 100085, China*

Omega 144: 103577, 2026 https://doi.org/10.1016/j.omega.2026.103577

Abstract

This work investigates three practical variants of the dial-a-ride problem with different objectives, called time-dependent dial-a-ride problems (TD-DARPs), where the travel time along each arc depends on the departure time of the vehicle from the starting node. We develop a general mathematical model to accommodate different objectives and attributes, and propose a feasibility test driven adaptive large neighborhood search algorithm based on the feature of time-dependent travel times. The algorithmic framework consists of three main modules: an adaptive destroy and repair procedure to enhance solution diversity, a tabu-based solution improvement method to refine solution quality, and a simulated annealing-based update criterion to escape local optima. Unlike the traditional and time-consuming feasibility test approach in the existing algorithms for TD-DARPs, our algorithm employs a novel feasibility test with linear time complexity, which leverages the first-in-first-out property and the structure of the travel time function to efficiently evaluate the feasibility of each insertion operation. Extensive numerical experiments on benchmark instances demonstrate that our algorithm outperforms the CPLEX solver and a state-of-the-art metaheuristic algorithm in terms of solution quality and running time.

Keywords: time-dependent; dial-a-ride problem; feasibility test; metaheuristic

1. Introduction

The dial-a-ride problem (DARP) (Cordeau and Laporte, 2003; Schenekemberg et al., 2022) is a well-known NP-hard problem that involves scheduling and routing vehicles to serve a set of passengers with specific pickup and delivery locations, while taking into account the passengers' maximum travel time constraint. Typical applications of DARP arise in door-to-door transportation services for patients, the elderly, individuals with disabilities, and last-mile transit (Zhang

* Corresponding author.
Email addresses: huwenjie@mail.nwpu.edu.cn (Wenjie Hu), yangw@nwpu.edu.cn (Yang Wang), jin-kao.hao@univ-angers.fr (Jin-Kao Hao), cschenwn@scut.edu.cn (Wei-Neng Chen), jiliuli@nwpu.edu.cn (Jiliu Li), jinxin109@huawei.com (Xin Jin)

et al., 2015; Detti et al., 2017; Sayarshad and Chow, 2015). From a practical perspective, certain scenarios require that all transport requests be served (Timothée et al., 2023), while others require that a subset of requests be selected for service due to vehicle resource limitations (Ouasaid and Saddoune, 2024).

Most DARP studies assume a constant travel time between nodes. However, in reality, the travel time may vary significantly between peak and off-peak hours. Time-dependent travel times have been considered in several vehicle routing problems (VRPs) (Ichoua et al., 2003; Liu et al., 2020; Pan et al., 2021). In the context of DARP, time-dependent travel times are also inevitably encountered. However, this feature is often neglected in existing research. In order to accurately reflect real traffic conditions and thus facilitate more informed and effective route planning, it is crucial to incorporate time-dependent travel times into the DARP formulation. While stochastic time-dependent models can represent random variations in travel times, they also introduce greater computational complexity and require extensive data. In contrast, deterministic time-dependent models strike a practical balance between realism and tractability by capturing systematic variations, such as peak-hour congestion, through historical time-of-day profiles. Building upon this motivation, the study of the challenging time-dependent dial-a-ride problem (TD-DARP) has significant practical importance.

To address the DARP and its variants, various exact and heuristic approaches have been proposed and implemented. Exact approaches for the DARP primarily rely on the branch-and-bound framework, which encompasses techniques like branch-and-cut (Braekers and Kovacs, 2016; Rist and Forbes, 2021), branch-and-price (Parragh et al., 2015; Su et al., 2024), and branch-and-price-and-cut (Qu and Bard, 2015; Karimi et al., 2024) algorithms. Since the DARP is NP-hard, exact approaches are only capable of solving small-sized instances to optimality within an acceptable computation time. For large instances, research focuses on the development of heuristic approaches, including large neighborhood search (Molenbruch et al., 2017; Masmoudi et al., 2020), tabu search (Kirchler and Calvo, 2013; Pandi et al., 2020), deterministic annealing (Braekers et al., 2014; Su et al., 2023), and hybrid algorithms (Gschwind and Drexl, 2019; Malheiros et al., 2021). Ritzinger et al. (2016) introduced a dynamic programming based hybrid large neighborhood search approach to solve large DARP instances. Schenekemberg et al. (2024) proposed a biased random key genetic algorithm combined with the Q-learning and local search to effectively solve the DARP involving the coordination of private fleets, common carriers and public transportation. For a comprehensive survey of the DARP, we refer the reader to the review Ho et al. (2018).

To the best of our knowledge, only a limited number of studies have investigated the TD-DARP (Xiang et al., 2008; Schilde et al., 2014; Hu and Chang, 2015; Zhao et al., 2022). Schilde et al. (2014) studied the dynamic DARP integrating time-dependent and stochastic travel speeds, and

proposed metaheuristic approaches to address both the deterministic TD-DARP and the stochastic TD-DARP. Hu and Chang (2015) developed a DARP formulation that incorporates time-dependent travel times and introduced a branch-and-price approach, which utilizes the large neighborhood search approach to solve the sub-problem aiming to enhance the computation efficiency. Zhao et al. (2022) studied the time-dependent profitable dial-a-ride problem, where a subset of requests are selected for service with the objective of maximizing the net profit. A hybrid algorithm combining the adaptive large neighborhood search with local search techniques was proposed to solve the problem. To achieve a more comprehensive study of the TD-DARP, this paper investigates three TD-DARP variants that differ in their objective and attributes, including the requirement to serve all transportation requests.

The feasibility test for a given TD-DARP route is a critical component of local search-based algorithms, as these algorithms require significant computational resources to evaluate the feasibility of each neighborhood move. In particular, the inclusion of time-dependent settings in the TD-DARP significantly complicates this process. In the literature on the DARP and its variants, feasibility test procedures can be broadly classified into three main categories. The first category is based on the classical eight-step evaluation scheme originally proposed by Cordeau and Laporte (2003), which remains one of the most widely used feasibility checking procedures in the DARP literature (Parragh et al., 2010; Chassaing et al., 2016; Malheiros et al., 2021). This approach propagates service times forward and backward along a route to verify time window, capacity, and ride time constraints. Several improvements and variants of this scheme have been proposed to enhance its computational efficiency. Notably, Sohrabi et al. (2024) developed a revised version of the eight-step procedure with a reduced time complexity, achieving linear time performance under the assumption of bounded vehicle capacity. Their method preserves the structure and elegance of the original scheme while significantly improving computational efficiency in the static DARP setting. The second category formulates the feasibility test as a shortest path problem in a weighted interval graph. Early work by Hunsaker and Savelsbergh (2002) demonstrated that feasibility test under maximum waiting time and ride time constraints can be performed in linear time for a given pickup–delivery sequence. Building on this insight, Firat and Woeginger (2011) showed that the feasibility test can be explicitly formulated as a shortest path problem in a vertex-weighted interval graph, leading to a simple linear time algorithm. The third category consists of preprocessing-based approaches, in which auxiliary information is computed in advance to enable fast feasibility checks during the search. A representative example is the constant-time feasibility test proposed by Gschwind and Drexl (2019), where auxiliary data are precomputed in a preprocessing step to enable efficient feasibility evaluation during the search process.

It is worth noting that the vast majority of the feasibility test methods discussed above are

designed for the classical DARP with static travel times. In particular, preprocessing-based approaches, such as the constant-time feasibility test proposed by Gschwind and Drexl (2019), achieve remarkable efficiency by relying on auxiliary data precomputed under the assumption of time-independent travel times. As a result, these methods do not directly extend to the time-dependent DARP. To the best of our knowledge, the only feasibility test explicitly developed for the TD-DARP is the method proposed by Zhao et al. (2022), which extends the classical eight-step evaluation framework to incorporate time-dependent travel times and ride time constraints. While this approach preserves feasibility correctness, it incurs a relatively high computational complexity, leading to costly neighborhood evaluations in local search algorithms. Consequently, there is a clear need for a new feasibility test for the TD-DARP that properly accounts for time-dependent travel times while retaining high computational efficiency.

In this paper, we introduce the time-dependent travel times into the DARP and study a class of time-dependent dial-a-ride problems that are characterized by three objectives: minimizing the total traveling cost, minimizing the total route duration, and maximizing the total profit. We investigate the characteristics of these problems and propose a novel feasibility test approach for the TD-DARP that is characterized by linear time complexity and shows significant improvements in evaluation efficiency. This approach can be classified within the preprocessing-based category. Based on this feasibility test, we develop a new feasibility test driven adaptive large neighborhood search (FT-ALNS) algorithm for effective solving of the three TD-DARP variants. The algorithm consists of an adaptive destroy and repair procedure to explore the solution space, a tabu-based solution improvement procedure to obtain high-quality local optimal solutions, and a simulated annealing (SA)-based update criterion to help escape from local optimal traps. The experimental results of the proposed algorithm on well-known benchmark instances show that it outperforms the existing state-of-the-art methods and provides a significant improvement in solution quality and solution time.

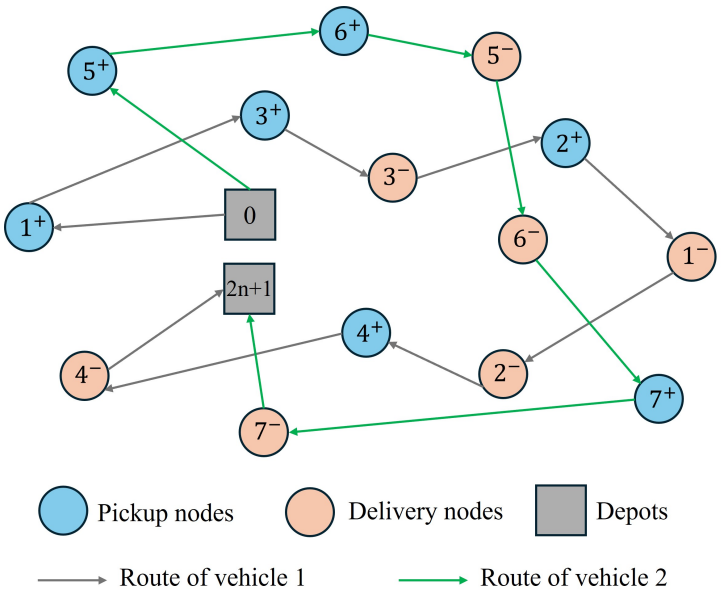
The remainder of this paper is organized as follows. §2 provides the description and the mathematical formulation for the three TD-DARP variants studied. §3 elaborates the latest service time function for the TD-DARP and followed by the feasibility test procedure. To solve the TD-DARP, §4 presents the general framework of the FT-ALNS algorithm for the TD-DARP and the details of three algorithmic components. Computational results are summarized in §5 to evaluate the performance of our algorithm. Finally, we conclude the paper and indicate future research directions in §6.

2. Problem Description and Mathematical Formulation

2.1. Problem Description

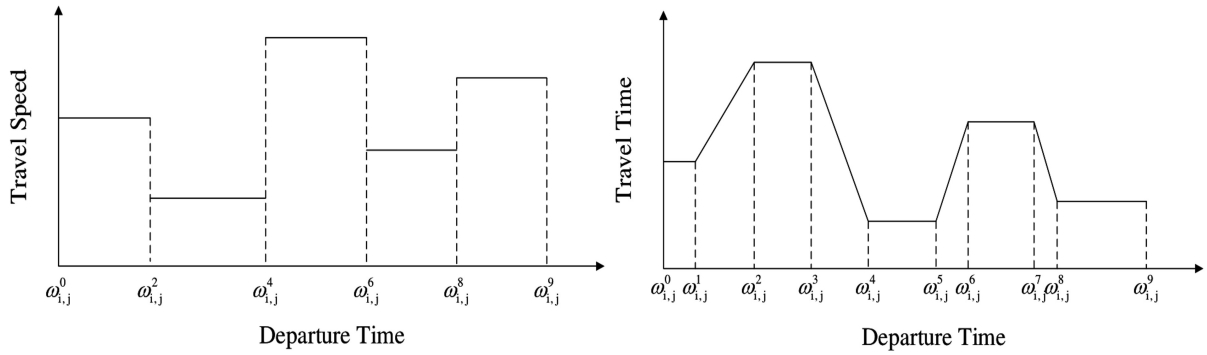
The studied TD-DARP is defined on a complete directed graph $G = (V, A)$, where $V = \{0, 2n+1\} \cup P \cup D$ is the set of nodes and A is the sets of arcs. Node 0 represents the starting depot while node $2n+1$ denotes the ending depot. The sets $P = \{1, ..., n\}$ and $D = \{n+1, ..., 2n\}$ comprise the pickup nodes and delivery nodes, respectively. The set of arcs A is defined as $A = \{(i, j) : i, j \in V, i \neq j, i \neq 2n+1, j \neq 0$, and $j \neq i-n$ for $i \in D\}$, where arcs from the delivery node to the pickup node of the same request are explicitly excluded. A request i is associated with a pickup node $i \in P$ and a delivery node $n+i \in D$ with a given maximum ride time L_i to ensure that the start service time at the delivery node $n+i$ must not exceed L_i time units after the end service time at the pickup node i. Each request i has d_i passengers requiring to be picked up from the pickup node i and dropped at the delivery node $n+i$. The demands of pickup node i and its delivery node $n+i$ are indicated by the positive demand d_i and the negative demand $d_{n+i} = -d_i$. Additionally, each node $i \in V$ has a non-negative service time duration s_i, a time window $[e_i, l_i]$ within which the service must be started, and a reward r_i for being served. Without loss of generality, we assume that $d_0 = d_{2n+1} = s_0 = s_{2n+1} = 0$. A homogeneous fleet K of vehicles is located at the origin depot 0. Each vehicle $k \in K$ is associated with the identical loading capacity C and the maximum route duration L. $c_{i,j}$ represents the routing cost from node i to node j. η denotes the cost per unit time. Figure 1 illustrates a TD-DARP solution with 7 requests and 2 vehicles. The sequence of vehicle 1's service (gray) is $\{0, 1^+, 3^+, 3^-, 2^+, 1^-, 2^-, 4^+, 4^-, 2n+1\}$, while the visiting order of vehicle 2 (green) is $\{0, 5^+, 6^+, 5^-, 6^-, 7^+, 7^-, 2n+1\}$.

