Transportation Research Part C 172 (2025) 105022

Contents lists available at ScienceDirect

Transportation Research Part C

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

The Dynamic Traveling Salesman Problem with Time-Dependent and Stochastic travel times: A deep reinforcement learning approach

Dawei Chen [a,*], Christina Imdahl [a], David Lai [b], Tom Van Woensel [a]

[a] Department of Industrial Engineering & Innovation Sciences, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, Netherlands

[b] Department of Business Analytics and Applied Economics, Queen Mary University of London, E1 4NS, United Kingdom

ARTICLE INFO

Keywords:
Dynamic traveling salesman problem
Time-dependent and stochastic travel times
Deep reinforcement learning

ABSTRACT

We propose a novel approach using deep reinforcement learning to tackle the Dynamic Traveling Salesman Problem with Time-Dependent and Stochastic travel times (DTSP-TDS). The main goal is to dynamically plan the route with the shortest tour duration that visits all customers while considering the uncertainties and time-dependence of travel times. We employ a reinforcement learning approach to dynamically address the stochastic travel times to observe changing states and make decisions accordingly. Our reinforcement learning approach incorporates a Dynamic Graph Temporal Attention model with multi-head attention to dynamically extract information about stochastic travel times. Numerical studies with varying amounts of customers and time intervals are conducted to test its performance. Our proposed approach outperforms other benchmarks regarding solution quality and solving time, including the rolling horizon heuristics and other existing reinforcement learning approaches. In addition, we demonstrate the generalization capability of our approach in solving the various DTSP-TDS in various scenarios.

1. Introduction

The Traveling Salesman Problem (TSP) is a combinatorial optimization problem that aims to find the shortest route for visiting a given set of customers, where the travel times between customers are deterministic and constant over time. However, in many transportation and logistics applications, the travel time is time-dependent during the day because of regular changes in traffic flow, such as rush hours and public holidays. The Traveling Salesman Problem with Time-Dependent travel times (TSP-TD) aims to design the route with the shortest tour duration in a graph, where the time-dependent travel time is represented as a time-discrete variable (Gao and Huang, 2012). Moreover, due to changing weather or traffic accidents, the time-dependent travel time is stochastic in reality. The time-dependent and stochastic travel time has been an important consideration for many practical problems, such as optimizing the routes for e-commerce delivery (Kitjacharoenchai et al., 2019; Huang et al., 2019; Özarık et al., 2021; Gouveia et al., 2023), door-to-door services (Huang et al., 2019; Pham and Kiesmüller, 2022), and planning healthcare service routes (Hashemi Doulabi et al., 2020), where tour duration comprises a significant portion of the cost. Ignoring the time-dependent and stochastic nature of travel times can cause significant deviations from the expected travel times on the planned route, leading to increased tour duration. It is crucial to dynamically optimize the routing in the operating stage in response to the changing travel times in real-time (Gmira et al., 2021a).

* Corresponding author.
E-mail address: d.chen@tue.nl (D. Chen).

https://doi.org/10.1016/j.trc.2025.105022
Received 26 April 2024; Received in revised form 25 October 2024; Accepted 22 January 2025
Available online 12 February 2025

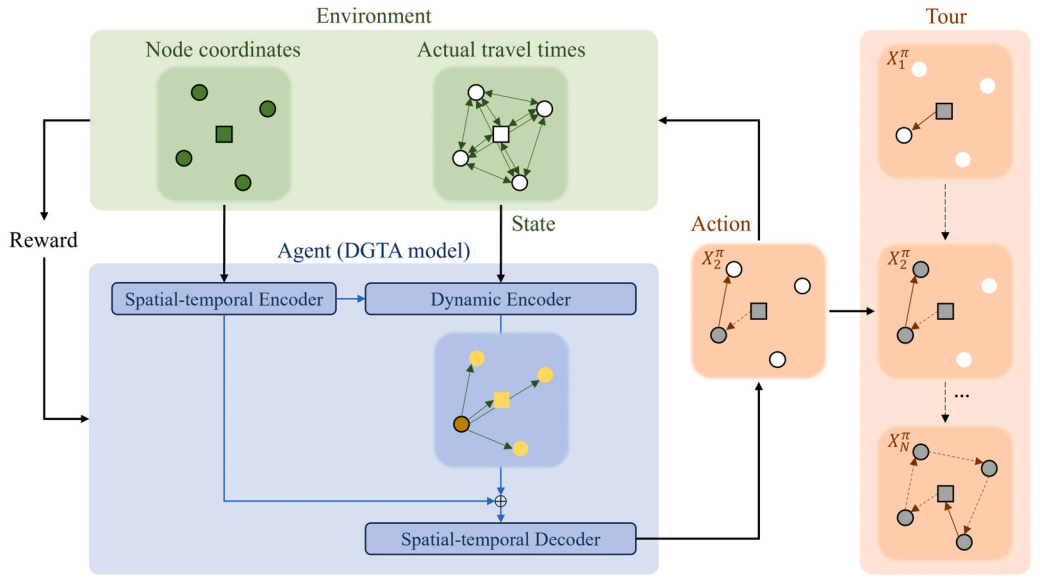

Fig. 1. The framework of DGTA-RL.

This paper addresses the challenges of the Dynamic Traveling Salesman Problem with Time-Dependent and Stochastic travel times (DTSP-TDS). The complexity of DTSP-TDS arises from the need to make sequential tour decisions while dealing with stochastic information (Ulmer et al., 2020). Traditional approaches can address system uncertainties and provide nearly optimal solutions for the TSP-TD. However, they struggle to effectively manage stochastic travel times swiftly because these approaches take extensive solving time to generate solutions, which inhibits their ability to respond to stochastic systems in real-time (Sun et al., 2019).

Recently, the Reinforcement Learning (RL) approach, which integrates the reinforcement learning algorithm with deep neural networks (DNN), has emerged as a promising approach for tackling complex online decision-making problems involving uncertainty and dynamicity (Pan and Liu, 2023). Hildebrandt et al. (2022) outlined the potential of RL approach to effectively handle the dynamic vehicle routing problem with stochastic information. After training, RL approach can generate the online tour in a short time. Some DTSP-TDS research uses reinforcement learning algorithms, such as Liu et al. (2022), which utilize a multilayer perceptron to solve the drone-assisted delivery problem. Zhang et al. (2021) proposed a neural network with a decoder–encoder structure to solve DTSP-TDS. They deal with the time-dependent travel time by combining the vectors of the current and the expected traffic conditions at each node. In their approach, each customer's location shares a common traffic condition value independent of the destination, which may not accurately represent the traffic conditions along the different arcs originating from that customer. However, integrating the matrix of time-dependent travel times will significantly increase computational complexity. To address this computational complexity, Guo et al. (2023) incorporated dimension-reduction and gate mechanisms in their model to encode the matrix of time-dependent travel times. Their model can solve the dynamic TSP with time-dependent and deterministic travel times, but they do not consider the stochastic characteristics of future travel times.

The Graph Temporal Attention model (GTA model) introduced by Gunarathna et al. (2022) addresses the dynamic TSP with time-dependent node locations by separately encoding static and dynamic states before decoding decisions. The GTA model consists of a spatial–temporal attention layer, a temporal decoder, and a multi-head attention layer. Expanding upon this framework, we proposes a Dynamic Graph Temporal Attention model (DGTA model) with Reinforcement Learning algorithm (DGTA-RL). To the best of our knowledge, this is the first RL approach to solve the DTSP-TDS. In contrast to the GTA model, the DGTA model employs several novel components, including a dynamic encoder, and a temporal pointer. These enhancements allow the DGTA model to efficiently handle stochastic and time-dependent travel times. Fig. 1 illustrates the decision-making process of the DGTA-RL, where the DGTA model is an agent interacting with a stochastic environment. At each decision step, the arrival times to all other nodes are realized in the dynamic encoder. Following the dynamic encoder, the spatial–temporal decoder generate the action at each decision step. The tour is the sequence of actions at each decision step.

We tested the effectiveness of the proposed DGTA-RL approach in solving the Static TSP-TD with Time-dependent and Deterministic travel times (STSP-TDD), besides the DTSP-TDS. While the DTSP-TDS is our main focus, the numerical study on the STSP-TDD provide insights on the impact of time-dependent travel time in route optimization. In solving STSP-TDD, the proposed DGTA-RL approach outperforms two existing RL approaches (AMDRGM-RL proposed by Guo et al. 2023 and NN-RL), which shows the effectiveness of DGTA-RL in handling time-dependent and deterministic travel times. For DTSP-TDS, we test DGTA-RL against two heuristic-based rolling approaches (Rolling-greedy and Rolling-opt proposed by Gmira et al. 2021a), and three RL approaches (AMDRGM-RL, NN-RL, and GTA-RL), demonstrating that DGTA-RL can address time-dependent and stochastic travel time in real-time. DGTA-RL outperforms heuristic-based rolling approaches in large-scale DTSP-TDS. DGTA-RL remains the best

approach among the three RL approaches in various scenarios with varying customer numbers and time intervals. Consequently, the DGTA-RL approach outperforms the heuristic-based rolling approaches with respect to solving time and solution quality for large-scale DTSP-TDS. Our main contributions are as follows:

(1) We propose a RL approach that enables real-time optimization in the dynamic and stochastic environment, allowing for the fast generation of good solutions for the DTSP-TDS.
(2) The embedding layer can process the time-dependent travel times by embedding the time interval and the customer coordinates. The numerical study demonstrates that the embedding layer and the dual attention layer, in cooperation with the temporal decoder, can reduce computational complexity and effectively solve large-scale DTSP-TDS from a temporal perspective.
(3) We add the novel dynamic encoder to select the corresponding representations according to the arrival time for each node. The dynamic encoder strengthen the connection between the nodes coordinates and the arrival times. The dynamic encoder can improve the ability of DGTA-RL in dealing with the stochastic travel time.
(4) The DGTA-RL has excellent generalization capability. The numerical study demonstrates that once the DGTA model has been trained in a multiple scenarios, it can efficiently generate tour for other various scenarios with different customer numbers and different uncertainty levels of stochastic travel times.