Figure 1. An example of the TD-DARP solution



To account for time-dependent travel times, we define $\tau_{i,j}(t)$ as the travel time from node i to j given a departure time from node i at time t. Following Ichoua et al. (2003), the speed profile divides a whole planning horizon into several time zones, each of which is associated with a constant speed. As illustrated in Figure 2 (left), the speed profile is represented by a step-wise function, which shows the expected constant speed at a given point in time. Points $\{\omega^2_{i,j}, \omega^4_{i,j}, \omega^6_{i,j}, \omega^8_{i,j}, \omega^9_{i,j}\}$, referred to as *speed breakpoints*, denote the time points where the vehicle speed changes. The speed profile is commonly converted into a continuous and piecewise linear travel time function $\tau_{i,j}(t)$ that satisfies the *first-in-first-out* (FIFO) property, where $t_1 + \tau_{i,j}(t_1) \leq t_2 + \tau_{i,j}(t_2)$, if $t_1 \leq t_2$. Figure 2 (right) illustrates the function $\tau_{i,j}(t)$ formulated through a series of discrete breakpoints. These breakpoints encompass the previously mentioned *speed breakpoints* and the *travel time breakpoints*. The travel time breakpoints $\{\omega^1_{i,j}, \omega^3_{i,j}, \omega^5_{i,j}, \omega^7_{i,j}\}$ are determined as the start time from node i to arrive at node j exactly at a speed breakpoint (e.g., $\omega^1_{i,j}$ is the start time at node i to arrive at node j at time $\omega^2_{i,j}$ with an unchanged speed). We use $\Omega_{i,j}$ to denote the set of time zones, in which a time zone $\Omega^m_{i,j} \in \Omega_{i,j}$ is defined by two breakpoints, e.g. $\Omega^m_{i,j} = [\omega_m, \omega_{m+1}]$.

Figure 2. An example of travel speed function (left) and travel time function (right) from i to j



Note that in a time-dependent network, travel time does not necessarily satisfy the triangle inequality. In situations where the direct link between two nodes i and j is heavily congested, it is possible to reach j more quickly via an indirect route. Specifically, this occurs when $\tau_{i,j}(t_i) \geq \tau_{i,k}(t_i) + \tau_{k,j}(t_k)$. This characteristic implies that the shortest-distance path is not always the fastest one, and therefore minimizing distance and minimizing travel duration correspond to two distinct optimization objectives. In our approach, we do not explicitly compute fastest paths; instead, time-dependent travel times are directly considered during route evaluation, and the search process is guided by the selected optimization objective.

Let $f_{i,j}(t) = t + \tau_{i,j}(t)$ be the arrival time at node j given a departure time t at node i. In addition, its reverse function $f^{-1}_{i,j}(t)$ represents the departure time from node i if the vehicle arrives node j at time t. Given the inherent properties of the function $\tau_{i,j}(t)$ and the FIFO property, the

following lemma holds.

Lemma 1. *For each pair of nodes i and j, both functions $f_{i,j}(t)$ and $f_{i,j}^{-1}(t)$ exhibit the following properties: no-decreasing, piecewise linear, and continuous.*

Since the travel time function $\tau_{i,j}(t)$ is piecewise linear and continuous by definition, and $f_{i,j}(t) = t + \tau_{i,j}(t)$ is obtained by adding a linear term to it, $f_{i,j}(t)$ inherits these properties. Furthermore, under the FIFO condition $t_1 + \tau_{i,j}(t_1) \leq t_2 + \tau_{i,j}(t_2)$ for $t_1 \leq t_2$, $f_{i,j}(t)$ is non-decreasing, which implies that its inverse $f_{i,j}^{-1}(t)$ is also non-decreasing, piecewise linear, and continuous.

We study a class of the time-dependent dial-a-ride problems by incorporating the setting of time-dependent travel times, as well as different features and objectives. Among these features, the extent to which customer requests must be satisfied plays a crucial role in distinguishing different TD-DARP variants. In certain applications, such as emergency rescue operations, it is imperative that all requests are fully served. In contrast, in congested or resource-limited environments such as peak-traffic scenarios, serving only a subset of requests may be acceptable in order to balance service quality and operational costs (Lera-Romero and Miranda-Bront, 2021; Zhao et al., 2022).

These variations in service requirements naturally lead to different optimization objectives. When full service is mandatory, one common objective focuses on minimizing the total travel distance, thereby reducing fuel consumption and greenhouse gas emissions. Alternatively, minimizing route duration can also be motivated by operational costs, as shorter working hours directly reduce labor-related costs. In many practical settings, such as logistics operations, driver wages are proportional to total working time or route duration, meaning that reducing travel time can lead to significant savings in personnel expenses. Moreover, under time-dependent conditions, the shortest path in distance is not necessarily the fastest in duration, making duration minimization a distinct and practically relevant objective. In scenarios where only partial service is acceptable, maximizing the revenue from the subset of served requests becomes particularly relevant, as the operator must prioritize which requests to accept while balancing economic efficiency. Based on these distinctions, we investigate three representative variants of the time-dependent dial-a-ride problem in this study:

- **Problem P1** involves fulfilling all requests with the objective of minimizing the total traveling cost.
- **Problem P2** also requires fulfilling all requests, but the objective is to minimize the total route duration.
- **Problem P3** selects a subset of requests for service, aiming to maximize the total profit.

2.2. Problem Formulation

We formulate a general mixed integer linear programming (MILP) model for three time-dependent dial-a-ride problems. Five sets of decision variables are introduced: $\boldsymbol{x} = \{x_{i,j}^{k,m} | i,j \in V, k \in K, m \in \{1,...,|\Omega_{i,j}|\}\}, \boldsymbol{y} = \{y_i^k | i \in V, k \in K\}, \boldsymbol{z} = \{z_i | i \in V\}, \mathbf{t} = \{\mathrm{t}_{i,j}^{k,m} | i,j \in V, k \in K, m \in \{1,...,|\Omega_{i,j}|\}\}, \boldsymbol{t} = \{t_i^k | i \in V, k \in K\}$, where $x_{i,j}^{k,m}$ is a binary variable such that $x_{i,j}^{k,m} = 1$ if vehicle k travels from node i to node j by departing from i in time zone $\Omega_{i,j}^m$, and 0 otherwise; y_i^k is a binary variable such that $y_i^k = 1$ if vehicle k visits node i, and 0 otherwise; z_i is the load of vehicle when leaving node i; $\mathrm{t}_{i,j}^{k,m}$ denotes the departure time that vehicle k travels from node i to node j during time zone $\Omega_{i,j}^m$; t_i^k represents the departure time that vehicle k travels from node i. Note that for the ending depot $2n+1$, t_{2n+1}^k represents the arrival time of vehicle k, since no departure or service occurs after reaching the ending depot. Table 1 summarizes these notations.

Table 1
Notation for definition of the TD-DARP

Sets and parameters	
V	Node set
$0(2n+1)$	Starting (ending) depot
P	Set of pickup nodes for request, $P = \{1,...,n\}$
D	Set of delivery nodes for request, $D = \{n+1,...,2n\}$
A	Set of routing arcs
K	Set of vehicles
$\Omega_{i,j}$	Set of time zone of arc $(i,j) \in A$
C	Capacity of vehicles
L_i	Maximum ride time of request $i \in P$
L	Maximum route duration
η	Cost per unit time
d_i	Demand of node $i \in V$
s_i	Non-negative service time duration of node $i \in V$
r_i	Reward for serving node $i \in V$
$[e_i, l_i]$	Time window of node $i \in V$ during which the service must be started
$c_{i,j}$	Routing cost of arc $(i,j) \in A$
$\tau_{i,j}(t)$	Travel time of arc $(i,j) \in A$ given a departure time t at node i
$f_{i,j}(t)$	Arrival time at node j along arc $(i,j) \in A$ given a departure time t at node i
$f_{i,j}^{-1}(t)$	Departure time from node i when a vehicle arrives node j at time t

2.2.1. The Objectives of the TD-DARPs

Each of the three TD-DARPs focuses on a different objective. Problem P1 is formulated with the objective function given in Equation (1) to minimize the total traveling cost. Problem P2 is designed with the objective function of minimizing the total route duration, as given in Equation

(2). For problem P3, the objective function in Equation (3) is to maximize the total profit, which is denoted as the reward associated with the served requests minus the cost incurred by the route duration.

$$[\text{P1}] \quad \min \sum_{(i,j)\in A} \sum_{k\in K} \sum_{m\in\Omega_{i,j}} c_{i,j} x_{i,j}^{k,m} \tag{1}$$

$$[\text{P2}] \quad \min \sum_{k\in K} (t_{2n+1}^{k} - t_{0}^{k}) \tag{2}$$

$$[\text{P3}] \quad \max \sum_{k\in K} \sum_{i\in P} r_i y_i^k - \eta \sum_{k\in K} (t_{2n+1}^{k} - t_{0}^{k}) \tag{3}$$

2.2.2. Pickup and Delivery Services Constraints

To optimize vehicle scheduling for pickup and delivery, we consider capacity, pairing and classic constraints in the VRP. The relevant constraints are as follows:

$$\sum_{k\in K} y_i^k = 1, \forall i \in P \cup D \tag{4}$$

$$\sum_{j\in P} \sum_{m\in\Omega_{0,j}} x_{0,j}^{k,m} = y_0^k = 1, \forall k \in K \tag{5}$$

$$\sum_{i\in D} \sum_{m\in\Omega_{i,2n+1}} x_{i,2n+1}^{k,m} = y_{2n+1}^k = 1, \forall k \in K \tag{6}$$

$$\sum_{i\in V\setminus\{2n+1\}} \sum_{m\in\Omega_{i,\mathfrak{h}}} x_{i,\mathfrak{h}}^{k,m} - \sum_{j\in V\setminus\{0\}} \sum_{m\in\Omega_{\mathfrak{h},j}} x_{\mathfrak{h},j}^{k,m} = 0, \forall \mathfrak{h} \in P \cup D, k \in K \tag{7}$$

$$\sum_{i\in V\setminus\{2n+1\}} \sum_{m\in\Omega_{i,j}} x_{i,j}^{k,m} = y_j^k, \forall j \in P \cup D, k \in K \tag{8}$$

$$\sum_{j\in V\setminus\{0\}} \sum_{m\in\Omega_{i,j}} x_{i,j}^{k,m} - \sum_{j\in V\setminus\{0\}} \sum_{m\in\Omega_{n+i,j}} x_{n+i,j}^{k,m} = 0, \forall i \in P, k \in K \tag{9}$$

$$z_i + d_j \leq z_j + (C + d_j)(1 - \sum_{m\in\Omega_{i,j}} x_{i,j}^{k,m}), \forall i, j \in V, k \in K \tag{10}$$

$$max\{0, d_i\} \leq z_i \leq min\{C, C + d_i\}, \forall i \in V \tag{11}$$

Constraints (4) ensure that all requests are served. Constraints (5) and (6) mean that a vehicle can only start from and end at the depot once, namely, the vehicle cannot return to the depot during the serving process. Constraints (7) are the classical flow conservation constraints, which make sure that the in-degree and the out-degree of a node in a route must be equal. Constraints (8) guarantee that the pickup node and delivery node are visited when the corresponding request is served. Constraints (9) force that the pick-up node and the corresponding delivery node are served by the same vehicle. Constraints (10) and (11) are the capacity constraints, which ensure that the loading capacity is respected.

2.2.3. Temporal Constraints

Temporal constraints comprise the precedence, time windows, ride time and maximum route duration constraints. In time-dependent networks, these constraints become more complicated to address. Figure 3 illustrates an example. There are four feasible breakpoints when the vehicle departs from node i. These breakpoints represent the points where the travel-time function changes its slope and are used for evaluating time-dependent feasibility, but they do not restrict the vehicle to depart exactly at those times. According to the travel time function $\tau_{i,j}(t)$, one breakpoint leads to an arrival time at node j that exceeds its time window and is therefore discarded. When traveling from node j to node $n+j$, a similar situation occurs, where one of the remaining breakpoints exceeds the time window of node $n+j$ and is removed. Consequently, only two breakpoints remain feasible upon arrival at node $n+j$. Finally, considering the ride time constraint associated with request i, only one breakpoint ultimately proves feasible when the vehicle reaches node $n+i$. The specific constraints are as follows:

$$t_i^k = \sum_{j \in V \setminus \{0\}} \sum_{m \in \Omega_{i,j}} \mathrm{t}_{i,j}^{k,m}, \forall i \in V \setminus \{2n+1\}, k \in K \tag{12}$$

$$t_i^k + \tau_{i,j}(t_i^k) + s_j \leq t_j^k + M(1 - x_{i,j}^{k,m}), \forall i, j \in P \cup D, m \in \Omega_{i,j}, k \in K \tag{13}$$

$$t_{n+i}^k \geq t_i^k, \forall i \in P, k \in K \tag{14}$$

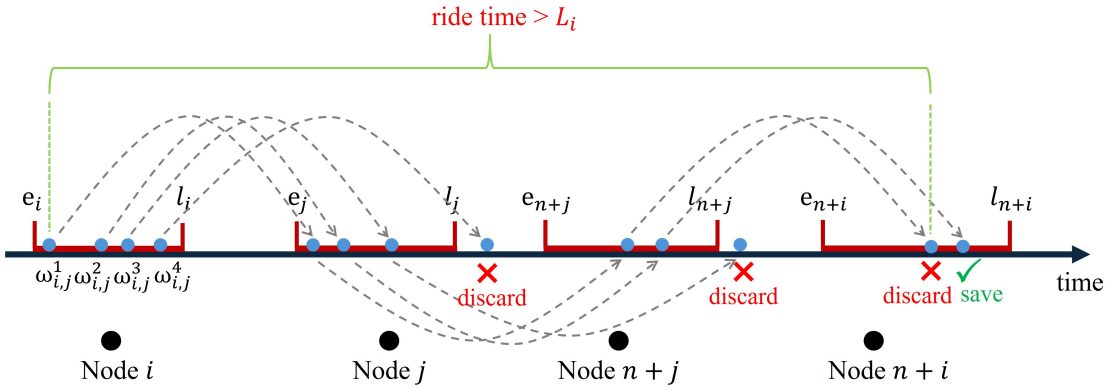
$$\omega_m x_{i,j}^{k,m} \leq \mathrm{t}_{i,j}^{k,m} \leq \omega_{m+1} x_{i,j}^{k,m}, \forall i, j \in V, m \in \Omega_{i,j}, k \in K \tag{15}$$

$$(e_i + s_i) y_i^k \leq t_i^k \leq (l_i + s_i) y_i^k, \forall i \in V, k \in K \tag{16}$$

$$(t_{n+i}^k - s_{n+i}) - t_i^k \leq L_i, \forall i \in P, k \in K \tag{17}$$

$$t_{2n+1}^k - t_0^k \leq L, \forall k \in K \tag{18}$$

Figure 3. Illustration of temporal constraints in time-dependent network



Constraints (12) define the departure time of vehicle k from node i. Constraints (13) are the continuity constraints. The ready time that vehicle k can departure from node j is not less

than the departure time of vehicle k from node i plus the travel time and service time of node j. The M parameter is equal to the upper limit of the time windows associated with node $2n+1$. Although the travel time function $\tau_{i,j}(t)$ is time-dependent and piecewise linear, all breakpoints, slopes, and intercepts are precomputed prior to optimization and treated as known parameters. Specifically, for each arc (i,j) and each time zone $m \in \Omega_{i,j}$, the travel time can be expressed as $\tau_{i,j}(t_i^k) = SL_{i,j}^m t_i^k + IT_{i,j}^m$, where $SL_{i,j}^m$ and $IT_{i,j}^m$ denote the slope and intercept of the m-th segment, respectively, and are known constants. Therefore, Constraints (13) remain linear. Constraints (14) are the precedence constraints, which affirm that the pick-up node i is visited before the corresponding delivery node $n+i$. Constraints (15) indicate that the departure time for a vehicle from a location must lie in exactly one time zone. Constraints (16) are the time windows constraints, which affirm that the start service time of each node is within the given time window. Constraints (17) are the ride time constraints, which guarantee that the start service time of the delivery node $n+i$ is within the end service time of corresponding pick-up node i plus the maximum ride time L_i. Constraints (18) are the maximum route duration constraints. Note that there is no restriction on the waiting time, allowing the vehicle to wait and delay the start service time voluntarily whenever it arrives at node i. This flexibility is crucial for the feasibility of routes under the maximum ride time constraint.