The remainder of this paper is organized as follows: Section 2 reviews related literature on time-dependent travel times and reinforcement learning. Section 3 introduces the DTSP-TDS and its corresponding Markov decision process. Section 4 describes the framework of the reinforcement learning algorithm. In Section 5, we detail the architecture of the DGTA model. Section 6 conducts numerical studies and analyzes the results by comparing the advantages of DGTA-RL with other approaches. Finally, Section 7 presents conclusions and promising future directions.

2. Related literature

To give a comprehensive overview of the TSP-TD, this section also reviews its generalized problem, the Vehicle Routing Problem with Time-Dependent Travel Times (VRP-TD), where the time-dependent travel times is considered in the same way as in the TSP-TD. We first discuss general approaches for solving the VRP-TD, particularly looking into the static and the dynamic variant, and then dive deeper into RL approaches for the time-dependent travel time.

2.1. Vehicle routing problem with time-dependent travel times

In the VRP-TD, vehicles must be assigned to customers or re-positioned on a grid, considering the time-dependent travel times. Depending on whether the realized travel time and the tour can be updated in the operating stage, we can divide the approach into dynamic or static approaches. In the static approach for VRP-TD, the route is planned in the planning stage and cannot be adjusted during the operating stage. In the dynamic approach for VRP-TD, the planned route can be dynamically adjusted during the operating stage.

Time-dependent travel time can be classified as deterministic or stochastic based on the availability and level of uncertainty in the information (Pillac et al., 2013). In problems with deterministic travel time, the actual travel time is equal to the expected travel time. In problems with stochastic travel times, we may only know the expected future travel times, not the actual realized travel times. In the VRP-TD, the stochastic travel time is captured in two ways: (1) Assuming known its distributions (Mao and Shen, 2018; Zhang et al., 2021); (2) Estimating its by the historical data (Schilde et al., 2014).

The approaches for solving VRP-TDs are divided into three main categories: (meta-)heuristic algorithm (Heuristics), Branch & Bound algorithm (B&B), and Markov Decision Process (MDP). We also classify the RL approach as a MDP-based method. An overview of relevant studies considering time-dependence or stochasticity of the travel time is presented in Table 1. In the following subsections, we will individually discuss studies that utilize either a static or dynamic approach.

2.1.1. Static approach for time-dependent travel times

In the problem with deterministic travel times, the actual travel time can be known in the planning stage (Cordeau et al., 2014; Arigliano et al., 2019; Sun et al., 2020; Gmira et al., 2021b). In addition to time-dependent travel time, some studies also consider some other time-dependent factors such as time-dependent traffic congestion (Xiao and Konak, 2016), variable time windows (Franceschetti et al., 2017), variable service time (Zhang et al., 2022), and time-dependent travel speeds (Pralet, 2023). If the distance between two nodes remains unchanged, we still regard the VRP with time-dependent travel speed (Avraham and Raviv, 2020) as a variant of VRP-TD. Their studies aim to minimize costs such as travel time, emissions, and driver wages using heuristics or branch & bound methods. Because the expected travel time is known in the planning stage, which may differ from its realized travel time in the operating stage (Laporte et al., 1992), some studies assume some distributions to model the stochastic travel time. For example, Taş et al. (2014b,a, 2013) assume a gamma distribution for the stochastic travel time and plan a flexible routing for the customers. Wang et al. (2023) consider the stochastic travel time in the planning stage and use a branch-cut-and-price method to offer a more reliable service for the customer with a soft time window. However, the distribution of stochastic travel times does not fully encapsulate the stochastic nature. Some researchers adopt data-driven approaches to thoroughly investigate factors influencing stochastic travel times (Avraham and Raviv, 2020). For example, Mao and Shen (2018) use RL approach to analyze historical data to categorize traffic flow into ‘congested’ or ‘uncongested’ states. Unlike heuristic and branch-and-bound approaches, the RL approach uses neural networks to learn traffic patterns and capture information on stochastic travel times (Avraham and Raviv, 2020).

Table 1
An overview of some studies considering time-dependence or stochasticity of the travel time.

Study	Strategy	Time_dependent travel time	Model for stochastic travel time	Method
Taş et al. (2013)	Static		Gamma distribution	Heuristics
Cordeau et al. (2014)	Static	✓	–	Branch & Bound
Taş et al. (2014a)	Static	✓	Gamma distribution	Heuristics
Taş et al. (2014b)	Static		Gamma distribution	Branch & Bound
Xiao and Konak (2016)	Static	✓	–	Heuristics
Franceschetti et al. (2017)	Static	✓	–	Heuristics
Mao and Shen (2018)	Static	✓	Normal distribution	RL(MDP)
Arigliano et al. (2019)	Static	✓	–	Branch & Bound
Avraham and Raviv (2020)	Static	✓	Normal distribution of speed	Branch & Bounds & Heuristics
Gmira et al. (2021b)	Static	✓	–	Heuristics
Pralet (2023)	Static	✓	–	Heuristics
Wang et al. (2023)	Static	✓	Normal distribution	Branch & Bound
Serrano et al. (2024)	Static	✓	Gamma distribution	Branch & Bound
Ichoua et al. (2003)	Dynamic	✓	–	Heuristics
Taniguchi and Shimamoto (2004)	Dynamic		–	Heuristics
Haghani and Jung (2005)	Dynamic	✓	–	Branch & Bound
Kim et al. (2005)	Dynamic		Normal distribution	MDP
Chen et al. (2006)	Dynamic	✓	–	Heuristics
Chang et al. (2009)	Dynamic	✓	Normal distribution	Heuristics
Schilde et al. (2014)	Dynamic	✓	Data_based estimation	Heuristics
Binart et al. (2016)	Dynamic		Discrete triangular distribution for speed	Heuristics
Bono et al. (2020)	Dynamic		Continuous bimodal distribution of speed	RL(MDP)
Gmira et al. (2021a)	Dynamic	✓	–	Heuristics
Zhang et al. (2021)	Dynamic		Normal distribution	RL(MDP)
Levering et al. (2022)	Dynamic		Stochastic processes in MDP	MDP
Liu et al. (2022)	Dynamic	✓	Limiting distribution of traffic condition	RL(MDP)
Lee et al. (2022)	Dynamic	✓	–	Heuristics
Guo et al. (2023)	Dynamic	✓	–	RL(MDP)
This study	Dynamic	✓	Gamma distribution	RL(MDP)

Note: The "–" in the column of "Model for stochastic travel time" indicates that the travel time is considered deterministic.

2.1.2. Dynamic approach for time-dependent travel times

In the planning stage, expected travel times cannot fully capture the stochastic nature of actual travel times, making it essential to incorporate real-time traffic conditions to enhance routing efficiency and service levels. In a dynamic approach, travel times are gradually revealed over time, allowing for adaptive routing adjustments during the operational stage. Several studies have explored different approaches for solving the Dynamic VRP with Time-Dependent and Deterministic travel times (DVRP-TDD), where future travel times are inaccessible at the beginning of the operating stage (Ichoua et al., 2003; Taniguchi and Shimamoto, 2004; Haghani and Jung, 2005; Chen et al., 2006; Lee et al., 2022; Guo et al., 2023).

In the DVRP-TDD, the travel times are known only for the current departure, while the travel times for future departures remain uncertain. This underscores the necessity for a more precise embedding of the uncertainties inherent in real-world scenarios, which gave rise to the Dynamic VRP with Time-Dependent and Stochastic travel times (DVRP-TDS). In the DVRP-TDS, the distributions of the travel times are known in the planning stage and stochastic travel times are revealed during the operating stage. A heuristic-based rolling approach can generate an initial plan based on historical data during the planning stage and provide an heuristic approach to handle stochastic travel times during the operational stage (Binart et al., 2016). Furthermore, Schilde et al. (2014) considered a dynamic dial-a-ride problem with dynamic customers and stochastic travel times by a heuristic-based rolling approach. They found that exploiting stochastic travel times in the planning stage led to significant improvements under certain conditions. In addition to the rolling approach based on heuristics, DVRP-TDS can also be modeled in an MDP and solved by a dynamic programming (Ulmer et al., 2019). With the dynamic programming, they find a policy for sequential decision-making that maximizes the expected rewards while dynamically considering the stochastic nature of travel times. Levering et al. (2022) uses a continuous-time MDP to develop an adaptive approach for the dynamic shortest path problem, considering the stochastic travel time. Liu et al. (2022) uses a RL approach to solve a dynamic flying sidekick traveling salesman problem with stochastic travel times considering truck–drone collaborative operations. Unlike the heuristic-based rolling approach, the RL approach do not require an initial plan and are implemented online using dynamic decision rules during the operating stage.

Considering stochastic travel times increases the complexity of the DTSP-TDS (Chang et al., 2009). As the size of the problem increases, the computational complexity of the heuristic algorithm may grow exponentially, leading to long solving times (Arigliano et al., 2019; Pralet, 2023; Cordeau et al., 2014; Gmira et al., 2021b; Franceschetti et al., 2017; Xiao and Konak, 2016). Long solving times render these approaches unsuitable for time-sensitive applications or real-time decision making (Mao and Shen, 2018). Some standard optimization procedures struggle to process time-dependent travel times, leading to suboptimal solutions when considering dynamic traffic conditions in the real world (Ehmke et al., 2012; Gmira et al., 2021a).

2.2. RL approach for time-dependent travel times

In recent years, the RL approach has been used to solve the TSP (Zhang et al., 2023) and its variant problem (Yan et al., 2022; Li et al., 2021b,a; Zhang et al., 2020). The main advantage of using the RL approach to solve DTSP-TDS is that the model can learn to generalize the knowledge of stochastic travel time extracted from many instances (Bono et al., 2020). Mao and Shen (2018) studied the adaptive routing problem in stochastic time-dependent networks and found that the solving time grows linearly with the amount of input data. As a result, the RL approach exhibits time-saving and data-driven superiority compared with time-consuming heuristics (Zhao et al., 2020; Zhang et al., 2020) and model-based dynamic programming methods (Mao and Shen, 2018).

In addition, the RL approach has been effectively applied to real-world logistic management challenges. For example, James et al. (2019) developed a RL approach to generate online tours in the Cologne transport network. Tang et al. (2020) designed a RL framework for managing automated electric taxi fleets using actual Beijing taxi data. Basso et al. (2022) studied the dynamic stochastic electric vehicle routing problem to minimize the expected energy consumption and tested it on the map of Luxembourg. These studies explored the feasibility of applying RL approach to solve real-world problems, but the time-dependent travel times in their problems are static. Some applications solve dynamic real-world problems with stochastic travel times to demonstrate the practicality of their solutions, such as the global synchro-modal shipment matching problem (Guo et al., 2022) and the traveling salesman problem with the truck–drone delivery (Liu et al., 2022). Although these RL approaches can make dynamic decisions according to changing traffic conditions, the fully-connected neural network they used cannot efficiently handle the stochastic travel time.

RL approach with some well-designed neural network is potential to solving the TSP due to its ability to learn complex decision policies (Hildebrandt et al., 2022). For example, Vinyals et al. (2015) introduced the pointer network, which integrates the attention mechanism into the sequence model (Sutskever et al., 2014) to enable the decoder to focus on some vital information. Bello et al. (2016) combined the RL algorithm with the pointer network to solve the TSP. They extended the pointer network by using the reinforcement algorithm proposed by Williams (1992) to train the pointer network. After that, Kool et al. (2018) introduced a multi-attention mechanism to pass weighted information among different nodes, considering the influence of adjacent structures on solutions. Compared with the fully-connected neural network, the pointer network (Vinyals et al., 2015) and multi-attention mechanism (Kool et al., 2018) have fewer parameters and higher performance. They possess more efficient capabilities to learn the complex decision policy, laying the research foundation for solving complex variants of the TSP, such as the DTSP-TDS. Gunarathna et al. (2022) developed a novel GTA model based on the pointer network and the multi-attention mechanism to handle the dynamic TSP with time-dependent node location. Although they did not consider the time-dependent and stochastic travel time, their architecture can handle the time-dependent feature. This paper introduces the DGTA model based on the GTA model to address the DTSP-TDS.

Due to the high uncertainty in time-dependent and stochastic travel times, the scale of the DTSP-TDS tends to be significantly large if there are many customers and a high updating frequency. Therefore, it is necessary to balance the trade-off between a large input of the time-dependent travel time matrix and large computational memory in the DTSP-TDS. To avoid the difficulty caused by travel time matrix, some researchers have incorporated traffic patterns at the nodes instead of the matrix of time-dependent and deterministic travel times (Bono et al., 2020; Zhang et al., 2021). However, the traffic patterns do not completely capture the stochasticity of the travel time information, and only a few studies on time-dependent travel times incorporate them. To efficiently input the time-dependent travel time, Guo et al. (2023) developed two deep attention models with dimension-reduction and gate mechanisms to solve the DVRP-TDD. Their dimension-reduction mechanism can handle travel time in a more memory-efficient way. However, their RL approach may not be effective in solving the DTSP-TDS because the travel times they considered are deterministic. Our DGTA model omits the input of stochastic travel times in the dual attention layer but updates the realized travel times in the dynamic encoder. This approach enables the simultaneous consideration of time-dependent and stochastic travel times while using fewer neural network parameters to train the policy.

3. Problem formulation in MDP

In the DTSP-TDS, the goal is to find the tour with the shortest duration for a single vehicle to visit a given set of customer nodes. The vehicle starts from the depot and must return to it after visiting all customers exactly once. We refer to the customers and the depot as nodes. The travel time between nodes is time-dependent and varies based on the departure time from the node. The expectation and standard deviations of the stochastic travel times are assumed to be known and unchanged during the operating stage. Before making a dynamic decision on the next node to visit, the actual travel times at the current time are realized. This realized travel time is assumed to be fixed at the current decision step. The objective is to minimize the total tour duration of the vehicle, which equals the time from start to return to the depot.

We discretize the horizon of the operating stage (operating horizon) into a series of time intervals represented by $\mathcal{T} = \{1, 2, \dots, T\}$, where T is the total number of time intervals. Each time interval is of equal length, denoted by Δt. The travel times are assumed to be fixed within a time interval. We define $\mathcal{N} = \{0, 1, 2, \dots, N\}$ as the set of nodes, where node 0 represents the depot, and nodes 1 to N represent customers. The location of the node i is given by the horizontal coordinate x_i and vertical coordinate y_i. There are an expected travel time $\hat{c}_{ij}(t)$ between two nodes $i, j \in \mathcal{N}$ in time interval $t \in \mathcal{T}$. We model DTSP-TDS as an MDP with several decision steps, where the tour is the sequence of decisions (actions) made at each step. We use d to represent the index of the current decision step. The MDP of DTSP-TDS is formulated with the following components:

State S The state consists of the initial state S_0 and subsequent states S_d. The initial state is the known information in the planning stage: the coordinates of each node, represented as a vector $(x_i, y_i)_{i\in\mathcal{N}}$ and the expected travel time matrix in all time intervals $\left(\hat{c}_{ij}(t)\right)_{i,j\in\mathcal{N},t\in\mathcal{T}}$. The subsequent state is the realized information during the operating stage, which contains:

- The vector of visited nodes, $\mathbf{V}_d = (v_0, \ldots, v_N)$, where $v_i = 1$ if node i has been visited, and $v_i = 0$ otherwise.
- The time of the current decision step, τ_d.
- The matrix of travel times realized at the current decision step, $\left(c_{ijd}\right)_{i,j\in\mathcal{N}}$.
- The vector of arrival times at the nodes, $\mathbf{A}_d = \left(A_d(j)\right)_{j\in\mathcal{N}}$.

The state at the initial decision step is set as $\tau_0 = 0$, $A_d(i) = c_{0,i,d+1}, i \in \mathcal{N}$.

Decision Variable X_d^π The decision is next node to visit determined by policy π.

State Transition The subsequent state S_{d+1} at decision step $d+1$ is determined by the current state S_d and the current decision X_d^π. The visited node and the next time in the subsequent state S_{d+1} is updated as follows:

$$v_{X_d^\pi} := 1 \tag{1}$$

$$\tau_{d+1} := A_d(X_d^\pi) \tag{2}$$

where the time interval of the next decision step is computed as $t_{d+1} = \left\lfloor \tau_{d+1} \,/\, \Delta t \right\rfloor$ is, so the expected travel time at the next decision step is $\left(\hat{c}_{ij}(t_{d+1})\right)_{i,j\in\mathcal{N}}$. If $t_d = t_{d+1}$, then the travel time in the next decision step remains consistent with the previous step: $c_{ij,d+1} := c_{ijd}$; otherwise $(t_d < t_{d+1})$, the travel times at the subsequent state are realized from a gamma distribution:

$$c_{ij,d+1} \sim \Gamma\left(\frac{\hat{c}_{ij}(t_{d+1})}{\beta}, \beta\right) \qquad \forall i, j \in \mathcal{N} \tag{3}$$

where β is the scale of the gamma distribution, which reflects the uncertainty of the travel time. $\hat{c}_{ij}(t_{d+1})$ represents the time dependence of the travel time. Based on the realized travel time, the arrival times from the next node to other nodes are computed as:

$$A_{d+1}(j) := \tau_{d+1} + c_{X_d^\pi, j, d+1} \qquad \forall j \in \mathcal{N} \tag{4}$$

Objective function The objective function (5) is to minimize the expected tour duration generated by the policy π. The duration of the tour is evaluated by stochastic travel times in each decision step, where $c_{X_d^\pi, X_{d+1}^\pi, d}$ represents the realized travel time from node X_d^π to node X_{d+1}^π in the decision step d. The number of decision steps is equal to the number of nodes in the problem.

$$\min_\pi \mathbb{E}\left[\sum_{d=0}^{N-1} c_{X_d^\pi, X_{d+1}^\pi, d}\right] \tag{5}$$

By formulating the DTSP-TDS as an MDP, we can apply the RL approach to find a near-optimal policy for generating decisions over time.