As a result, we formulate the problem P1 with the objective function (1), subject to constraints (4)-(18). Problem P2 is formulated with the objective function (2), subject to constraints (4)-(18). Problem P3 is defined with the objective function (3), subject to constraints (5)-(18) and the modified constraint (19). In this problem, serving all requests is not mandatory, and therefore constraint (4) used in P1 and P2 is replaced by the following constraint:

$$\sum_{k \in K} y_i^k \leq 1, \forall i \in P \cup D \tag{19}$$

which ensures that each request can be served at most once, preventing multiple vehicles from serving the same customer and avoiding duplicate revenue collection.

3. Feasibility Test

The feasibility test procedure constitutes a fundamental component in the proposed algorithm and is frequently invoked throughout the search process. As such, it plays a critical role in determining the overall efficiency of the algorithm. As discussed in the introduction, existing feasibility test procedures for the DARP can be broadly classified into three main categories, namely eight-step evaluation schemes, shortest path formulations on interval graphs, and preprocessing-based approaches. Representative studies include the revised eight-step procedure of Sohrabi et al. (2024),

the expression of the feasibility test as a shortest path problem by Firat and Woeginger (2011), and the constant-time feasibility procedure proposed by Gschwind and Drexl (2019). These methods represent some of the most efficient feasibility test procedures developed for the classical DARP with static travel times. Their efficiency relies on the use of cumulative quantities along the route (e.g., forward time slacks, or cumulative travel times) that can be consistently maintained along a fixed pickup–delivery sequence under static travel times.

In the time-dependent DARP, however, travel times vary with departure times, affecting not only direct connections but also the propagation of feasibility along the entire route. Moreover, the computation of these cumulative quantities depends not only on the earliest or latest service times but also on the feasible breakpoints between consecutive nodes. These breakpoints are determined jointly by the travel time function, time window restrictions, and ride time constraints, forming segmented feasibility regions that invalidate the fixed-time assumptions underlying constant or linear time feasibility checks. Here, "segmented" refers to the piecewise linear structure of the functions. Consequently, although these procedures are efficient and effective in the static DARP, their assumptions make them difficult to extend directly to the time-dependent case.

To the best of our knowledge, the only feasibility test explicitly developed for the TD-DARP is the method proposed by Zhao et al. (2022). However, this method incurs a relatively high computational complexity, leading to costly neighborhood evaluations in local search algorithms. To overcome these limitations of existing feasibility tests, this section presents a novel and efficient feasibility testing method specifically tailored for integration into the TD-DARP algorithm.

3.1. The Latest Start Service Time Function

As previously described, a feasible TD-DARP route must satisfy the following constraints: capacity, pairing and precedence, time windows, ride time, maximum route duration. Each time a request is inserted into the route, the insertion operation should be evaluated to determine whether it violates these constraints. Due to the independence of vehicle capacity constraints from temporal constraints, the procedure for checking violations of the capacity constraints in the TD-DARP is analogous to that in the DARP. Ensuring the pairing and precedence constraints during the insertion of a request is straightforward. Therefore, the challenge in feasibility testing lies in efficiently evaluating whether the insertion operation satisfies the temporal constraints. To address this challenge, we first introduce two theorems that provide a foundation for our feasibility testing approach.

Let γ_{v_i} denote the feasible start service time of node v_i, with $\gamma_{v_i}^-$ and $\gamma_{v_i}^+$ representing its earliest and latest feasible start service times, respectively. Time schedule of TD-DARP route R is represented by $T_R = (\gamma_{v_1}, ..., \gamma_{v_r}) \in \mathscr{T}_R$, where $\mathscr{T}_R$ denotes the set of all feasible time schedules

associated with route R. Specifically, $T_R(t_{v_j})$ represents the time schedule in which all nodes of route R start service at their corresponding feasible times when the start service time γ_{v_j} of node v_j is not later than t_{v_j}. Accordingly, we define $\mathscr{T}_R(t_{v_j})$ as the set of all such feasible time schedules. The cumulative travel times between node v_i and v_j at a given departure time t is denoted as $\Gamma_{v_i,v_j}(t)$, representing the total travel time along the sequence of arcs from node v_i to node v_j within the same route. The latest start service time function $\gamma^+_{v_j}(t_{v_i})$ represents the latest feasible start time to serve node v_j if the start service time of node v_i is not later than t_{v_i}, where $\gamma_{v_j}(t_{v_i})$ denotes the start service time function under a given t_{v_i}, and $t_{v_i} \in [\gamma^-_{v_i}, \gamma^+_{v_i}], \forall i, j \in R, i \leq j$.

Theorem 1. *For any TD-DARP feasible route $R = (v_1, ..., v_r)$, let $\gamma^+_{v_i}(t_{v_j}) = \max\limits_{T_R=(\gamma_{v_1},...,\gamma_{v_r})\in\mathscr{T}_R(t_{v_j})} \{\gamma_{v_i}\}$ be the latest start service time of each node v_i if the start service time γ_{v_j} of node v_j is not later than t_{v_j}, then $T^+_R(t_{v_j}) = (\gamma^+_{v_1}(t_{v_j}), ..., \gamma^+_{v_r}(t_{v_j}))$ represents the time schedule in which all nodes of route R start service at their latest feasible times given t_{v_j}, and $T^+_R(t_{v_j}) \in \mathscr{T}_R(t_{v_j})$.*

Proof. A feasible schedule $T_R(t_{v_j}) = (\gamma_{v_1}(t_{v_j}), ..., \gamma_{v_r}(t_{v_j}))$ must satisfy the temporal constraints:

$$\gamma_{v_i}(t_{v_j}) \in [e_{v_i}, l_{v_i}], \forall i, j \in R \tag{20}$$

$$\gamma_{v_i}(t_{v_j}) + s_{v_i} + \tau_{v_i,v_{i+1}}(\gamma_{v_i}(t_{v_j}) + s_{v_i}) \leq \gamma_{v_{i+1}}(t_{v_j}), \forall i, j \in R \tag{21}$$

$$\gamma_{v_i}(t_{v_j}) + s_{v_i} + L_{v_i} \geq \gamma_{v_i+n}(t_{v_j}), \forall i \in R \cap P, j \in R \tag{22}$$

According to the definition of $\gamma^+_{v_i}(t_{v_j})$, for each moment $\gamma^+_{v_i}(t_{v_j}), i = 1, ..., r-1$, there exits a feasible time schedule $T'_R(t_{v_j}) = (\gamma_{v_1}(t_{v_j}), ..., \gamma^+_{v_i}(t_{v_j}), ..., \gamma_{v_r}(t_{v_j})) \in \mathscr{T}_R(t_{v_j})$. We have $\gamma_{v_i}(t_{v_j}) \leq \gamma^+_{v_i}(t_{v_j}), i = 1, ..., r-1$. Since $T'_R(t_{v_j}) \in \mathscr{T}_R(t_{v_j})$, it is obvious that $\gamma^+_{v_i}(t_{v_j}) \in [e_{v_i}, l_{v_i}]$. For each node $v_i, i = 1, ..., r-1$, because $\gamma^+_{v_i}(t_{v_j}) + s_{v_i} + \tau_{v_i,v_{i+1}}(\gamma^+_{v_i}(t_{v_j}) + s_{v_i}) \leq \gamma_{v_{i+1}}(t_{v_j})$ and $\gamma_{v_{i+1}}(t_{v_j}) \leq \gamma^+_{v_{i+1}}(t_{v_j})$, then $\gamma^+_{v_i}(t_{v_j}) + s_{v_i} + \tau_{v_i,v_{i+1}}(\gamma^+_{v_i}(t_{v_j}) + s_{v_i}) \leq \gamma^+_{v_{i+1}}(t_{v_j})$ holds. For each pair of node $v_i - n$ and v_i, there exists $\gamma_{v_i-n}(t_{v_j}) + s_{v_i-n} + L_{v_i-n} \geq \gamma^+_{v_i}(t_{v_j})$ and $\gamma^+_{v_i-n}(t_{v_j}) \geq \gamma_{v_i-n}(t_{v_j})$, hence $\gamma^+_{v_i-n}(t_{v_j}) + s_{v_i-n} + L_{v_i-n} \geq \gamma^+_{v_i}(t_{v_j})$ holds. Therefore, Theorem 1 holds. □

Theorem 2. *For any TD-DARP feasible route $R = (v_1, ..., v_r)$, the latest ready time $\delta_{v_i}(t_{v_j})$ on node v_i when node v_j departs at time t_{v_j} is a non-decreasing and piecewise linear function, which is defined in a certain continuous closed interval and has a group of jump discontinuities.*

Proof. The latest ready time $\delta_{v_i}(t_{v_j})$ denotes the latest feasible time at which node v_i becomes ready to start service, given that node v_j departs at t_{v_j}, $\forall i, j \in R, i \leq j$. It is evident that the theorem holds when $r = 1$, in which $R = (v_1), \delta_{v_i}(t_{v_j}) = t_{v_1} - s_{v_1}$. Assume that the theorem holds for the route $R = (v_1, v_2, ..., v_j)$ when $r \geq 1$. In other words, for each vertex $v_i, \forall i = 1, ..., r, \delta_{v_i}(t_{v_j})$ is non-decreasing and piecewise linear, associated with a group of q_i jump discontinuities denoted by

$(g_{i,1}, g_{i,2}, ..., g_{i,q_i})$, and defined in a closed interval $[a_{v_r}, b_{v_r}]$. A new route $R' = (v_1, v_2, ..., v_j, v_{j+1})$ is derived by extending original route R with an additional vertex $j+1$. To facilitate the subsequent derivation, we use $\tau^{-1}_{v_i,v_j}(t)$ to denote the travel time inverse function, yielding the travel time of edge (v_i, v_j) given the arrival time t of node v_j. $\phi^+_{v_i}(t_{v_j})$ is defined to represent the latest service time on node v_i at a given start service time t_{v_j} of node $v_j, \forall i, j \in R, i \leq j$. Below, we prove that the theorem also holds for route R'. We have the following two cases:

Case 1. If $v_{j+1} \in P$, the functions $\phi^+_{v_i}(t_{v_j})$ can be computed using Equation (23)

$$\phi^+_{v_i}(t_{v_{j+1}}) = \begin{cases} \delta_{v_i}(t_{v_{j+1}} - \tau^{-1}_{v_j,v_{j+1}}(t_{v_{j+1}})) & if\ i = 1, ..., j \\ t_{v_{j+1}} & if\ i = j+1 \end{cases} \tag{23}$$

where $t_{v_{j+1}} \in [max\{\gamma^-_{v_j} + s_{v_j} + \tau_{v_j,v_{j+1}}(\gamma^-_{v_j} + s_{v_j}), e_{v_{j+1}}\}, l_{v_{j+1}}]$. Function $\delta_{v_i}(t_{v_j})$ can be computed by Equation (24)

$$\delta_{v_i}(t_{v_{j+1}}) = \begin{cases} \phi^+_{v_i}(t_{v_{j+1}} - s_{v_{j+1}}) & if\ t_{v_{j+1}} \in [max\{\gamma^-_{v_j} + s_{v_j} + \tau_{v_j,v_{j+1}}(\gamma^-_{v_j} + s_{v_j}), e_{v_{j+1}}\} + s_{v_{j+1}}, l_{v_{j+1}} + s_{v_{j+1}}] \\ \phi^+_{v_i}(l_{v_{j+1}}) & if\ t_{v_{j+1}} \in [l_{v_{j+1}} + s_{v_{j+1}}, T] \end{cases} \tag{24}$$

$$[a_{v_{j+1}}, b_{v_{j+1}}] = [max\{\gamma^-_{v_j} + s_{v_j} + \tau_{v_j,v_{j+1}}(\gamma^-_{v_j} + s_{v_j}), e_{v_{j+1}}\} + s_{v_{j+1}}, T] \tag{25}$$

As mentioned before, both $f_{v_j,v_{j+1}}(t_{v_j}) = t_{v_j} + \tau_{v_j,v_{j+1}}(t_{v_j})$ and $f^{-1}_{v_j,v_{j+1}}(t_{v_{j+1}}) = t_{v_{j+1}} - \tau^{-1}_{v_j,v_{j+1}}(t_{v_{j+1}})$ are non-decreasing, piecewise linear continuous, and functions due to the FIFO property. In terms of Equations (23), (24) and (25), functions $\delta_{v_i}(t_{v_j}), \forall i \in 1, ..., r$, defined in $[a_{v_{j+1}}, b_{v_{j+1}}]$ are non-decreasing and piecewise linear. Furthermore, they all have a group of original jump discontinuities $(g_{i,1}, g_{i,2}, ..., g_{i,q_i})$.