4. Reinforcement learning algorithm

We use $\mathbf{X}^\pi = (X_0^\pi, \ldots, X_{N-1}^\pi)$ to represent the tour in which nodes are visited. The reinforcement learning algorithm optimizes the policy π to generate a tour $\mathbf{X}^\pi$ for a instance s in two steps. First, we use the DGTA model (described in Section 5) with parameters θ to generate the probability of the tour $\mathbf{X}^\pi$ for instance s, which we denote by $p_\theta(\mathbf{X}^\pi|s)$. $p_\theta(\mathbf{X}^\pi|s)$ is computed by the multiplication of the likelihoods of the decisions X_d^π in state S_d, for each decision step:

$$p_\theta(\mathbf{X}^\pi|s) = \prod_{d=0}^{N-1} p_\theta\left(X_d^\pi|S_d, s\right) \tag{6}$$

Then, we use a sampling strategy at each decision step to select the next node based on the probabilities outputted from the DGTA model. Due to uncertainty in travel times, there is a high variance in the rewards of the tour obtained by the sampling strategy. To mitigate the issue of high variance with the different instances, we use a baseline model (with parameters $\bar{\theta}$) and a greedy strategy to generate the baseline tour ($\mathbf{X}^{\bar{\pi}}$). The baseline model is a copy of the DGTA model with less frequently updated parameters (Gunarathna et al., 2022). Unlike the sampling, the greedy strategy selects the next node with the highest probability.

To optimize the parameters θ, we employ the REINFORCE algorithm (Williams, 1992) as detailed in Algorithm 1. Our training procedure consists of conducting I iterations, where in each iteration the model is trained in multiple batches, each of which contains B instances. We denote an instance from a batch as s_b. Subsequently, we employ Monte Carlo sampling to approximate the gradient $\mathcal{J}(\theta)$ based on these instances. We use the costs defined in the objective function (5) as the loss associated with a tour $\mathbf{X}^\pi$, denoted by $\mathcal{L}(\mathbf{X}^\pi)$. We then utilize the gradient $\nabla\mathcal{J}(\theta)$ to update the parameters θ in the current model. The baseline policy will be updated with the current policy if their loss is not significantly different, as determined by a paired t-test performed on validation instances with a significance level set at 5%.

5. Dynamic graph temporal attention model

The Dynamic Graph Temporal Attention (DGTA) model is a key component of DGTA-RL, which captures time-dependent travel times in the stochastic environment and makes sequential decisions. The DGTA model consists of a spatial–temporal encoder, a dynamic encoder and a spatial–temporal decoder (Fig. 2). There are an embedding layer and a dual attention layer with multiple blocks in the spatial–temporal encoder. We refer to the information flow in the architecture as "representation", such as the output

Algorithm 1 REINFORCE algorithm

Input: Iteration number I, batch size B for each batch, clipping threshold c
Generate the validation instances S^{valid}; the training instances $S_{\text{iter}}^{\text{train}}$ for each iteration.
Initialize parameters $\theta \leftarrow \theta_0$, $\bar{\theta} \leftarrow \theta_0$, learning rate η
for iter = 1 to I **do**
  for each batch in $S_{\text{iter}}^{\text{train}}$ **do**
    Generate tour by the policy π for each instance: $\mathbf{X}^{\pi}$.
    Generate baseline tour by the policy $\bar{\pi}$ for each instance: $\mathbf{X}^{\bar{\pi}}$.
    $\nabla \mathcal{J}(\theta) \leftarrow \frac{1}{B} \Sigma_{b=1}^{B} [\mathcal{L}(\mathbf{X}^{\pi}|s_b) - \mathcal{L}(\mathbf{X}^{\bar{\pi}}|s_b)] \nabla_{\theta} \log p_{\theta}(\mathbf{X}^{\pi}|s_b)$
    Apply gradient clipping: $\nabla \mathcal{J}(\theta) \leftarrow \text{clip}(\nabla \mathcal{J}(\theta), -c, c)$
    Update the policy parameters: $\theta \leftarrow \theta + \eta \nabla \mathcal{J}(\theta)$
  end for
  if t-test$(\mathcal{L}(\mathbf{X}^{\pi}), \mathcal{L}(\mathbf{X}^{\bar{\pi}})|S^{\text{valid}}) < 5\%$ **then**
    $\bar{\theta} \leftarrow \theta$
  end if
end for
Output: The parameters in the DGTA model θ

of the embedding layer (denoted $\mathbf{H}^{(0)}$ in Fig. 2). First, the node's coordinates and the time interval index are encoded into spatial–temporal representation in the spatial–temporal encoder. Then, the dynamic encoder encodes the realized travel time and integrates it with the spatial–temporal representation. After that, the spatial–temporal decoder utilizes spatial and temporal pointers to extract the representations. Finally, the probability layer in the spatial–temporal decoder output the probability of visiting the next node, as previously described in Eq. (6). For simplicity, Fig. 2 shows the dual attention layer and the attention layer in dynamic encoder in one block.

5.1. Spatial–temporal encoder

The embedding layer is the first layer of the spatial–temporal encoder to embed the node coordinates and the index of the time intervals. After that, the embedding is encoded into the spatial–temporal representation in the dual attention layer.

5.1.1. Embedding layer

We want to represent the expected travel time by the time interval of the departure time, and coordinates of the departure and destination nodes. However, the node coordinate is a spatial feature, and the time interval is a temporal feature, so they have different scales. As a result, the node coordinate and the time interval cannot be connected directly. We need to expand the features of node coordinates and time intervals separately. Specifically, we add a temporal dimension in the first dimension of the node coordinates vector $\mathbf{G} = (x_i, y_i)_{i \in \mathcal{N}} \in \mathbb{R}^{N \times 2}$, expand it into $\mathbf{G}^* \in \mathbb{R}^{T \times N \times 2}$. In the same way, the time interval vector $\mathbf{T} = (1, 2, \ldots, T) \in \mathbb{R}^{T \times 1}$ is also expanded into $\mathbf{T}^* \in \mathbb{R}^{T \times N \times 1}$ along the second dimension. Then, $\mathbf{G}^*$ is concatenated with $\mathbf{T}^*$ along the third dimension and transformed by Sigmoid activation function:

$$\mathbf{H}^{(0)} = \text{Sigmoid}\left((\mathbf{G}^* \parallel \mathbf{T}^*) \cdot W^E\right) \tag{7}$$

where $W^E \in \mathbb{R}^{3 \times D}$ is a trainable parameter, D refers to the scale of the hidden dimension in the embedding layer, and $\parallel$ means the combine these two vectors along the last dimension into a single vector. The output from the embedding layer with dimension $T \times N \times D$ is represented as $\mathbf{H}^{(0)} = \left(h_{i,t}^{(0)}\right)_{i \in \mathcal{N}, t \in \mathcal{T}}$. The embedding layer compresses the time-dependent time interval initially represented as a three-dimensional matrix, into a two-dimensional embedding, which is also compatible with the subsequent dual attention layer.

Dual attention layer

The dual attention layer captures the spatial and temporal relationships of travel times between nodes through L_1 blocks. In each block, the spatial and temporal attention mechanisms with M heads operate in parallel, and their outputs are integrated into a new representation for the next block. In the spatial and temporal attention mechanisms, we use the same dimensions for query, key, and value representations as $D^m = D/M$ at each head, so the scale for query, key, and value parameters is $\mathbb{R}^{D \times D^m}$. The spatial attention mechanism encodes the dependencies between nodes at each time interval. The representations of all nodes at a given time interval t, which is defined as $H_t^{(l,S)} := \left(h_{i,t}^{(l-1)}\right)_{i \in \mathcal{N}}$, are the input of the lth block:

$$\bar{H}_t^{(l,S)} := \mathbf{MHA}_l\left(H_t^{(l,S)}; \mathbf{W}^{(l,S)}\right) \tag{8}$$

where $\mathbf{W}^{(l,S)}$ represents the set of trainable parameters in the spatial attention mechanism. The superscript S means "Spatial", which is used to distinguish temporal attention (T). The details of the attention mechanism **MHA** are shown in Appendix A. Then, the

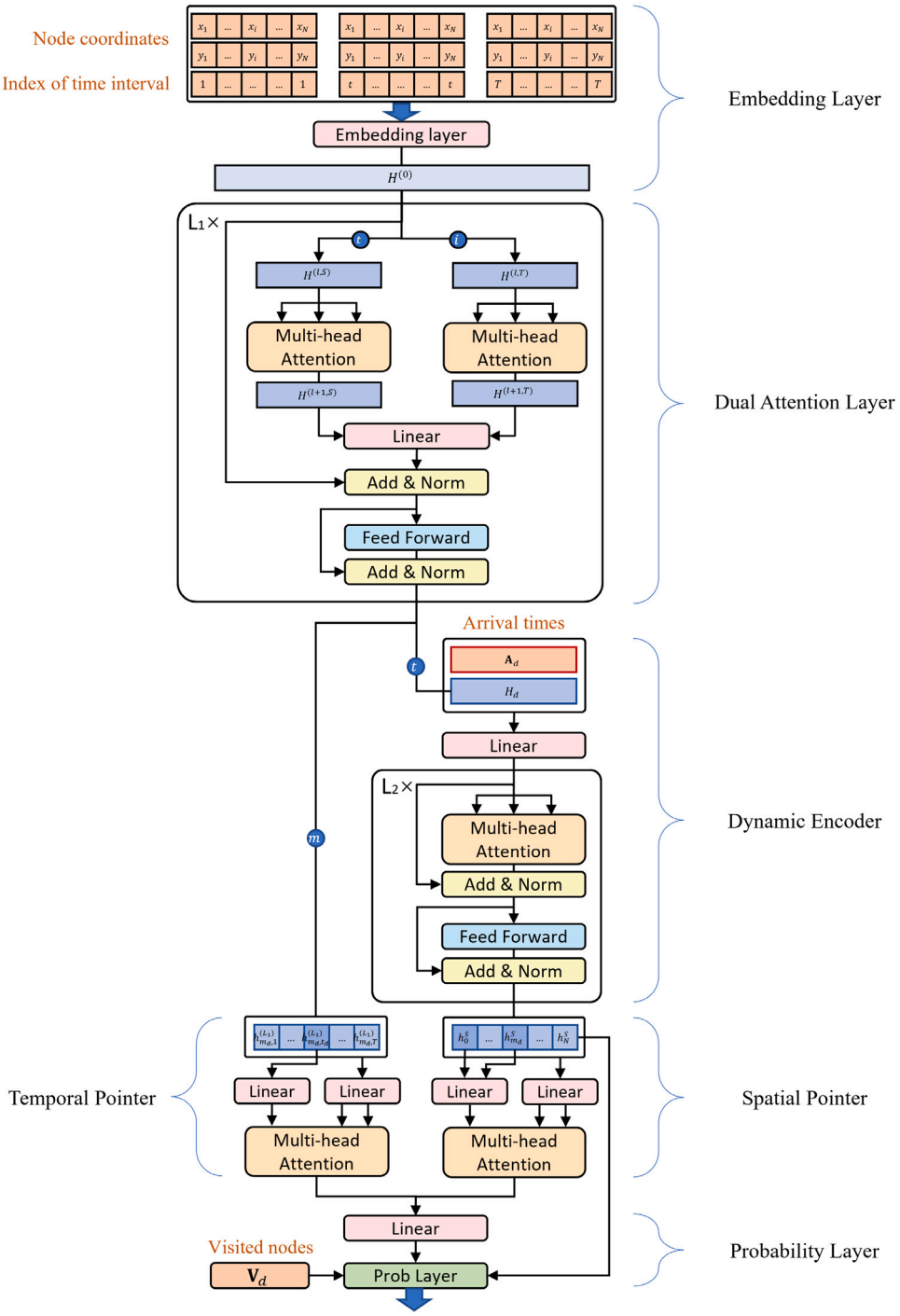

Fig. 2. The architecture of the DGTA model.

representations $\bar{H}_t^{(l,S)}$ at every time intervals are concatenated into $\bar{\mathbf{H}}^{(l,S)}$:

$$\bar{\mathbf{H}}^{(l,S)} := \text{concat}\left(\bar{H}_1^{(l,S)}, \bar{H}_2^{(l,S)}, \ldots, \bar{H}_T^{(l,S)}\right) \tag{9}$$

where concat() is the operator to combine the vectors along the first dimension to a single vector. The temporal attention mechanism encodes each node's dependencies between different time intervals in parallel. The representations of node i at all time intervals,

which is defined as $H_i^{(l,T)} := \left(h_{i,t}^{(l-1)}\right)_{t\in\mathcal{T}}$, are encoded with the trainable parameters $\mathbf{W}^{(l,T)}$ in the lth temporal attention mechanism, and then concatenated into $\bar{\mathbf{H}}^{(l,T)}$:

$$\bar{H}_i^{(l,T)} := \mathbf{MHA}_l\left(H_i^{(l,T)}; \mathbf{W}^{(l,T)}\right) \tag{10}$$

$$\bar{\mathbf{H}}^{(l,T)} := \text{concat}\left(\bar{H}_0^{(l,T)}, \bar{H}_1^{(l,T)}, \ldots, \bar{H}_N^{(l,T)}\right) \tag{11}$$

The $\bar{\mathbf{H}}^{(l,S)}$ and the transpose of $\bar{\mathbf{H}}^{(l,T)}$ are integrated by a trainable parameter $\mathbf{W}^{(l,I)} \in \mathbb{R}^{2D\times D}$, the integrated representation $\bar{\mathbf{H}}^{(l)}$ can provide spatial and temporal perspectives for the DTSP-TDS:

$$\bar{\mathbf{H}}^{(l)} := \left(\bar{\mathbf{H}}^{(l,S)} \parallel \left(\bar{\mathbf{H}}^{(l,T)}\right)^T\right) \cdot \mathbf{W}^{(l,I)} \tag{12}$$

Finally, a batch normalization ($\mathbf{BN}_l$) and a feed-forward layer ($\mathbf{FF}_l$) are applied to the integrated representation, as the transformer architecture (Vaswani et al., 2017):

$$\mathbf{H}^{(l)} = \mathbf{BN}_l\left(\mathbf{H}^{(l)} + \mathbf{FF}_l(\mathbf{BN}_l\left(\mathbf{H}^{(l)} + \bar{\mathbf{H}}^{(l)}\right))\right) \tag{13}$$

Consequently, the concatenation mechanism forwards $\mathbf{H}^{(l)}$, which is represented as $\left(h_{i,t}^{(l)}\right)_{i\in\mathcal{N},t\in\mathcal{T}}$, to the next block of the dual attention layer. The output of the dual attention layer is represented as $\mathbf{H}^{(L_1)} = \left(h_{i,t}^{(L_1)}\right)_{i\in\mathcal{N},t\in\mathcal{T}}$, which is defined as the spatial–temporal representation for the following dynamic encoder and spatial–temporal decoder.

5.2. Dynamic encoder

The spatial–temporal representation serves as the input for the dynamic encoder; however, not all parts of this representation are relevant to the current decision. The spatial–temporal representations is selected and encoded based on the current subsequent state from the environment. In the following dynamic encoder, spatial–temporal decoder and probability layer, we use the subscript d to denote the representations for the current decision step.

In the dynamic encoder, the spatial–temporal representation in the time interval of the current decision step t_d is combined into a related graph: $H_d := \left(h_{i,t_d}^{(L_1)}\right)_{i\in\mathcal{N}}$. We use $m_d = X_{d-1}^{\pi}$ to represent the current node for the simplification. The temporal representation is composed of the representations at the current node, which is represented as $H_d^T := \left(h_{m_d,t}^{(L_1)}\right)_{t\in\mathcal{T}}$. Then, the combination of the $H_d \in \mathbb{R}^{N\times D}$ and the vector of arrival time at the nodes $\mathbf{A}_d \in \mathbb{R}^{N\times 1}$ is transformed by a trainable parameter $W^S \in \mathbb{R}^{(D+1)\times D}$ into the representation $\bar{H}_d^S \in \mathbb{R}^{N\times D}$:

$$\bar{H}_d^S := (H_d \parallel \mathbf{A}_d) \cdot W^S \tag{14}$$

Then, an attention layer encodes the representation $\bar{H}_d^S$ by the multi-head attention mechanism with L_2 blocks (Appendix A). The process of lth block in the multi-head attention mechanism is as follow:

$$\hat{H}_d^{(l,D)} = \mathbf{BN}_l\left(H_d^{(l,D)} + \mathbf{MHA}_l(H_d^{(l,D)}; \mathbf{W}^{(l,D)})\right) \tag{15}$$

$$H_d^{(l+1,D)} = \mathbf{BN}_l\left(H_d^{(l,D)} + \mathbf{FF}_l(\hat{H}_d^{(l,D)})\right) \tag{16}$$

where the input of the initial block is $H_d^{(1,D)} := \bar{H}_d^S$, $\mathrm{W}^{(l,D)}$ is the trainable parameter in the lth block of the attention layer. The output from the dynamic attention is defined as a spatial representation for the following spatial–temporal decoder, $H_d^S := H_d^{(L_2,D)}$.

5.3. Spatial–temporal decoder

There are spatial and temporal pointers to compute spatial and temporal attention in the spatial–temporal decoder, separately. The spatial representation $H_d^S \in \mathbb{R}^{N\times D}$ is the input of the spatial pointer. The temporal representation $H_d^T \in \mathbb{R}^{T\times D}$ is the input of the temporal pointer. The outputs from the spatial and temporal pointers are integrated by an integration layer. Finally, the probability layer output the probability for the next visit node.