Case 2. If $v_{j+1} \in D$, then $v_{j+1} - n \in P$ is the corresponding pickup node, which must satisfy its maximum ride time constraint, $t_{v_{j+1}} - \phi^+_{v_{j+1-n}}(t_{v_{j+1}}) - s_{v_{j+1-n}} \leq L_{v_{j+1-n}}, \forall t_{v_{j+1}} \in [max\{\gamma^-_{v_j} + s_{v_j} + \tau_{v_j,v_{j+1}}(\gamma^-_{v_j} + s_{v_j}), e_{v_{j+1}}\}, l_{v_{j+1}}]$. Consequently, a set of feasible domains H of $t_{v_{j+1}}$, $\{[\mu^h_{v_{j+1}}, \omega^h_{v_{j+1}}]\}_{h \in H}$, can be determined. Based on this, the functions $\phi^+_{v_i}(t_{v_j})$ can be obtained directly from Equation (23), and the functions $\delta_{v_i}(t_{v_j})$ can be calculated by Equation (26).

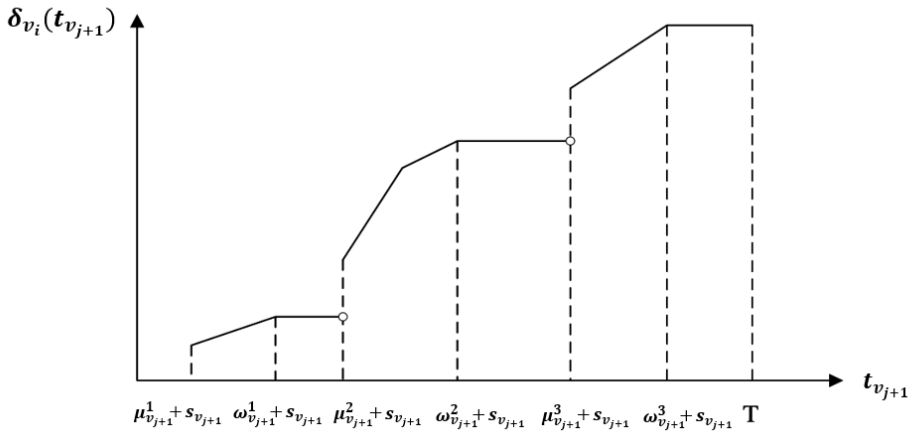
$$\delta_{v_i}(t_{v_{j+1}}) = \begin{cases} \phi^+_{v_i}(t_{v_{j+1}} - s_{v_{j+1}}) & if\ t_{v_{j+1}} \in [\mu^h_{v_{j+1}} + s_{v_{j+1}}, \omega^h_{v_{j+1}} + s_{v_{j+1}}] \\ \phi^+_{v_i}(\omega^h_{v_{j+1}}) & if\ t_{v_{j+1}} \in (\omega^h_{v_{j+1}} + s_{v_{j+1}}, \mu^{h+1}_{v_{j+1}} + s_{v_{j+1}}) \\ \phi^+_{v_i}(\omega^{|H|}_{v_{j+1}}) & if\ t_{v_{j+1}} \in [\omega^{|H|}_{v_{j+1}} + s_{v_{j+1}}, T] \end{cases} \tag{26}$$

where $t_{v_{j+1}} \in [a_{v_{j+1}}, b_{v_{j+1}}]$ such that $a_{v_{j+1}} = \mu^1_{v_{j+1}} + s_{v_{j+1}}$ and $b_{v_{j+1}} = T$. According to Equations (23) and (26), functions $\delta_{v_i}(t_{v_j}), \forall i \in 1, ..., r$, defined in domain $[a_{v_{j+1}}, b_{v_{j+1}}]$ are non-decreasing and piecewise linear function with at least $|H|$ jump discontinuities.

Therefore, Theorem 2 holds. □

Figure 4 shows an example of the latest ready time function $\delta_{v_i}(t_{v_{j+1}})$ with three jump discontinuities in domain $[\mu^1_{v_{j+1}} + s_{v_{j+1}}, T]$. In the subsequent sections, we divide the feasibility test procedure for the insertion of request i into two phases. The primary phase involves the evaluation of inserting the pickup node v_i, while the secondary phase focuses on evaluating the insertion of the delivery node $v_i + n$.

Figure 4. An example of the latest ready time function $\delta_{v_i}(t_{v_{j+1}})$



3.2. Evaluation of Pickup Insertion

Algorithm 1 provides a detailed description of the process for inserting a pickup node v_i after node v_k in the original TD-DARP route $R = (v_1, ..., v_k, ..., v_r)$, where $k \in \{1, ..., r-1\}$. Given an as-early-as-possible schedule $T = (\gamma^-_{v_1}, ..., \gamma^-_{v_r})$ for route R, the algorithm will detect whether the newly generated route $R' = (v_1, ..., v_k, v_i, v_{k+1}, ..., v_r)$ has a feasible schedule $T' = (\gamma'_{v_1}, ..., \gamma'_{v_k}, \gamma'_{v_i}, \gamma'_{v_{k+1}}, ..., \gamma'_{v_r})$.

The basic idea is to evaluate whether the time increment resulting from node insertion falls within an acceptable range through the latest start service time function. Firstly, the pickup node v_i is inserted at the earliest possible time γ'_{v_i} (Line 1). If γ'_{v_i} exceeds its latest start time window, this insertion is infeasible (Lines 2 and 3). Next, we determine the start service time $\overline{\gamma}_{v_i}(\gamma_{v_k})$ at node v_i, which is a linear function of γ_{v_k} (Line 4). This computation relies on the earliest service time function at the current node v_k, denoted by $\gamma^-_{v_k}(\gamma_{v_k})$, which is precomputed during the preprocessing stage described in Section 3.4 and serves as the initialization for the subsequent propagation. The service time function $\overline{\gamma}_{v_{k+1}}(\gamma_{v_k})$ of the successor node v_{k+1} is then recursively

calculated through $\overline{\gamma}_{v_i}(\gamma_{v_k})$ (Line 5). As the insertion of node v_i augments the service time at node v_{k+1}, a feasibility check is required. According to Theorem 1, this check can be performed through the latest start service time function $\gamma^+_{v_{k+1}}(\gamma_{v_k})$. If there exists an interval I for which function $\gamma^+_{v_{k+1}}(\gamma_{v_k})$ exceeds function $\overline{\gamma}_{v_{k+1}}(\gamma_{v_k})$, it indicates that the time increment resulting from the insertion of node v_i lies within the acceptable range of the service time of node v_{k+1} (Lines 6–10).

Algorithm 1 Evaluate insertion of pickup

Input: node v_k, pickup node v_i

1: $\gamma'_{v_i} = max\{e_{v_i}, \gamma^-_{v_k} + s_{v_k} + \tau_{v_k,v_i}(\gamma^-_{v_k} + s_{v_k})\}$
2: **if** $\gamma'_{v_i} > l_{v_i}$ **then**
3: **return** false
4: $\overline{\gamma}_{v_i}(\gamma_{v_k}) = max\{e_{v_i}, \gamma^-_{v_k}(\gamma_{v_k}) + s_{v_k} + \tau_{v_k,v_i}(\gamma^-_{v_k}(\gamma_{v_k}) + s_{v_k})\}$
5: $\overline{\gamma}_{v_{k+1}}(\gamma_{v_k}) = max\{\gamma^-_{v_{k+1}}, \overline{\gamma}_{v_i}(\gamma_{v_k}) + s_{v_i} + \tau_{v_i,v_{k+1}}(\overline{\gamma}_i(\gamma_{v_k}) + s_{v_i})\}$
6: $I = \{\gamma_{v_k} | \gamma^+_{v_{k+1}}(\gamma_{v_k}) > \overline{\gamma}_{v_{k+1}}(\gamma_{v_k}), \forall \gamma_{v_k} \in [\gamma^-_{v_k}, \gamma^+_{v_k}]\}$
7: **if** $I == \varnothing$ **then**
8: **return** false
9: **else**
10: **return** true

3.3. Evaluation of Delivery Insertion

Algorithm 2 describes the process for inserting a delivery node $v_i + n$ after node v_j in the original TD-DARP route $R = (v_1, ..., v_k, ..., v_j, ..., v_r)$. Given an as-early-as-possible schedule $T = (\gamma^-_{v_1}, ..., \gamma^-_{v_r})$, the algorithm will detect whether the newly generated route $R' = (v_1, ..., v_k, v_i, v_{k+1}..., v_j, v_i + n, v_{j+1}, ..., v_r)$ has a feasible schedule $T' = (\gamma'_{v_1}, ..., \gamma'_{v_k}, \gamma'_{v_i}, \gamma'_{v_{k+1}}, ..., \gamma'_{v_j}, \gamma'_{v_i+n}, \gamma'_{v_{j+1}}, ..., \gamma'_{v_r})$.

Firstly, we calculate the earliest service time for pickup node v_i and its delivery node $v_i + n$ (Lines 1–4). Note that nodes v_{k+1} and v_j are not necessarily consecutive in the route. Therefore, the cumulative travel times function $\Gamma_{v_{k+1},v_j}(t)$ is used in Line 3 to capture the total travel time along the subsequence from v_{k+1} to v_j, given a departure time from v_{k+1}. If γ'_{v_i+n} exceeds the latest start time window of node $v_i + n$, this insertion is infeasible (Lines 5 and 6). Otherwise, we recursively determine the start service time for nodes $v_i, v_{k+1}, v_j, v_i + n$ and v_{j+1} as functions of γ_{v_k} (Lines 7–11). The latest start service time function $\gamma^+_{v_{j+1}}(\gamma_{v_k})$ is utilized to identify the feasible interval I, in which $\gamma^+_{v_{j+1}}(\gamma_{v_k}) > \overline{\gamma}_{v_{j+1}}(\gamma_{v_k})$ is satisfied (Lines 12–15). Note that in the TD-DARP, the ride time constraints of a request restrict the service time for both the pickup node and delivery node. Thus, it is crucial to identify the specific feasible interval I' within I, which satisfies the constraint $\overline{\gamma}_{v_i+n}(\gamma_{v_k}) < \overline{\gamma}_{v_i}(\gamma_{v_k}) + L_{v_i}$ (Line 16). If the non-empty set I' is obtained, the insertion of the pickup and delivery nodes is deemed feasible (Line 20). It indicates that the

insertion operation has an acceptable impact on the original route schedule of service times, and complies with the ride time constraints of request. Specially, if the newly inserted pickup node v_i is served immediately followed by the corresponding delivery node v_i+n, where $k=j$, or if there is only one node intervening between the newly inserted pickup node v_i and the delivery node v_i+n, such that $k+1=j$, then only slight adjustments to the algorithm are required, with the overall framework remaining unchanged.

Algorithm 2 Evaluate insertion of delivery

Input: node v_k, v_j, pickup node v_i, delivery node v_i+n

$\gamma'_{v_i} = max\{e_{v_i}, \gamma^-_{v_k} + s_{v_k} + \tau_{v_k,v_i}(\gamma^-_{v_k} + s_{v_k})\}$

$\gamma'_{v_{k+1}} = max\{\gamma^-_{v_{k+1}}, \gamma'_{v_i} + s_{v_i} + \tau_{v_i,v_{k+1}}(\gamma'_{v_i} + s_{v_i})\}$

$\gamma'_{v_j} = max\{\gamma^-_{v_j}, \gamma'_{v_{k+1}} + s_{v_{k+1}} + \Gamma_{v_{k+1},v_j}(\gamma'_{v_{k+1}} + s_{v_{k+1}})\}$

$\gamma'_{v_i+n} = max\{e_{v_i+n}, \gamma'_{v_j} + s_{v_j} + \tau_{v_j,v_i+n}(\gamma'_{v_j} + s_{v_j})\}$

if $\gamma'_{v_i+n} > l_{v_i+n}$ **then**

  return false

$\overline{\gamma}_{v_i}(\gamma_{v_k}) = max\{e_{v_i}, \gamma^-_{v_k}(\gamma_{v_k}) + s_{v_k} + \tau_{v_k,v_i}(\gamma^-_{v_k}(\gamma_{v_k}) + s_{v_k})\}$

$\overline{\gamma}_{v_{k+1}}(\gamma_{v_k}) = max\{\gamma^-_{v_{k+1}}, \overline{\gamma}_{v_i}(\gamma_{v_k}) + s_{v_i} + \tau_{v_i,v_{k+1}}(\overline{\gamma}_{v_i}(\gamma_{v_k}) + s_{v_i})\}$

$\overline{\gamma}_{v_j}(\gamma_{v_k}) = max\{\gamma^-_{v_j}, \overline{\gamma}_{v_{k+1}}(\gamma_{v_k}) + s_{v_{k+1}} + \Gamma_{v_{k+1},v_j}(\overline{\gamma}_{v_{k+1}}(\gamma_{v_k}) + s_{v_{k+1}})\}$

$\overline{\gamma}_{v_i+n}(\gamma_{v_k}) = max\{e_{v_i+n}, \overline{\gamma}_{v_j}(\gamma_{v_k}) + s_{v_j} + \tau_{v_j,v_i+n}(\overline{\gamma}_{v_j}(\gamma_{v_k}) + s_{v_j})\}$

$\overline{\gamma}_{v_{j+1}}(\gamma_{v_k}) = max\{\gamma^-_{v_{j+1}}, \overline{\gamma}_{v_i+n}(\gamma_{v_k}) + s_{v_i+n} + \tau_{v_i+n,v_{j+1},}(\overline{\gamma}_{v_i+n}(\gamma_{v_k}) + s_{v_i+n})\}$

$I = \{\gamma_{v_k} | \gamma^+_{v_{j+1}}(\gamma_{v_k}) > \overline{\gamma}_{v_{j+1}}(\gamma_{v_k}), \forall \gamma_{v_k} \in [\gamma^-_{v_k}, \gamma^+_{v_k}]\}$

if $I == \varnothing$ **then**

  return false

else

  $I' = \{\gamma_{v_k} | \overline{\gamma}_{v_i+n}(\gamma_{v_k}) < \overline{\gamma}_{v_i}(\gamma_{v_k}) + L_{v_i}, \forall \gamma_{v_k} \in I\}$

  if $I' == \varnothing$ **then**

    return false

  else

    return true

3.4. Proof of the Linear Time Property

We now show that the feasibility test procedures outlined in Algorithms 1 and 2 have both linear time complexity $O(n_{seg})$, where n_{seg} denotes the number of intervals of the linear segmentation function. Obviously, all input data can be obtained in constant time $O(1)$, including time windows $[e_{v_i}, l_{v_i}]$, service duration s_{v_i}, and ride time L_{v_i}, $\forall v_i \in P \cup D$. The travel time function $\tau_{v_i,v_j}(t)$ can compute the travel time from node v_i to node v_j at a given departure time t in linear time $O(n_{seg})$. The earliest service time $\gamma^-_{v_i}$, the earliest service time function $\gamma^-_{v_i}(t)$ for node v_i with its own service time t, and the cumulative travel times $\Gamma_{v_i,v_j}(t)$ between node v_i and v_j at a given

departure time t, $\forall i, j \in R$, can be calculated in a preprocessing step. Finally, the preprocessing procedure yields the bound $\gamma^+_{v_j}(t_{v_i})$, thus facilitating the efficient computation of feasible domains I and I' in linear time $O(n_{seg})$. As a result, Theorem 3 holds.

Theorem 3. *For a feasible TD-DARP route $R = (v_1, ..., v_r)$, the feasibility of inserting a request can be tested in linear time $O(n_{seg})$.*

The preprocessing step requires the computation of three critical functions: the earliest start service time function $\gamma^-_{v_i}(t)$, the cumulative travel times function $\Gamma_{v_i,v_j}(t)$, and the latest start service time function $\gamma^+_{v_j}(t_{v_i})$. Equation (27) provides the formula for calculating $\gamma^-_{v_i}(t)$. The time complexity of the calculation is $O(|R|)$, with $|R|$ denoting the number of nodes served in route R.

$$\gamma^-_{v_i}(t) = t, t \in [\gamma^-_{v_i}, \gamma^+_{v_i}], \forall i \in R \tag{27}$$

The cumulative travel times function, denoted as $\Gamma_{v_i,v_j}(t)$, can be computed using Equation (28). The time complexity of the computation is $O(|R|^2 n_{seg})$, with n_{seg} denoting the number of segments of the piecewise function.

$$\Gamma_{v_i,v_j}(t) = \begin{cases} \Gamma_{v_i,v_{j-1}}(t) + s_{v_{j-1}} + \tau_{v_{j-1},v_j}(t + \Gamma_{v_i,v_{j-1}}(t) + s_{v_{j-1}}) & if\ j = i+1, ..., k \\ 0 & if\ j = i \end{cases} \tag{28}$$

The computation of the function $\gamma^+_{v_j}(t_{v_i})$ uses the function $\delta_{v_i}(t_{v_j})$. For a feasible route $R = (v_1, ..., v_r)$, the label-setting approach is applied to perform backward recursion at each node. Precisely, we consider the backward route $R_{backward} = (v_i, v_{i-1}, ..., v_r)$ and extend node v_{i+1} in front of the route, thereby generating a modified route $R'_{backward} = (v_{i+1}, v_i, v_{i-1}, ..., v_r)$. When $v_j = v_r$, the function $\gamma^+_{v_r}(t_{v_i}) = l_{v_r}$. If $v_i \in P, \forall j = i, i-1, ..., r$, the functions $\gamma^+_{v_j}(t_{v_i})$ can be computed using Equation (29), (30) and (31).

$$\gamma^+_{v_i}(t_{v_i}) = min\{\gamma^+_{v_{i-1}}(t_{v_i}) - \tau^{-1}_{v_i,v_{i-1}}(\gamma^+_{v_{i-1}}(t_{v_i})) - s_{v_i}, l_{v_i}\} \tag{29}$$

$$t_{v_i} = t_{v_{i+1}} + s_{v_{i+1}} + \tau_{v_{i+1},v_i}(t_{v_{i+1}} + s_{v_{i+1}}) \tag{30}$$

$$\begin{aligned} \gamma^+_{v_i}(t_{v_{i+1}}) = min\{&\gamma^+_{v_{i-1}}(t_{v_{i+1}} + s_{v_{i+1}} + \tau_{v_{i+1},v_i}(t_{v_{i+1}} + s_{v_{i+1}})) - \\ &\tau^{-1}_{v_i,v_{i-1}}(\gamma^+_{v_{i-1}}(t_{v_{i+1}} + s_{v_{i+1}} + \tau_{v_{i+1},v_i}(t_{v_{i+1}} + s_{v_{i+1}}))) - s_{v_i}, l_{v_i}\} \end{aligned} \tag{31}$$

If $v_i \in D$, then the corresponding pickup node $v_i - n \in P$ must comply with its maximum ride time constraint, $\gamma^+_{v_i}(t_{v_{i+1}}) - \delta_{v_i-n}(t_{v_{i+1}} + s_{v_{i+1}}) - s_{v_i-n} \leq L_{v_i-n}, \forall t_{v_{i+1}} \in [max\{\gamma^-_{v_i} - \tau^{-1}_{v_{i+1},v_i}(\gamma^-_{v_i}) -$

$s_{v_{i+1}}, e_{v_{i+1}}\}, l_{v_{i+1}}]$. The function $\gamma^+_{v_j}(t_{v_i})$ can be computed using Equation (32) in $O(|R|^2 n_{seg})$ computation time. According to these equations and Theorem 2, function $\gamma^+_{v_j}(t_{v_i})$ is also a non-decreasing and piecewise linear function. Theorem 4 summarizes the main result of the preprocessing step.

$$\gamma^+_{v_i}(t_{v_{i+1}}) = min\{\gamma^+_{v_i}(t_{v_{i+1}}), \delta_{v_i-n}(t_{v_{i+1}} + s_{v_{i+1}}) + s_{v_i-n} + L_{v_i-n}\} \tag{32}$$

Theorem 4. *For the TD-DARP, the computational complexity of auxiliary data in the preprocessing step is $O(|R|^2 n_{seg})$, where $|R|$ is the number of nodes visited by the route R, and n_{seg} is the number of intervals of the travel time function (the number of segments of the segment function).*

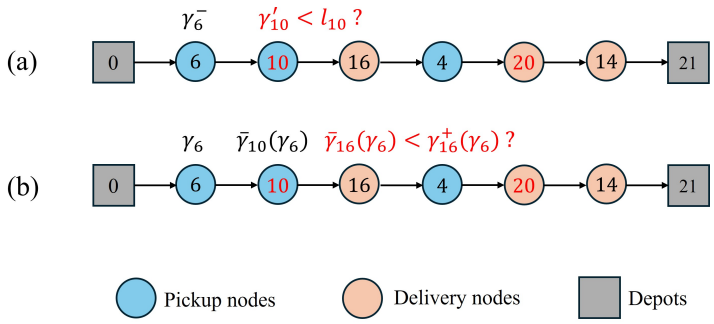
3.5. Illustrative Example of the Feasibility Test

To better demonstrate the process of the feasibility test, we include a small numerical example. Consider a set of requests with $n = 10$. A given partial route is $\{0, 6, 16, 4, 14, 21\}$, where nodes 0 and 21 denote the starting depot and ending depot, respectively. Nodes 6 and 16 represent the pickup and delivery nodes of request 6, respectively, while other requests follow the same correspondence. The goal is to evaluate the feasibility of inserting the pickup node 10 and the delivery node 20 immediately after nodes 6 and 4, respectively, resulting in the new route $\{0, 6, 10, 16, 4, 20, 14, 21\}$. The process consists of two stages: evaluating the insertion of the pickup node 10 and then the delivery node 20.

The evaluation begins with assessing the feasibility of inserting the pickup node 10. Following Algorithm 1, the earliest possible service time of node 10 is computed as $\gamma'_{10} = \max\{e_{10}, \gamma^-_6 + s_6 + \tau_{6,10}(\gamma^-_6 + s_6)\}$. As shown in Figure 5(a), if γ'_{10} exceeds its latest time window bound l_{10}, the insertion of the pickup node is infeasible, and the delivery node 20 does not need to be evaluated. Otherwise, the functions $\overline{\gamma}_{10}(\gamma_6)$ and $\overline{\gamma}_{16}(\gamma_6)$ are computed, and the insertion is further evaluated using the function $\gamma^+_{16}(\gamma_6)$, as illustrated in Figure 5(b). If a feasible domain $\gamma_6 \in I$ exists such that $\gamma^+_{16}(\gamma_6) > \overline{\gamma}_{16}(\gamma_6)$, the pickup insertion is feasible, and the algorithm proceeds to evaluate the delivery insertion.

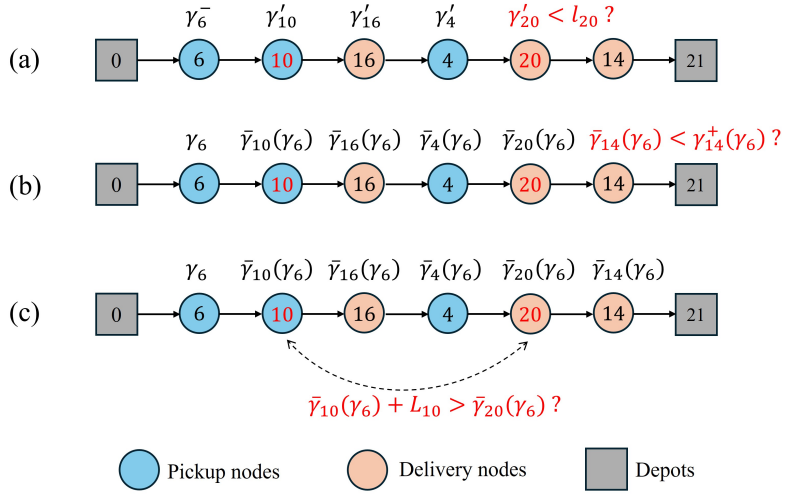
According to Algorithm 2, the earliest feasible service times of the involved nodes are calculated sequentially as $\gamma'_{10}, \gamma'_{16}, \gamma'_4, \gamma'_{20}$. As shown in Figure 6(a), if γ'_{20} exceeds its latest start time window l_{20}, the delivery insertion is infeasible. Otherwise, the functions $\overline{\gamma}_{10}(\gamma_6)$, $\overline{\gamma}_{16}(\gamma_6)$, $\overline{\gamma}_4(\gamma_6)$, $\overline{\gamma}_{20}(\gamma_6)$, and $\overline{\gamma}_{14}(\gamma_6)$ are successively determined. First, the feasibility of the insertion is evaluated using $\gamma^+_{14}(\gamma_6)$, as illustrated in Figure 6(b). If there exists a feasible domain $\gamma_6 \in I$ such that $\gamma^+_{14}(\gamma_6) > \overline{\gamma}_{14}(\gamma_6)$, it indicates that the additional time introduced by the insertion of node 10 and node 20 remains within the acceptable range of the service time at node 14. Next, the ride time constraints

Figure 5. Illustrative example of pickup insertion evaluation



are evaluated, as shown in Figure 6(c). If a subdomain $I' \in I$ satisfies $\overline{\gamma}_{20}(\gamma_6) < \overline{\gamma}_{10}(\gamma_6) + L_{10}$, the ride time constraints are respected. Hence, if the above evaluations are all satisfied, the insertion of the pickup and delivery nodes for request 10 is considered feasible.

Figure 6. Illustrative example of delivery insertion evaluation



4. Solution Approach

This section presents a feasibility test driven adaptive large neighborhood search (FT-ALNS) algorithm for solving TD-DARP. The adaptive large neighborhood search (ALNS) framework is a well-established optimization method for tackling complex routing problems. It provides a flexible and effective metaheuristic structure that can dynamically adapt search strategies based on their past performance (Ropke and Pisinger, 2006; Pisinger and Ropke, 2007). The proposed FT-ALNS algorithm integrates a variety of destruction and repair strategies through the use of a dynamic selection mechanism. Moreover, the integration of efficient feasibility test method within the ALNS framework enhances both the computational efficiency and overall performance of the algorithm.

4.1. General Framework

FT-ALNS consists of three key components: an adaptive destroy and repair procedure to explore the solution space, a tabu-based solution improvement procedure to enhance solution quality,

and a simulated annealing-based update criterion to help escape local optima. Algorithm 3 outlines the general framework of FT-ALNS. At the beginning, a basic greedy construction procedure is used to generate an initial solution S, where each request is inserted into the route at the best position, either minimizing cost or maximizing profit (Line 1). The destroy and repair operators are initialized with identical weights ρ^- and ρ^+ (Line 2). FT-ALNS then proceeds with an iterative search process. In each iteration, a new solution S', derived from the incumbent solution S (Line 5), first undergoes destruction and repair operations (Lines 6–8). This is followed by a tabu-based solution improvement procedure to further improve its quality (Line 9).

During the search process, each request insertion operation within the repair operation and tabu search requires a feasibility evaluation. First, the feasibility of inserting the pickup node is assessed using Algorithm 1. If the pickup insertion is feasible, Algorithm 2 is then used to evaluate the feasibility of inserting the delivery node. If the improved solution S' outperforms the best solution S^{best} found so far, it is accepted as the new S^{best} (Lines 10 and 11). The simulated annealing-based solution update criterion is then applied to determine whether the incumbent solution S should be updated (Lines 12 and 13). After each iteration, the weights of the operators are updated (Line 14). The algorithm terminates when a prefixed maximum number of iterations is reached and returns the best-found solution S^{best}.

Algorithm 3 Feasibility test driven adaptive large neighborhood search algorithm

Input: A given problem instance I
Output: The best found solution S^{best}

1: $S \leftarrow$ InitializationOperation(I) ▷ *Greedy construction with feasibility test*
2: $S^{best} \leftarrow S$, $\rho^- = \{1, ..., 1\}$, $\rho^+ = \{1, ..., 1\}$
3: $iter \leftarrow 0$
4: **while** $iter < iter_{max}$ **do**
5: $S' \leftarrow S$
6: Select a pair of destroy and repair operators based on the weights ρ^- and ρ^+
7: $S'_{part} \leftarrow$ DestroyOperation(S')
8: $S' \leftarrow$ RepairOperation(S'_{part}) ▷ *Repair step with feasibility test*
9: $S' \leftarrow$ TabuSearch(S') ▷ *Neighborhood move with feasibility test, see section 4.3*
10: **if** $f(S') < f(S^{best})$ **then**
11: $S^{best} \leftarrow S'$
12: **if** S' or S^{best} meet the acceptance criterion **then** ▷ *Updating strategy is triggered, see section 4.4*
13: Update S
14: Update the weights ρ^- and ρ^+ of operators
15: $iter \leftarrow iter + 1$
16: **return** The best found solution S^{best}

4.2. Adaptive Destroy and Repair Procedure

The adaptive destroy and repair procedure is designed to improve the solution by systematically applying a range of destroy and repair operators, which are selected based on their performance.