Spatial pointer

The spatial pointer analyzes the relevance between the current node and other nodes, especially the remaining customers and the depot. In the spatial representation $H_d^S = \left(h_i^S\right)_{i\in\mathcal{N}}$. We slack the representations of depot and current node into a spatial context representation $C_d^S \in \mathbb{R}^{1\times 2D}$:

$$C_d^S := h_0^S \parallel h_{m_d}^S \tag{17}$$

Then, we use the multi-head attention layer to analyze the connection between the context representation and the spatial representation. We use query $W_q^S \in \mathbb{R}^{2D\times D}$, key $W_k^S \in \mathbb{R}^{D\times D}$ and value $W_v^S \in \mathbb{R}^{D\times D}$ to transform the spatial representation:

$$Q_d^S = C_d^S W_q^S, K_d^S = H_d^S W_k^S, V_d^S = H_d^S W_v^S \tag{18}$$

Finally, the spatial attention for each node with their relevance to others is computed as follows:

$$h_d^S = \text{Softmax}\left(\frac{Q_d^S \cdot (K_d^S)^T}{\sqrt{D}}\right) \cdot V_d^S \tag{19}$$

Temporal pointer

Like the spatial pointer, the temporal pointer analyzes the representations at the current decision step with the other representations in the other time intervals, especially the future time intervals. We also use the multi-head attention layer to combine the context representation C_d^T with the temporal representation H_d^T. However, only the representation at the current decision step is considered in the context representation in the temporal pointer:

$$C_d^T := \{h_{m_d,t_d}^{(L_1)}\} \in \mathbb{R}^{1\times D} \tag{20}$$

After that, we use query $W_q^T \in \mathbb{R}^{D\times D}$, key $W_k^T \in \mathbb{R}^{D\times D}$ and value $W_v^T \in \mathbb{R}^{D\times D}$ parameters in the attention mechanism and compute the temporal attention h_d^T for each time interval with their relevance to others in the following equations:

$$Q_d^T = C_d^T W_q^T, K_d^T = H_d^T W_k^T, V_d^T = H_d^T W_v^T \tag{21}$$

$$h_d^T = \text{Softmax}\left(\frac{Q_d^T \cdot (K_d^T)^T}{\sqrt{D}}\right) \cdot V_d^T \tag{22}$$

Probability layer

After the temporal and spatial attention, the representations are integrated into a single attention $h_d \in \mathbb{R}^D$ by an integration layer with a trainable parameter $W^I \in \mathbb{R}^{2D\times D}$:

$$h_d := (h_d^S \parallel h_d^T) \cdot W^I \tag{23}$$

The masks $\mathbf{V}_d$ in the probability layer effectively prevent the model from repeatedly revisiting the served customers, ensuring the solution's feasibility. It is noted that the spatial representation H_d^S is skipped from the spatial pointer to the probability function (Eq. (24)). We use the "tanh" and "Softmax" activation functions to compute the selection probabilities for each node.

$$p_\theta\left(X_d | S_d\right) = \text{Softmax}\left[\tanh\left(\frac{h_d \cdot (H_d^S)^T}{\sqrt{D}}\right)\right] \cdot \mathbf{V}_d \tag{24}$$

6. Numerical study

The numerical study evaluates the performance of the DGTA-RL based on the instances of DTSP-TDS generated in Section 6.1. The iteration for the REINFORCE algorithm is determined based on the DGTA-RL performances in Section 6.2. With the determined iteration number, we evaluate the performance of DGTA-RL and the benchmark approaches over the simulated scenarios (Sections 6.3 to 6.6). In the REINFORCE algorithm, we fix the learning rate at $\eta = 0.001$ and apply a maximum L2 norm of 1.0 for gradient clipping. The batch size is set to 128. In the DGTA model, we configure $D = 128, M = 8, L_1 = 5, L_2 = 3$. The numerical study is conducted on the Snellius Cluster High-Performance Computer, specifically, a NVIDIA Quadro P5000 with 16 GB memory, and a Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40 GHz with 256 GB RAM.

6.1. Numerical setting

In the numerical study, we generate DTSP-TDS instances based on the simulation proposed by Cordeau et al. (2014). In each simulation, the depot is located in the center of a square at coordinates [50, 50]. First, an angle is randomly drawn from a uniform distribution over $[0, 2\pi]$ to determine the direction of the customer. Then, the Euclidean distance from the depot is generated according to a normal distribution with a given standard deviation, σ. Customers are generated within a $[0, 100]^2$ square. We calculate the expected travel time $\hat{c}_{ij}(t)$ based on the expected speed $\hat{v}_{ij}(t)$ and the Euclidean distance $d(i, j)$ as: $\hat{c}_{ij}(t) = \frac{d(i,j)}{\hat{v}_{ij}(t)}$. The expected speed $\hat{v}_{ij}(t)$ is determined by the congestion level at the departure time, the congestion factor, and the maximum speed: $\hat{v}_{ij}(t) = b_{h(t)} \cdot \delta_{z(i,j),h(t)} \cdot u_{z(i)}$, where $b_{h(t)}$ represents the congestion level in time interval t, and $h(t)$ indicates the congestion period of the time interval t. Since the entire operating horizon is equally divided into three congestion periods, $b_{h(t)}$ are set to 0.50, 1.00, and 0.50 for congestion periods 1, 2, and 3, respectively. $\delta_{z(i,j),h(t)}$ represents the congestion factor, where $z(i, j)$ is the congestion zone for nodes i and j. These congestion zones are defined by concentric circular areas centered at the depot, with the boundaries of these zones formed by two circles of radius 20 and 40 units, respectively. The maximum speeds $u_{z(i)}$ are set to 26, 36, and 50 for congestion zones 1, 2, and 3, respectively. We use the congestion factors of the traffic pattern "A" from Cordeau et al. (2014). The customer locations vary in different instances, so their travel times correspondingly differ. We simulate the realization of the stochastic travel time using a gamma distribution with the mean equal to the expected travel time. Our numerical study evaluates the DGTA-RL approach in various scenarios with different number of nodes ($N = 10, 20, 30, 40, 50$), time interval ($T = 6, 12, 24$), standard deviation of the normal distribution ($\sigma = 5, 10, 15, 20, 25$), and scale of the gamma distribution ($\beta = 0.6, 0.8, 1.0, 1.2, 1.4$). We maintain the length of the operating horizon as $\Delta t \cdot T = 240$, so Δt changes with T.

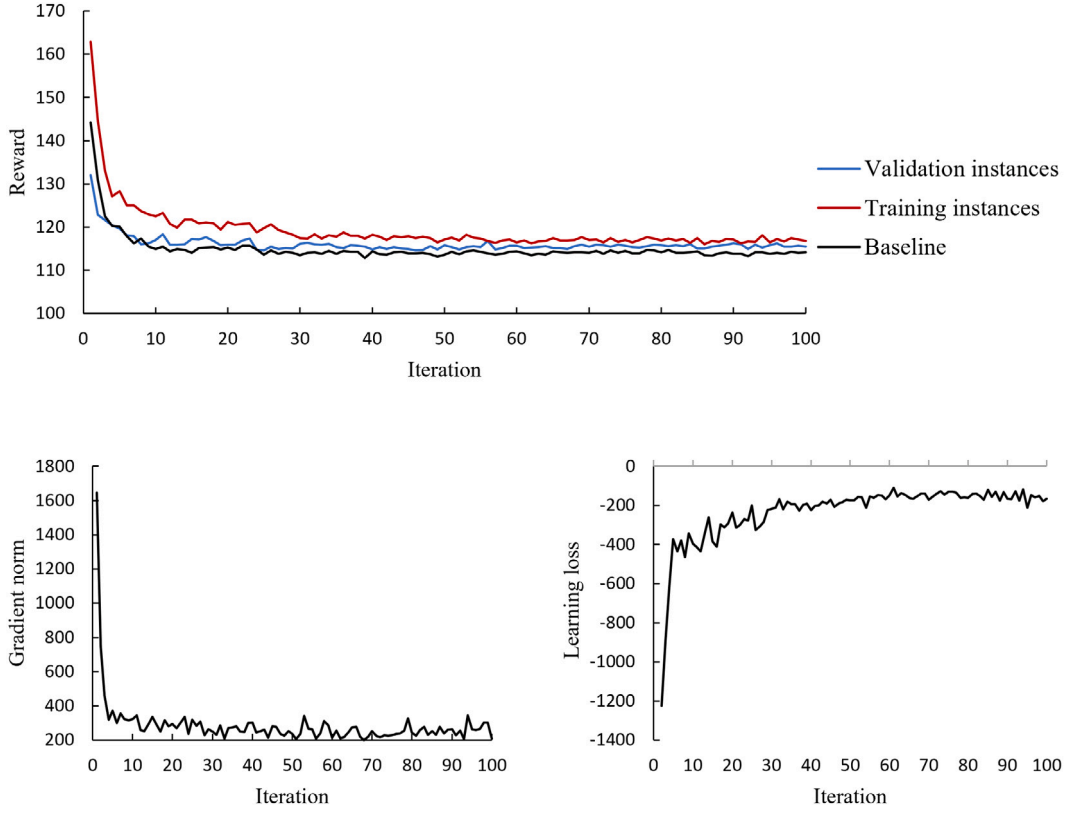

Fig. 3. Learning curve of DGTA-RL.

6.2. Learning curve and convergence

This section discusses the learning curve of DGTA-RL in the scenario with $(N, T, \sigma, \beta) = (30, 12, 15, 1.0)$. We train our DGTA model within 100 iterations; each iteration contains 1280 training instances. After each iteration, a paired t-test is conducted on 1280 validation instances to assess the model's performance. Fig. 3 presents the learning curves in reward, the L2-norm of the gradient (gradient norm), and the learning loss. The learning loss is defined as the difference between the rewards obtained from the policy π and the baseline policy $\bar{\pi}$, calculated as $\mathcal{L}(\mathbf{X}^{\pi}) - \mathcal{L}(\mathbf{X}^{\bar{\pi}})$. The gradient norm reflects the scale of the model parameters that need to be adjusted during training.

The learning curve of reward shows that the "Validation instances" values consistently decrease until epoch 50, after which they stabilize at approximately 115. The trend of gradient norm represents the magnitude of the gradient is initially high during early training, indicating greater updates to the model parameters. The learning loss, which are negative, initially decrease in magnitude, indicating that the model is learning. Around 50th epoch, the gradient norm and learning loss have significantly stabilized, suggesting that further improvements are minimal. Based on this convergence, we will train the model for 50 iterations in subsequent numerical studies, with each iteration containing 1280 training instances.

6.3. Comparison on the STSP-TDD

The difficulty in solving the DTSP-TDS lies in time-dependent travel times and stochastic changes. This section investigates DGTA-RL's ability to address time-dependent travel time in the STSP-TDD. In the STSP-TDD, the realized travel time is aligned with the expected travel time in the simulated instance of DTSP-TDS. The difference in travel time on a given arc between STSP-TDD and DTSP-TDS is shown in Fig. 4, where the length of a time interval is $\Delta t = 20$. The dot represents the current time, the solid lines represent the actual travel times realized at the current time, and the dashed lines represent actual travel times in the future, which is unknown at the current time. Fig. 4(a) shows the available information in STSP-TDD, where we know the time-dependent travel times at the beginning of the planning; Fig. 4(b) shows the available information in DTSP-TDS, where we know the realized travel times before the current time and the probability distribution of stochastic travel times in the future.

We compare DGTA-RL with the optimal solution from MIP, heuristic-based rolling approaches, and RL approaches in various scenarios with different numbers of nodes ($N = 10, 20, 30, 40, 50$) and time steps ($T = 6, 12, 24$), while the normal distribution and gamma distribution remain the same, $(\sigma, \beta) = (15, 1.0)$. The STSP-TDD can be formulated in the Mixed Integer Programming (MIP)

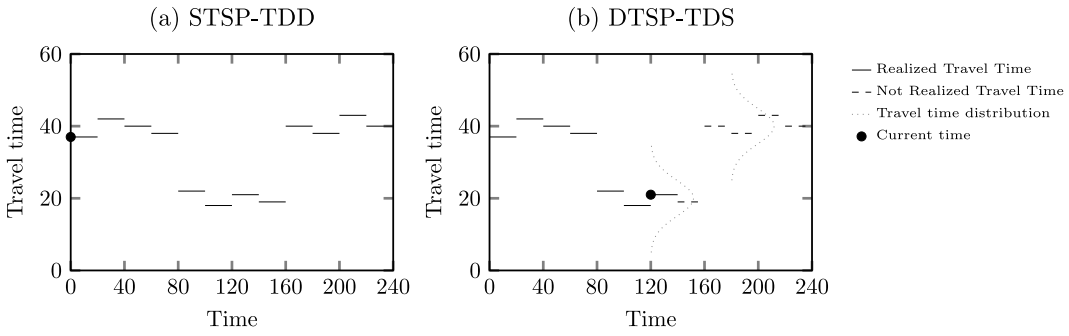

Fig. 4. The diagrams of STSP-TDD and DTSP-TDS.

and solved by the solver Gurobi. The heuristic-based rolling approaches include a greedy heuristic (Greedy) and a novel tabu search (GmiraTS, proposed by Gmira et al. 2021b). The RL approaches we compared include a state-of-the-art RL approach (AMDRGM-RL, proposed by Guo et al. 2023), and a basic RL approach with a fully-connected neural network (NN-RL). To ensure a fair comparison, AMDRGM-RL and NN-RL are trained by the REINFORCE algorithm with the same hyperparameters. Here are the details of these heuristic-based rolling approaches and RL approaches.

- MIP: The Mixed-Integer Programming (MIP) of STSP-TDD is solved using the Gurobi solver.
- Greedy: A greedy heuristic generates the tour by a step-by-step policy, selecting the next node with the shortest travel time at each step.
- GmiraTS: A tabu search heuristic proposed by Gmira et al. (2021b) to solve SVRP-TDD. To apply it to solve STSP-TDD, we removed the constraints for the capacity and time window. In this heuristic, we employed 2-opt exchange operators as the neighborhood structure.
- DGTA-RL: The DGTA-RL is proposed to solve DVRP-TDD in MDP, so the travel times at every time interval cannot be inputted and analyzed simultaneously. We still generate the tour dynamically, where the routing is made only based on the current travel time at each step.
- AMDRGM-RL: A RL approach with a deep attention model (AMDRGM model) to solve DVRP-TDD in MDP (Guo et al., 2023). The AMDRGM model they introduced is based on the RL model of Kool et al. (2018) with novel a dimension-reduction mechanism and a gate mechanism.
- NN-RL: A RL approach using a deep neural network with three fully-connected layers as the agent.
- GTA-RL: The GTA-RL proposed by Gunarathna et al. (2022) to solve the dynamic TSP with time-dependent node locations. The GTA model consists of a spatial–temporal attention layer, a temporal decoder, and a multi-head attention layer.

Table 2 presents the performance of DGTA-RL and the benchmark approaches in solving STSP-TDD in various scenarios. We use two key indicators to evaluate the performance of the model in each scenarios: the average tour duration for all instances (represented as "Ave.Dur.") and the total solving time for all instances (represented as "Time") measured in seconds. We accumulate the total solving time for all testing instances, as the solving time of the RL approaches for a single instance could be too tiny to capture. The bold font is the best result among the three RL approaches.

Due to the large number of testing instances, the MIP can only be solved in the small-scale scenario with $(N, T) = (10, 6)$ in the limited time. The solving time for all the instances in this scenario is approximately 45 h, which is the longest among all the approaches. Compared with the RL approaches, the solving time of GmiraTS is relatively long, that is mainly because GmiraTS must solve each new instance. In comparison, the RL approaches do not have such long time because once their model has been trained, they can quickly compute the solution for each new instance. As a result, the DGTA-RL's solving time is much shorter than GmiraTS. In the large-scale scenario with $(N, T) = (50, 24)$, DGTA-RL only takes 19 s to generate the tours, which is only 0.2% of the solving time taken by GmiraTS (10,086 s). In addition, the solving times of AMDRGM-RL exhibit an increasing trend with more nodes. Its solving time increases from 2s in the small-scale scenario with $(N, T) = (10, 6)$ to 31 s in the large-scale scenario with $(N, T) = (50, 24)$. In contrast to AMDRGM-RL, DGTA-RL mitigates the growth trend in the solving time due to the dual attention layer. Different from inputting the travel time matrix (with the scale of $T \times N \times N$) into the AMDRGM-RL, the combination of the time interval with the node location (with the scale of $T \times N \times 3$) can be inputted into the dual attention layer of DGTA model.

"Imp." calculates the improvement ratio of DGTA-RL relative to the other benchmark approaches. MIP and GmiraTS can get the better performances in all scenarios among these benchmark approaches because the input of information for them differs from the RL approaches. As a static approach, MIP, GmiraTS, and Greedy can search for the offline routing with future information on time-dependent travel time. However, as a dynamic approach, the DGTA model can only analyze the current travel time, so it cannot make decisions based on future information on the time-dependent travel time. The best solution among the RL approach with the shortest tour duration is bold in this table. Greedy takes less solving time, but its solution quality is not as good as GmiraTS. DGTA-RL achieves a good balance between solution quality and solving time. It can produce solutions close to GmiraTS, and the gap between DGTA-RL and GmiraTS's performance does not widen as N and T increase, which is around −2% to −5% in all scenarios.

Table 2
The results for the STSP-TDD in different scenarios.

Scenario		MIP		GmiraTS		Greedy		AMDRGM-RL		NN-RL		GTA-RL		DGTA-RL							
N	T	Ave.Dur.	Time	Ave.Dur.	Time	Ave.Dur.	Time	Ave.Dur.	Time	Ave.Dur.	Time	Ave.Dur.	Time	Ave.Dur.	Time	Imp1.	Imp2.	Imp3.	Imp4.	Imp5.	Imp6.
10	6	78.48	45(h)	80.08	327	88.77	1	88.10	2	93.92	5	84.64	2	**82.74**	2	−5%	−3%	7%	6%	14%	2%
	12	–	–	79.91	324	87.73	1	89.70	2	92.91	6	83.98	2	**82.44**	2	–	−3%	6%	9%	13%	2%
	24	–	–	79.71	328	86.54	1	89.48	2	92.60	4	83.98	3	**81.89**	3	–	−3%	6%	9%	13%	3%
20	6	–	–	106.67	1,882	118.00	4	126.39	5	143.80	7	115.21	4	**111.05**	4	–	−4%	6%	14%	29%	4%
	12	–	–	106.58	1,862	116.94	4	125.41	4	183.92	7	116.36	5	**109.30**	5	–	−2%	7%	15%	68%	6%
	24	–	–	105.90	1,883	115.57	5	124.27	4	141.76	7	114.81	6	**109.34**	7	–	−3%	6%	14%	30%	5%
30	6	–	–	124.41	3,920	136.18	9	154.15	11	211.23	17	138.28	7	**128.89**	7	–	−3%	6%	20%	64%	7%
	12	–	–	124.14	3,937	135.97	9	152.73	9	202.33	17	140.50	7	**129.73**	8	–	−4%	5%	18%	56%	8%
	24	–	–	123.34	3,910	135.31	10	148.54	10	201.37	15	139.46	9	**129.83**	9	–	−5%	4%	14%	55%	7%
40	6	–	–	139.56	6,440	153.18	16	194.56	36	315.82	27	167.25	9	**145.38**	17	–	−4%	5%	34%	117%	15%
	12	–	–	139.50	6,586	153.19	16	189.10	23	272.79	28	166.10	11	**144.33**	11	–	−3%	6%	31%	89%	15%
	24	–	–	138.45	6,593	153.09	16	247.68	25	255.80	28	163.26	15	**145.20**	15	–	−5%	5%	71%	76%	12%
50	6	–	–	155.26	10,060	172.84	25	272.14	38	594.40	42	205.99	13	**161.24**	16	–	−4%	7%	69%	269%	28%
	12	–	–	155.39	10,189	173.00	25	300.22	31	314.34	42	219.45	14	**163.67**	19	–	−5%	6%	83%	92%	34%
	24	–	–	153.88	10,086	173.69	24	269.43	31	264.36	45	208.14	19	**159.55**	20	–	−4%	9%	69%	66%	30%

Note: Imp. represents the relative improvement ratio on Ave.Dur. compared to DGTA-RL, which is calculated as:

Imp1. = (MIP - DGTA-RL)/DGTA-RL × 100%

Imp1. = (GmiraTS- DGTA-RL)/DGTA-RL × 100%

Imp2. = (Greedy - DGTA-RL)/DGTA-RL × 100%

Imp3. = (AMDRGM-RL - DGTA-RL)/DGTA-RL × 100%

Imp4. = (NN-RL - DGTA-RL)/DGTA-RL × 100%

Imp5. = (GTA-RL - DGTA-RL)/DGTA-RL × 100%.