A destroy operator removes a subset of elements from the current solution, and these elements are subsequently reinserted using a repair operator. The repair operator requires evaluating the feasibility of each request insertion through a formal test procedure. In our algorithm, a carefully designed set of destroy and repair operators is used to refine the solution.

Specifically, the destroy operators consist of random removal and entire route removal (Hemmelmayr et al., 2012), as well as several adapted operators inspired by the literature, including the Shaw removal (Shaw, 1998), the worst removal (Ropke and Pisinger, 2006), and the zero-split removal, originally introduced for the DARP by Parragh et al. (2010). These operators have been tailored in this study to handle the TD-DARP. The *Shaw removal* approach removes a set of requests that exhibit similarity. The relatedness measure $R(i,j) = \alpha(c_{i,j} + c_{i+n,j+n}) + \beta(|\gamma_i^- - \gamma_j^-| + |\gamma_{i+n}^- - \gamma_{j+n}^-|) + \varphi(\sum_{k\in K} ky_i^k = \sum_{k\in K} ky_j^k)$ is defined to quantify the similarity between two requests i and j, where γ_i^- and γ_{i+n}^- represent the earliest start service times for the pickup and delivery nodes of request i. The term $\sum_{k\in K} ky_i^k = \sum_{k\in K} ky_j^k$ is used to determine whether both requests are served by the same vehicle. The *Shaw removal* operator starts by randomly selecting and removing a seed request. In each subsequent iteration, a request similar to those already removed is chosen for removal, continuing until q requests have been removed. The *worst removal* operator determines the removal gain of each request based on the current objective of the problem. Specifically, different metrics are used depending on whether the objective is to minimize total travel cost, minimize total route duration, or maximize total profit. For the cost- and duration-minimization problems, the removal gain represents the reduction in the total cost or total duration achieved by removing a given request, and the q requests with the largest gains are removed. For the profit-maximization problem, the removal gain corresponds to the decrease in total profit after removing the request, and the q requests with the smallest gains are removed, as they contribute the least to the overall profit. The *zero split removal* approach randomly selects a sequence between two arcs where the vehicle load is zero, and removes the requests within that sequence.

The repair operators include basic greedy reinsertion and regret-M reinsertion (Hemmelmayr et al., 2012), as well as an improved greedy reinsertion operator. The *improved greedy reinsertion* operator, designed specifically for problem P3, aims to enhance the selection of profitable combinations of requests. At each step, an unserved request is randomly selected and inserted into the best feasible position in the current routes, where the "best" position is determined by the maximum improvement in the overall profit after insertion. Unlike the standard greedy strategy, this operator allows the insertion of a request even if it temporarily reduces the total profit, as such moves can create opportunities for subsequent insertions of other requests with closely aligned time windows. These subsequent insertions may jointly yield a net profit increase by combining

their service rewards with only a small additional route duration.

The adaptive mechanism is based on a roulette wheel selection strategy, which is designed to choose a pair of destroy and repair operators in each iteration while dynamically adjusting their selection probabilities. Initially, all operators are assigned an identical weight ρ. When a new solution is found, the reward Ψ for the selected operator is updated based on the solution's quality: if the solution improves the best found or the incumbent solution, Ψ is incremented by ω_1 or ω_2, respectively. If the solution is worse than the incumbent solution but still accepted, Ψ is incremented by ω_3. At the end of every specified number of iterations, the weight of each operator is updated using the formula $\rho = \lambda\rho + (1-\lambda)\Psi$, where $\lambda \in [0,1]$ is a user-defined parameter that controls the sensitivity of weight adjustments. After updating, Ψ is reset to zero. The selection probability of operator i is determined by its historical performance, calculated as $\rho_i / \sum_{k=1}^{n} \rho_k$ where n is the total number of operators.

4.3. Tabu-Based Solution Improvement

To enhance search intensification, a tabu search procedure is incorporated into the proposed FT-ALNS algorithm. Building upon the solution obtained from the procedure described in Section 4.2, tabu search explores the neighborhood through a unified relocation framework that simultaneously considers both intra-route and inter-route relocations. In each iteration, the algorithm performs a full exploration of all feasible relocations by removing the pickup and delivery nodes of each request from their current positions and reinserting them into the most favorable positions, either within the same route or in a different one based on the calculated move gain. The request v_1 with the highest move gain is selected and relocated to the corresponding position within route R_1. A feasibility test procedure is applied throughout the relocation process to ensure the validity and efficiency of each insertion operation.

After the relocation process is executed, the move is recorded as (v_1, R_1) and added to the tabu list, preventing request v_1 from being moved to route R_1 again during the current tabu tenure. The tabu tenure is dynamically defined as $\lfloor 3\sqrt{N} \rfloor$, where N denotes the number of requests in the instance. This adaptive setting allows the tabu list size to scale appropriately with problem complexity, remaining short enough to avoid excessive restriction in small instances while long enough to prevent cycling in larger ones.

Importantly, the tabu list is reinitialized at the beginning of each tabu search phase, so that tabu restrictions apply only within the current phase and do not carry over between successive ones. Since certain promising moves may be temporarily restricted by the tabu mechanism, the search is not terminated immediately after a single non-improving iteration. Instead, it continues for a predefined number of consecutive non-improving iterations, allowing new improving moves to

become available once some tabu tenures expire. When no further improvement is observed within this threshold, the tabu search phase terminates.

4.4. Simulated Annealing-Based Solution Update Criterion

For the incumbent solution S, a simulated annealing (SA)–based acceptance criterion (Ropke and Pisinger, 2006) is employed to decide whether S should be replaced in each iteration. Such SA-like strategies, together with restart mechanisms from the best-known solution, are incorporated here to improve diversification and escape local optima. In our criterion, a superior solution is always accepted as the new incumbent S, while an inferior solution S' may also be accepted with a probability defined as $P = e^{\frac{-(f(S')-f(S)}{T^{\theta}})}$ for minimization problems. Here, the temperature parameter T^{θ} plays a crucial role as a control variable, gradually decreasing over iterations to balance exploration and exploitation. Additionally, the best found solution S^{best} is periodically utilized to update S, injecting high-quality information into the search process and potentially enhancing the algorithm's convergence performance.

5. Computational Results

This section reports the algorithm configuration, benchmark instances, and computational results for the three TD-DARP variants. The proposed algorithm was implemented in C++ and compiled using Visual Studio 2019 on a 64-bit Windows 10 platform. The MILP models were solved with IBM ILOG CPLEX 12.9 in single-threaded mode. All experiments were performed on a system with an Intel(R) Core(TM) i7-8700 CPU (3.20 GHz) and 8 GB of RAM.

5.1. Instances

Our experimental studies are conducted on a dataset of 120 instances, categorized into seven groups. Groups a and b are derived from the widely used DARP benchmark instances proposed by Cordeau (2006) and Cordeau and Laporte (2007). Groups AA, BB, CC, and DD originate from the TD-DARP benchmark sets provided by Zhao et al. (2022). These six groups are referred to as the regular groups in this study. Group Extend is generated by modifying the original instances from Zhao et al. (2022). All instances used in this study are available for download at `https://github.com/nwpu-orms/TD-DARP`.

- Group a comprises 24 instances of small, medium and large sizes, with requests ranging from 16 to 96 and the number of vehicles varying between 2 and 8. Each vehicle has a capacity of $C = 3$, while the service time $s_i = 3$, demand $d_i = 1$, and the maximum ride time $L_i = 30$ are identical for all requests. To reduce problem complexity, the time window tightening approach from Cordeau (2006) is applied. For speed profiles, the breakpoint settings are consistent

across all instances, reflecting similar congestion patterns during the day. Following Ichoua et al. (2003), three speed categories are defined for arcs: (1) normal speed on regular arcs, (2) slow speed on congested arcs, and (3) fast speed on free-flow arcs. To capture typical urban traffic dynamics, the day is divided into five time zones representing morning peak, midday off-peak, evening peak, late evening, and night hours. The travel speeds in each category vary across these time zones, with reduced speeds during peak periods and higher speeds during off-peak hours. These settings are applied consistently across all instances to ensure comparability.

- Group b consists of 24 instances of small, medium and large sizes. These instances between 16 and 96 requests, with the number of vehicles varying from 2 to 8. Each vehicle has a capacity of $C = 6$, and the maximum ride time for each request is $L_i = 45$. The demand d_i is randomly drawn from a uniform distribution over the set $\{1, ..., C\}$, with the service time s_i set equal to d_i for each request. In addition, the time window tightening approach and the speed profiles are the same as those used in Group a.
- Each of the groups AA, BB, CC, and DD consists of 14 instances, with the number of requests ranging from 10 to 75 in increments of 5. A single vehicle is deployed in each instance. Group AA features narrow time windows and small vehicle capacity, Group BB has narrow time windows and large vehicle capacity, Group CC includes wide time windows and small vehicle capacity, and Group DD is characterized by wide time windows and large vehicle capacity. Instances are classified as small-sized (10–30 requests), medium-sized (35–55 requests), or large-sized (60–75 requests).
- Group Extend consists of 16 large-sized instances. These instances are created by modifying the instances in Groups CC and DD, which contain 40 to 75 requests, by delaying pickup and delivery nodes l_i values by 60 units. This modification expands the time windows by 50%, thereby increasing the problem's complexity.

5.2. Parameter Settings

We adopt the parameter configuration method described in Ropke and Pisinger (2006). Initially, parameter values are determined through empirical estimation. During the fine-tuning process, one parameter is varied across multiple values while the others remain fixed. We selected all instances with no more than 24 requests from Groups a and b, all instances with no more than 30 requests from Groups AA, BB, CC, and DD, and all instances with no more than 50 requests from Group Extend. For each configuration, the algorithm was run five times independently on these selected instances, and the parameter setting yielding the best average performance was adopted in the subsequent experiments. Table 2 summarizes the final parameter settings of FT-ALNS, where N

represents the total number of requests. Specifically, we set $iter_{max} = 300$ for the instances in Group Extend, and $iter_{max} = 100$ for the remaining instances.

Table 2
The values of relevant parameters in FT-ALNS

Parameter	Description	Value
α	Weight factor in *Shaw removal*	0.25
β	Weight factor in *Shaw removal*	0.25
φ	Weight factor in *Shaw removal*	0.5
q	The number of removed requests	$0.5 \times N$
ρ	The initial weight of each operator	1
ω_1	Score increment	0.5
ω_2	Score increment	0.3
ω_3	Score increment	0.2
λ	Weight factor in the adaptive mechanism	0.5
T^{θ}	The initial temperature	1000
c	Cooling rate	0.95
$iter_{max}$	Maximum number of iterations	100, 300

5.3. Performance Comparison of FT-ALNS and CPLEX Solver

In this section, we present a comparative analysis of the results obtained by our FT-ALNS algorithm and those produced by CPLEX for problems P1 and P2, using instances from Groups *a* and *b*. Since the algorithm proposed in Zhao et al. (2022) outperforms CPLEX in solving P3, we provide performance comparisons with this algorithm in the following section. The detailed results comparing FT-ALNS and CPLEX for problems P1 and P2 are summarized in Tables 3 and 4. Columns *Obj* and *LB* report the best objective value and lower bound obtained by CPLEX within a 10-hour time limit. If *Obj* matches *LB*, the solution is considered optimal, and the corresponding objective value is marked with an asterisk. To evaluate the performance of our FT-ALNS algorithm, we conducted ten runs on each instance. Columns *Best* and *Avg* show the best and average objective values, while *Time (s)* indicates the average computation time. For readability, computation times are rounded to integers. The relative gap, shown in *Gap*, is calculated as (*Best* - *Obj*)/*Obj* $\times$100%. The best-found results are highlighted in bold.

As shown in Tables 3 and 4, CPLEX is only able to solve some small and medium-sized instances. Note that Tables 3 and 4 are based on the same set of small and medium-sized benchmark instances. However, the number of instances reported in Table 4 is smaller than that in Table 3. This is because the two problem variants consider different optimization objectives: P1 minimizes the total traveling cost, whereas P2 minimizes the total route duration. The latter objective leads to a more computationally challenging model, and as a result, CPLEX fails to find feasible solutions for several instances of P2 within the imposed time limit. These instances are therefore excluded from Table 4. For larger instances, the solver fails to provide any feasible solution within the time limit of 36000 seconds. In contrast, our FT-ALNS algorithm produces high-quality solutions in a relatively

short time. For problem P1, FT-ALNS obtains the best-found solutions over all 19 instances, with 8 instances outperforming CPLEX by an average gap of -1.17%. When comparing average solution values, FT-ALNS yields identical results to CPLEX on 2 instances and outperforms CPLEX on 5 instances, with an average gap of -1.28% over these instances. For problem P2, FT-ALNS achieves the best-found solutions on 13 instances, with 5 instances exceeding CPLEX's results, yielding an average gap of -6.53%. In terms of average solution quality, FT-ALNS matches CPLEX on 6 instances and produces better solutions on 5 instances, with an average gap of -6.28% on those instances. In summary, our FT-ALNS algorithm significantly outperforms the CPLEX solver in both solution quality and computational time.

Table 3
Comparison of CPLEX and FT-ALNS for solving problem P1: small and medium instances

Instance	CPLEX			FT-ALNS			Gap
	Obj	LB	Time (s)	Best	Avg	Time (s)	
a2-16	**277.53***	277.53	83	**277.53***	277.55	12	-
a2-20	**328.55***	328.55	163	**328.55***	329.08	17	-
a2-24	**396.34***	396.34	895	**396.34***	397.04	27	-
a3-18	**272.87***	272.87	1,674	**272.87***	273.52	9	-
a3-24	327.06	298.26	36,000	**322.51**	324.19	17	-1.39%
a3-30	459.33	398.92	36,000	**458.62**	462.51	27	-0.15%
a3-36	496.70	427.89	36,000	**494.46**	499.39	39	-0.45%
a4-16	260.57	225.86	36,000	**255.36**	255.51	8	-2.00%
a4-24	353.20	281.42	36,000	**352.69**	353.98	13	-0.14%
b2-16	**282.40***	282.40	167	**282.40***	282.44	12	-
b2-20	**324.12***	324.12	57	**324.12***	324.12	14	-
b2-24	**434.00***	434.00	2,392	**434.00***	434.07	22	-
b3-18	**294.37***	294.37	10,468	**294.37***	294.82	8	-
b3-24	**368.40**	337.31	36,000	**368.40**	368.40	13	-
b3-30	507.99	432.50	36,000	**504.01**	505.94	20	-0.78%
b3-36	**583.54**	479.21	36,000	**583.54**	585.91	27	-
b4-16	**286.91***	286.91	7,986	**286.91***	287.07	4	-
b4-24	357.50	296.78	36,000	**356.10**	356.79	10	-0.39%
b4-32	516.65	375.81	36,000	**495.78**	501.35	17	-4.04%

Table 4
Comparison of CPLEX and FT-ALNS for solving problem P2: small and medium instances

Instance	CPLEX			FT-ALNS			Gap
	Obj	LB	Time (s)	Best	Avg	Time (s)	
a2-16	**402.19***	402.19	363	**402.19***	402.19	77	-
a2-20	**462.40***	462.40	600	**462.40***	462.40	178	-
a2-24	**525.29***	525.29	674	**525.29***	525.29	311	-
a3-18	328.67	211.19	36,000	**322.74**	323.15	98	-1.80%
a3-24	479.36	186.74	36,000	**463.85**	464.09	216	-3.23%
a4-16	**241.01**	-48.70	36,000	**241.01**	241.17	56	-
a4-24	509.94	-581.53	36,000	**385.25**	388.36	187	-24.45%
b2-16	**533.93***	533.93	261	**533.93***	533.93	68	-
b2-20	**603.23***	603.23	297	**603.23***	603.23	89	-
b2-24	**769.62***	769.62	5,619	**769.62***	769.62	202	-
b3-18	**391.22**	213.44	36,000	394.16	394.16	75	0.75%
b3-24	**585.93**	370.32	36,000	**585.93**	586.36	144	-
b4-16	317.97	63.80	36,000	**317.64**	317.70	35	-0.10%
b4-32	721.83	18.85	36,000	**699.92**	702.88	266	-3.04%

Table 5 summarizes the experimental results of FT-ALNS on medium and large sized instances.

We observe that computational time increases as the number of requests and vehicles grows. In addition, the group of instances a, characterized by shorter ride times, are more challenging to solve due to the difficulty of satisfying the maximum ride time constraints. The best-found solutions reported in Table 5 can serve as benchmarks for future studies.

Table 5
Results of problem P1 for the remaining medium and large instances solved by FT-ALNS

Instance	FT-ALNS			Instance	FT-ALNS		
	Best	Avg	Time (s)		Best	Avg	Time (s)
a4-32	457.04	460.41	24	b4-32	495.78	501.35	17
a4-40	529.53	540.21	35	b4-40	647.58	650.17	28
a4-48	636.33	642.79	58	b4-48	652.30	659.99	46
a5-40	491.44	495.21	38	b5-40	607.25	613.01	26
a5-50	640.22	647.83	57	b5-50	736.84	742.22	34
a5-60	761.71	777.75	74	b5-60	888.67	899.85	54
a6-48	597.43	608.41	52	b6-48	706.52	713.78	28
a6-60	758.15	776.89	73	b6-60	851.18	864.81	45
a6-72	875.18	887.94	108	b6-72	978.76	986.79	78
a7-56	702.86	716.76	61	b7-56	822.85	829.79	36
a7-70	843.99	853.22	99	b7-70	906.19	915.91	60
a7-84	1,009.22	1,028.44	147	b7-84	1,197.27	1,218.42	95
a8-64	716.19	729.20	85	b8-64	825.46	840.42	49
a8-80	914.95	927.79	136	b8-80	1,040.42	1,051.89	71
a8-96	1,152.94	1,177.82	240	b8-96	1,185.08	1,205.49	125

To further examine the impact of distinct optimization objectives, Table 6 reports a quantitative comparison of the optimal solutions obtained under P1 and P2 for several representative instances. As shown in the table, for the same instance, the solution optimized for distance (P1) and that optimized for duration (P2) exhibit pronounced differences not only in distance-based cost but also in total route duration. In particular, minimizing distance does not necessarily yield short route durations, while solutions optimized for duration may incur significantly higher travel distance. This comparison demonstrates that distinct objectives yield markedly different solution characteristics and operational trade-offs, reinforcing the flexibility of the proposed FT-ALNS algorithm in handling diverse optimization goals.

Table 6
Comparison of distance-based cost and route duration under P1 and P2

Instance	P1		P2	
	Distance-based cost	Route duration	Distance-based cost	Route duration
a2-16	277.53	960	356.56	402.19
a2-20	328.55	1200	396.29	462.40
a2-24	396.34	1440	470.49	525.29
b2-16	282.40	960	342.63	533.93
b2-20	324.12	1200	374.30	603.23
b2-24	434.00	1440	502.15	769.62

5.4. Performance Comparison of FT-ALNS and the State-of-the-Art Algorithm

To evaluate our FT-ALNS algorithm for problem P3, we compare it against the adaptive large neighborhood search with local search (ALNS-LS) algorithm proposed by Zhao et al. (2022), using instances from Groups AA, BB, CC and DD. Tables 7 through 9 present the comparative results for small, medium, and large instances. Column N_R shows the number of served requests. The relative gap, displayed in *Gap*, is calculated as (*Best (FT-ALNS) - Best (ALNS-LS)*)/*Best (ALNS-LS)* $\times 100\%$.

From Table 7, we observe that our FT-ALNS algorithm matches the best solutions for 19 out of 20 small instances and improves the best solution for instance BB10, resulting in an average gap of 1.10% on best solutions. When average solution values are considered, FT-ALNS attains an average gap of 0.71%. Table 8 reports the results on medium instances. In terms of best solutions, FT-ALNS yields identical results for 11 instances, better results for 4 instances, and worse results for 5 instances, with an average gap of 0.07%. For average solutions, the comparison leads to an average gap of -3.66%, indicating a slight disadvantage relative to ALNS-LS on these instances. Finally, Table 9 summarizes the results for large instances. FT-ALNS produces better best solutions on 12 instances and worse results on only 2 instances, yielding an average gap of 1.03% on best solutions. When average solution quality is considered, FT-ALNS attains an average gap of -0.52%. The gaps for average solutions are computed analogously to those for best solutions, based on the average objective values obtained by the two methods.

Table 7
Comparative results of problem P3 for the small instances between ALNS-LS and FT-ALNS

Instance	ALNS-LS (Zhao et al., 2022)				FT-ALNS				Gap
	Best	N_R	Avg	Time (s)	Best	N_R	Avg	Time (s)	
AA10	**26.89**	2	26.89	< 1	**26.89**	2	26.89	1	-
BB10	32.62	3	32.62	< 1	**39.78**	3	39.78	< 1	21.95%
CC10	**30.47**	2	30.47	< 1	**30.47**	2	30.47	< 1	-
DD10	**54.05**	3	54.05	< 1	**54.05**	3	54.05	1	-
AA15	**40.56**	4	40.56	< 1	**40.56**	4	38.94	2	-
BB15	**35.62**	4	35.62	< 1	**35.62**	4	35.62	3	-
CC15	**94.08**	5	94.08	< 1	**94.08**	5	94.08	6	-
DD15	**55.07**	3	55.07	< 1	**55.07**	3	55.07	2	-
AA20	**89.33**	6	60.86	< 1	**89.33**	6	87.47	15	-
BB20	**143.04**	11	143.04	< 1	**143.04**	11	140.36	23	-
CC20	**94.71**	6	94.71	3	**94.71**	6	87.66	12	-
DD20	**209.70**	16	209.70	3	**209.70**	16	206.45	28	-
AA25	**227.10**	13	227.10	2	**227.10**	14	206.67	37	-
BB25	**172.32**	13	172.32	2	**172.32**	13	141.85	26	-
CC25	**266.45**	18	227.10	3	**266.45**	18	246.88	39	-
DD25	**296.34**	18	296.34	5	**296.34**	18	282.92	84	-
AA30	**285.38**	16	285.38	2	**285.38**	16	253.15	31	-
BB30	**304.63**	21	304.63	3	**304.63**	21	299.76	55	-
CC30	**318.83**	19	318.83	7	**318.83**	19	318.83	51	-
DD30	**386.88**	21	386.88	13	**386.88**	21	382.00	74	-

Due to differences in experimental environments, we do not attempt to compare the computation times reported in Zhao et al. (2022). However, since the source code of ALNS-LS is available, we run it on our computational platform using the instances from Group Extend. Following Zhao

Table 8
Comparative results of problem P3 for the medium instances between ALNS-LS and FT-ALNS

Instance	ALNS-LS (Zhao et al., 2022)				FT-ALNS				Gap
	Best	N_R	Avg	Time (s)	Best	N_R	Avg	Time (s)	
AA35	**314.90**	20	314.90	9	**314.90**	20	257.62	37	-
BB35	**370.26**	22	370.26	11	**370.26**	22	351.69	60	-
CC35	**410.70**	23	410.70	17	**410.70**	23	407.28	83	-
DD35	**436.81**	24	436.81	28	436.00	24	406.51	114	-0.19%
AA40	**371.25**	22	371.25	24	**371.25**	22	363.75	67	-
BB40	**356.02**	22	356.02	48	**356.02**	22	347.13	61	-
CC40	**476.93**	24	476.93	97	**476.93**	24	451.28	143	-
DD40	**503.47**	24	502.18	128	502.69	24	473.66	131	-0.15%
AA45	**391.08**	19	391.08	39	**391.08**	19	386.17	78	-
BB45	**371.63**	21	371.63	55	**371.63**	21	362.88	68	-
CC45	**481.22**	23	480.15	126	478.29	24	467.44	137	-0.61%
DD45	**551.50**	25	544.67	178	**551.50**	26	516.49	153	-
AA50	**468.56**	23	468.56	55	**468.56**	23	467.96	120	-
BB50	474.13	23	474.13	68	**474.26**	23	457.40	145	0.03%
CC50	**564.15**	26	564.15	189	562.91	26	546.26	174	-0.22%
DD50	697.66	29	686.72	279	**704.54**	29	663.65	275	0.99%
AA55	596.10	28	587.79	98	**598.05**	25	573.81	228	0.33%
BB55	**704.70**	30	697.19	143	**704.70**	30	691.20	431	-
CC55	**667.26**	29	645.32	233	665.86	29	638.89	334	-0.21%
DD55	727.01	29	711.35	389	**737.01**	29	706.07	292	1.38%

Table 9
Comparative results of problem P3 for the large instances between ALNS-LS and FT-ALNS

Instance	ALNS-LS (Zhao et al., 2022)				FT-ALNS				Gap
	Best	N_R	Avg	Time (s)	Best	N_R	Avg	Time (s)	
AA60	463.22	23	463.22	179	**464.80**	23	457.32	101	0.34%
BB60	582.53	28	579.18	112	**604.16**	28	570.90	188	3.71%
CC60	692.22	29	677.25	290	**698.62**	29	665.99	378	0.92%
DD60	781.72	31	771.39	419	**812.95**	33	767.70	420	4.00%
AA65	745.26	33	729.97	225	**751.60**	31	716.72	352	0.85%
BB65	655.24	28	640.49	280	**657.30**	29	642.55	288	0.32%
CC65	**725.44**	31	710.16	330	714.32	32	684.56	394	-1.53%
DD65	**788.84**	33	782.09	506	**788.84**	31	759.92	512	-
AA70	686.43	29	660.27	332	**705.95**	28	664.79	283	2.84%
BB70	**670.48**	30	655.22	501	**670.48**	30	654.71	290	-
CC70	**761.11**	32	739.17	498	754.61	32	736.36	406	-0.85%
DD70	812.25	31	772.31	889	**815.00**	32	785.05	378	0.34%
AA75	768.34	31	751.12	551	**771.89**	31	756.38	358	0.46%
BB75	751.33	32	726.04	570	**753.07**	32	716.83	427	0.23%
CC75	819.75	33	798.20	668	**848.36**	34	820.96	491	3.49%
DD75	911.40	33	864.66	1,471	**923.81**	35	867.36	444	1.36%

et al. (2022), we execute ALNS-LS with 10 different random seeds, recording the best objective value, the number of served requests, the average objective value, and the average computation time. The comparative results of ALNS-LS and FT-ALNS algorithm are presented in Table 10.

As shown in Table 10, FT-ALNS algorithm performs exceptionally well on 16 instances from Group Extend. Specifically, FT-ALNS obtains the best-found solutions for 12 instances, with an average gap of 0.09% across all instances. When average solution values are considered, FT-ALNS yields an average gap of -1.12%. Although the algorithm by Zhao et al. (2022) shows slightly better average results on some instances, this difference can be attributed to the design of FT-ALNS, which prioritizes intensification and computational efficiency. Consequently, while FT-ALNS achieves highly competitive best solutions, it may exhibit slightly higher variability across runs. As shown in Figure 7, FT-ALNS significantly reduces computation time compared to ALNS-LS, with the advantage becoming more pronounced as the instance size increases. On average, FT-ALNS offers a speedup factor of 8.59 over ALNS-LS, with a maximum speedup of 17.85.

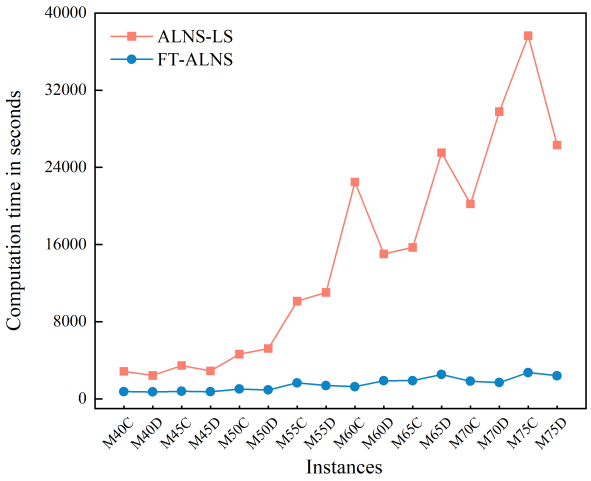
Table 10
Comparative results of problem P3 for the extended instances between ALNS-LS and FT-ALNS

Instance	ALNS-LS (Zhao et al., 2022)				FT-ALNS				Gap
	Best	N_R	Avg	Time (s)	Best	N_R	Avg	Time (s)	
M40C	500.55	24	498.05	2,856	**501.60**	25	492.48	755	0.21%
M40D	**545.98**	25	533.01	2,425	**545.98**	25	530.13	720	-
M45C	**529.74**	26	519.66	3,456	**529.74**	26	520.97	785	-
M45D	586.21	28	573.16	2,903	**587.57**	28	560.20	736	0.23%
M50C	610.07	27	602.25	4,628	**611.22**	27	602.37	1,023	0.19%
M50D	**725.11**	29	703.16	5,218	725.06	29	700.29	923	-0.01%
M55C	694.63	29	682.66	10,122	**696.83**	29	678.80	1,658	0.32%
M55D	**800.63**	31	791.07	11,029	794.61	31	767.00	1,381	-0.75%
M60C	772.19	32	749.52	22,477	**772.77**	32	729.86	1,259	0.08%
M60D	**861.93**	33	840.24	15,021	857.23	32	841.89	1,876	-0.55%
M65C	789.83	32	779.29	15,691	**790.78**	32	757.89	1,889	0.12%
M65D	**883.44**	33	864.75	25,523	872.33	32	852.61	2,528	-1.26%
M70C	807.87	33	799.79	20,207	**815.78**	33	795.65	1,827	0.98%
M70D	**851.27**	32	840.34	29,775	**851.27**	32	825.83	1,694	-
M75C	874.79	34	867.36	37,643	**889.15**	34	860.81	2,728	1.64%
M75D	984.97	36	972.86	26,308	**986.83**	36	966.91	2,398	0.19%

5.5. Evaluation of Feasibility Test Effectiveness

To further evaluate the effectiveness of our proposed feasibility test approach, we embed the feasibility test of Zhao et al. (2022) into our ALNS framework, resulting in a variant referred to as FT-ALNS(Zhao). We then compare it against our proposed FT-ALNS algorithm under identical experimental conditions. A total of 16 benchmark instances are selected from Groups AA, BB, CC, DD, and Extend, ensuring that each group and problem scale (with the number of requests ranging from 10 to 75 in increments of 5) is represented. Each instance was solved independently 10 times, and both the best and average results are reported. The computational results are summarized in Table 11. The column *Gap* shows the relative improvement of FT-ALNS over FT-ALNS(Zhao), computed as (*Best (FT-ALNS) - Best (FT-ALNS (Zhao))*)/*Best (FT-ALNS (Zhao))* $\times 100\%$.

Figure 7. Computation times of ALNS-LS and FT-ALNS on extended instances



From the results, it can be observed that FT-ALNS attains the best solutions in 15 out of 16 instances, outperforming FT-ALNS (Zhao) on 8 instances, matching it on 7, and yielding a slightly inferior solution on only one. The average gap is 0.78% in favor of FT-ALNS. Regarding average solution quality, FT-ALNS also demonstrates a competitive advantage. Specifically, FT-ALNS yields better average solutions on 8 instances, inferior average solutions on 6 instances, and identical average solutions on the remaining 2 instances when compared with FT-ALNS (Zhao). The resulting average gap over all instances is 0.19%, computed analogously based on the average solution values of the two methods. The differences observed in the best and average solution values can be attributed to the fact that different feasibility tests induce slightly different feasible regions, which in turn affect the set of solutions explored during the search. A more accurate feasibility test enables the algorithm to explore a broader solution space. In terms of computation time, FT-ALNS demonstrates a substantial efficiency advantage, achieving an average speedup factor of 30 and a maximum of 64 over FT-ALNS (Zhao). This remarkable improvement highlights the efficiency of the proposed feasibility test. Overall, these results confirm that the proposed feasibility test approach not only maintains high-quality solutions but also enables significantly faster evaluations within the ALNS framework, thereby enhancing the algorithm's overall performance and scalability for large-scale TD-DARP instances.

6. Conclusion

The time-dependent dial-a-ride problem extends the classical dial-a-ride problem by incorporating travel times that vary based on the vehicle's departure time. This NP-hard problem has numerous practical applications, including door-to-door transportation services for patients and the elderly, as well as solving last-mile delivery challenges. In this paper, we study three variants

Table 11
Comparative results of FT-ALNS with different feasibility test approaches

Instance	FT-ALNS (Zhao)				FT-ALNS				Gap
	Best	N_R	Avg	Time (s)	Best	N_R	Avg	Time (s)	
AA10	**26.89**	2	26.89	6	**26.89**	2	26.89	1	-
BB15	**35.62**	4	35.62	52	**35.62**	4	35.62	3	-
CC20	**94.71**	6	94.71	170	**94.71**	6	87.66	12	-
DD25	292.65	18	278.39	644	**296.34**	18	282.92	84	1.26%
AA30	**285.38**	16	257.23	644	**285.38**	16	253.15	31	-
BB35	**370.26**	22	345.91	1,214	**370.26**	22	351.69	60	-
CC40	**476.93**	24	450.45	1,672	**476.93**	24	451.28	143	-
DD45	537.74	25	494.96	6,605	**551.50**	26	516.49	153	2.56%
AA50	**468.56**	23	468.31	4,449	**468.56**	23	467.96	120	-
BB55	699.40	30	670.08	9,100	**704.70**	30	691.20	431	0.76%
CC60	**702.65**	28	689.06	16,666	698.62	29	665.99	378	-0.57%
DD65	775.06	31	759.57	30,837	**788.84**	31	759.92	512	1.78%
AA70	682.24	28	633.88	18,129	**705.95**	28	664.79	283	3.48%
BB75	751.02	30	719.71	26,950	**753.07**	32	716.83	427	0.27%
M75C	871.21	34	856.51	50,616	**889.15**	34	860.81	2,728	2.06%
M75D	978.92	35	971.48	78,966	**986.83**	36	966.91	2,398	0.81%

of TD-DARP and develop a general mathematical model that accommodates different objectives and characteristics. More importantly, we propose a novel feasibility test approach with linear time complexity to efficiently evaluate the feasibility of a given TD-DARP route. We then introduce a feasibility-driven adaptive large neighborhood search (FT-ALNS) algorithm to solve the three TD-DARP variants. Our approach outperforms both the CPLEX solver and the state-of-the-art TD-DARP heuristic in terms of solution quality and computational efficiency.

In future work, we intend to introduce electric vehicles into the TD-DARP framework. Unlike traditional fuel vehicles, electric vehicles have a limited range that requires careful management of their charging behavior during the service process. This limitation can significantly affect both the service order and the service time. Future research will focus on exploring this challenge and developing enhanced feasibility test approaches and metaheuristics to effectively address it.

CRediT authorship contribution statement

Wenjie Hu: Methodology, Software, Visualization, Writing – original draft, Writing – review and editing. **Yang Wang:** Funding acquisition, Methodology, Supervision, Writing – review and editing. **Jin-Kao Hao:** Conceptualization, Investigation, Writing – review and editing. **Wei-Neng Chen:** Conceptualization, Writing – review and editing. **Jiliu Li:** Conceptualization, Funding acquisition, Methodology, Supervision, Writing – review and editing. **Xin Jin:** Data curation, Methodology, Software.

Acknowledgments

The authors are grateful to Dr. Jingyi Zhao for kindly sharing the benchmark instances and the code used in Zhao et al. (2022). This work was supported in part by the National Key Research and Development Project No. 2023YFE0206200.

References

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

Braekers, K., Kovacs, A.A., 2016. A multi-period dial-a-ride problem with driver consistency. Transportation Research Part B: Methodological 94, 355–377.

Chassaing, M., Duhamel, C., Lacomme, P., 2016. An els-based approach with dynamic probabilities management in local search for the dial-a-ride problem. Engineering Applications of Artificial Intelligence 48, 119–133.

Cordeau, J.F., 2006. A branch-and-cut algorithm for the dial-a-ride problem. Operations research 54, 573–586.

Cordeau, J.F., Laporte, G., 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transportation Research Part B: Methodological 37, 579–594.

Cordeau, J.F., Laporte, G., 2007. The dial-a-ride problem: models and algorithms. Annals of Operations Research 153, 29–46.

Detti, P., Papalini, F., de Lara, G.Z.M., 2017. A multi-depot dial-a-ride problem with heterogeneous vehicles and compatibility constraints in healthcare. Omega 70, 1–14.

Firat, M., Woeginger, G.J., 2011. Analysis of the dial-a-ride problem of hunsaker and savelsbergh. Operations Research Letters 39, 32–35.

Gschwind, T., Drexl, M., 2019. Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. Transportation Science 53, 480–491.

Hemmelmayr, V.C., Cordeau, J.F., Crainic, T.G., 2012. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. Computers & Operations Research 39, 3215–3228.

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

Hu, T.Y., Chang, C.P., 2015. A revised branch-and-price algorithm for dial-a-ride problems with the consideration of time-dependent travel cost. Journal of Advanced Transportation 49, 700–723.

Hunsaker, B., Savelsbergh, M., 2002. Efficient feasibility testing for dial-a-ride problems. Operations research letters 30, 169–173.

Ichoua, S., Gendreau, M., Potvin, J.Y., 2003. Vehicle dispatching with time-dependent travel times. European Journal of Operational Research 144, 379–396.

Karimi, M., Camiat, F., Desaulniers, G., Gendreau, M., 2024. An exact branch-and-price-and-cut algorithm for a practical and large-scale dial-a-ride problem. Journal of the Operational Research Society , 1–15.

Kirchler, D., Calvo, R.W., 2013. A granular tabu search algorithm for the dial-a-ride problem. Transportation Research Part B: Methodological 56, 120–135.

Lera-Romero, G., Miranda-Bront, J.J., 2021. A branch and cut algorithm for the time-dependent profitable tour problem with resource constraints. European Journal of Operational Research 289, 879–896.

Liu, C., Kou, G., Zhou, X., Peng, Y., Sheng, H., Alsaadi, F.E., 2020. Time-dependent vehicle routing problem with time windows of city logistics with a congestion avoidance approach. Knowledge-Based Systems 188, 104813.

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 129, 105196.

Masmoudi, M.A., Hosny, M., Demir, E., Pesch, E., 2020. Hybrid adaptive large neighborhood search algorithm for the mixed fleet heterogeneous dial-a-ride problem. Journal of Heuristics 26, 83–118.

Molenbruch, Y., Braekers, K., Caris, A., 2017. Benefits of horizontal cooperation in dial-a-ride services. Transportation Research Part E: Logistics and Transportation Review 107, 97–119.

Ouasaid, S., Saddoune, M., 2024. An enhanced approach for the dial a ride problem with drivers preferences. arXiv preprint arXiv:2406.04506 .

Pan, B., Zhang, Z., Lim, A., 2021. Multi-trip time-dependent vehicle routing problem with time windows. European Journal of Operational Research 291, 218–231.

Pandi, R.R., Ho, S.G., Nagavarapu, S.C., Dauwels, J., 2020. A generic gpu-accelerated framework for the dial-a-ride problem. IEEE Transactions on Intelligent Transportation Systems 22, 6473–6488.

Parragh, S.N., Doerner, K.F., Hartl, R.F., 2010. Variable neighborhood search for the dial-a-ride problem. Computers & Operations Research 37, 1129–1138.

Parragh, S.N., Pinho de Sousa, J., Almada-Lobo, B., 2015. The dial-a-ride problem with split requests and profits. Transportation Science 49, 311–334.

Pisinger, D., Ropke, S., 2007. A general heuristic for vehicle routing problems. Computers & operations research 34, 2403–2435.

Qu, Y., Bard, J.F., 2015. A branch-and-price-and-cut algorithm for heterogeneous pickup and delivery problems with configurable vehicle capacity. Transportation Science 49, 254–270.

Rist, Y., Forbes, M.A., 2021. A new formulation for the dial-a-ride problem. Transportation Science 55, 1113–1135.

Ritzinger, U., Puchinger, J., Hartl, R.F., 2016. Dynamic programming based metaheuristics for the dial-a-ride problem. Annals of Operations Research 236, 341–358.

Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40, 455–472.

Sayarshad, H.R., Chow, J.Y., 2015. A scalable non-myopic dynamic dial-a-ride and pricing problem. Transportation Research Part B: Methodological 81, 539–554.

Schenekemberg, C.M., Chaves, A.A., Coelho, L.C., Guimarães, T.A., Avelino, G.G., 2022. The dial-a-ride problem with private fleet and common carrier. Computers & Operations Research 147, 105933.

Schenekemberg, C.M., Chaves, A.A., Guimarães, T.A., Coelho, L.C., 2024. Hybrid metaheuristic for the dial-a-ride problem with private fleet and common carrier integrated with public transportation. Annals of Operations Research , 1–39.

Schilde, M., Doerner, K.F., Hartl, R.F., 2014. Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. European Journal of Operational Research 238, 18–30.

Shaw, P., 1998. Using constraint programming and local search methods to solve vehicle routing problems, in: International conference on principles and practice of constraint programming, Springer. pp. 417–431.

Sohrabi, S., Ziarati, K., Keshtkaran, M., 2024. Revised eight-step feasibility checking procedure with linear time complexity for the dial-a-ride problem (darp). Computers & Operations Research 164, 106530.

Su, Y., Dupin, N., Parragh, S.N., Puchinger, J., 2024. A branch-and-price algorithm for the electric autonomous dial-a-ride problem. Transportation Research Part B: Methodological 186, 103011.

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 309, 1091–1111.

Timothée, C.H., Samuel, V., Thibaud, M., 2023. The assignment-dial-a-ride-problem. Health Care Management Science 26, 770–784.

Xiang, Z., Chu, C., Chen, H., 2008. The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments. European Journal of Operational Research 185, 534–551.

Zhang, Z., Liu, M., Lim, A., 2015. A memetic algorithm for the patient transportation problem. Omega 54, 60–71.

Zhao, J., Poon, M., Zhang, Z., Gu, R., 2022. Adaptive large neighborhood search for the time-dependent profitable dial-a-ride problem. Computers & Operations Research 147, 105938.