Moreover, the gap between the DGTA-RL and the MIP is 5% in the small-scale scenario with $(N, T) = (10, 6)$. The improvement ratio between the RL approaches indicates that DGTA-RL consistently achieves superior performance over all scenarios compared to AMDRGM-RL, NN-RL and GTA-RL. For example, DGTA-RL exhibits a 30% improvement over GTA-RL in the large-scale scenario with $(N, T) = (50, 24)$. The performance of DGTA-RL exceeds that of AMDRGM-RL even more significantly, achieving a remarkable 69% improvement rate in the same scenario.

In summary, NN-RL demonstrates the fundamental capability of the RL approach for solving STSP-TDD. AMDRGM-RL and GTA-RL outperform NN-RL in terms of the average tour duration. DGTA-RL is far better than AMDRGM-RL and NN-RL while maintaining a small gap with GmiraTS and MIP, demonstrating the superiority of handling time-dependent travel time in STSP-TDD. DGTA-RL not only retains the advantage of a short solving time inherent in the RL approaches, but also demonstrates superior solution quality, particularly when applied to large-scale STSP-TDD.

6.4. Comparison on the DTSP-TDS

This section evaluates DGTA-RL's performance in handling time-dependent and stochastic travel times for DTSP-TDS in various scenarios with different numbers of nodes ($N = 10, 20, 30, 40, 50$) and time steps ($T = 6, 12, 24$), while the normal distribution and gamma distribution remain the same, $(\sigma, \beta) = (15, 1.0)$. We compare DGTA-RL with two heuristic-based rolling approaches (Rolling-greedy and Rolling-opt) and two RL approaches (AMDRGM-RL and NN-RL) with respect to average tour duration and solving time. The performance of GTA-RL will be compared exclusively in the ablation study (Section 6.5). The architecture of AMDRGM-RL and NN-RL is the same as in Section 6.3, but the implementation is adapted to solve DTSP-TDS. The details of the heuristic-based rolling approaches and RL approaches are as follows.

- Rolling-greedy: The rolling greedy heuristic solves the DTSP-TDS based on the travel times realized at each decision step. At each decision step, it selects the node with the shortest travel time as next node, repeating this step until all nodes are visited.

- Rolling-opt: The rolling heuristic is proposed by Gmira et al. (2021a) to solve the DVRP-TDD. Firstly, an initial tour is based on the expected travel times. Then, the rolling procedure updates the visited nodes and adjusts the tour during the operating stage. The rolling procedure optimizes the remaining nodes by 2-opt exchanges based on the realized travel times at each decision step. If the adjusted tour for the remaining nodes is better than the current tour for the remaining nodes, the adjusted tour will replace the current tour and become the tour for the next decision step.
- AMDRGM-RL: Since AMDRGM-RL is proposed for the DVRP-TDD, there is no special structure for the expected travel times in the AMDRGM model, so AMDRGM-RL makes the decision without the expected travel time information.
- NN-RL: The structure of the NN model is the same as in Section 6.3. We input the expected and realized travel times into the model, which is the same as the input of DGTA-RL.

Since no existing RL approach addresses the DTSP-TDS, NN-RL serves as the only fair comparison to the DGTA-RL for solving DTSP-TDS. Since GmiraTS becomes ineffective in the STSP-TDD, we introduce GmiraTS[E], GmiraTS[R], and MIP to establish the static benchmark for the deterministic variant of DTSP-TDS without uncertainty. The details of the static benchmarks are as follows:

Table 3
The results for the DTSP-TDS in different scenarios.

Scenario		Rolling-greedy		Rolling-opt		AMDRGM-RL		NN-RL		DGTA-RL						GmiraTS[E]	GmiraTS[R]	MIP
N	T	Ave.Dur.	Time	Ave.Dur.	Time	Ave.Dur.	Time	Ave.Dur.	Time	Ave.Dur.	Time	Imp1.	Imp2.	Imp3.	Imp4.	Ave.Dur.	Ave.Dur.	Ave.Dur.
10	6	84.72	2	**74.79**	530	89.40	2	98.64	5	76.48	2	11%	−2%	17%	29%	81.39	69.29	64.76
	12	84.23	2	**75.35**	537	91.80	2	99.72	4	77.20	2	9%	−2%	19%	29%	80.66	69.19	–
	24	84.81	2	**76.66**	523	90.62	2	96.66	2	76.89	3	10%	0%	18%	26%	80.69	69.43	–
20	6	113.26	5	103.10	2,635	128.43	6	144.66	8	**102.61**	4	10%	0%	25%	41%	108.93	94.93	–
	12	113.07	6	103.62	2,601	123.83	11	155.23	9	**101.62**	5	11%	2%	22%	53%	108.82	95.10	–
	24	114.62	6	103.60	2,568	123.44	10	145.60	7	**102.45**	6	12%	1%	20%	42%	108.38	95.44	–
30	6	128.99	11	121.09	5,376	193.83	8	243.00	16	**112.26**	7	15%	8%	73%	116%	127.38	108.65	–
	12	130.76	11	121.18	5,122	159.58	9	208.80	16	**114.05**	7	15%	6%	40%	83%	126.97	109.75	–
	24	133.71	11	121.19	5,466	174.41	9	206.92	16	**112.66**	9	19%	8%	55%	84%	126.36	110.84	–
40	6	141.96	18	136.08	8,858	200.09	25	345.57	28	**125.94**	9	13%	8%	59%	174%	143.14	120.14	–
	12	145.74	18	137.24	8,977	196.56	17	272.64	27	**124.48**	11	17%	10%	58%	119%	143.16	121.63	–
	24	152.00	19	136.68	16,984	192.06	18	277.99	26	**125.18**	15	21%	9%	53%	122%	142.41	123.50	–
50	6	153.94	28	153.40	13,794	357.83	32	466.62	43	**132.17**	13	16%	16%	171%	253%	161.42	130.72	–
	12	160.21	27	154.54	13,714	257.36	33	374.75	40	**132.45**	14	21%	17%	94%	183%	161.65	133.12	–
	24	171.75	29	153.31	24,847	242.57	33	315.14	40	**130.10**	19	32%	18%	86%	142%	159.15	136.22	–

Note: Imp. represents the relative improvement ratio on Ave.Dur. compared to DGTA-RL, which is calculated as:
Imp1. = (Rolling-greedy - DGTA-RL)/DGTA-RL × 100%
Imp2. = (Rolling-opt - DGTA-RL)/DGTA-RL × 100%
Imp3. = (AMDRGM-RL - DGTA-RL)/DGTA-RL × 100%
Imp4. = (NN-RL - DGTA-RL)/DGTA-RL × 100%.

- GmiraTS[E]: GmiraTS optimizes the tour based on the expected travel time in the road network. We evaluate the optimized tour derived from GmiraTS[E] on the realized travel times to demonstrate the performance of the GmiraTS[E] model for DTSP-TD.
- GmiraTS[R]: GmiraTS optimizes the tour based on the assumption that the stochastic travel time is known during the planning stage. However, in reality, this value is only revealed during the operating stage.
- MIP: The deterministic variant of DTSP-TDS is formulated as a Mixed-Integer Programming (MIP) problem and solved using the Gurobi solver for exact optimization.

Table 3 compares the performance of DGTA-RL with the other benchmark approaches. GmiraTS[R] and GmiraTS[E] represent the solution of DTSP-TDS assuming full knowledge of travels time or assuming expected travel times in the solution. We mainly report them to provide reasonable ranges for the tour duration and therefore exclude their solving times. Besides, "Imp." represents the improvement ratio in performance compared to DGTA-RL. The bold font is the best result among the DGTA-RL and the benchmark approaches. The main conclusion of the comparison is as follows:

(1) Due to its simplicity, Rolling-greedy is faster than Rolling-opt. However, its performance in terms of average tour duration is consistently outperformed by DGTA-RL in all scenarios. The gap between Rolling-greedy and DGTA-RL widens as the problem scale increases. In the small-scale scenario with $(N,T) = (10,6)$, DGTA-RL's average tour duration is 76.48 compared to Rolling-greedy's 84.72, demonstrating a 11% improvement. While, in the large-scale scenario with $(N,T) = (50,24)$, DGTA-RL outperforms Rolling-greedy by 32%. This trend indicates that DGTA-RL scales more effectively, maintaining high solution quality. DGTA-RL has the advantage of slightly shorter solving times, which takes 19 s compared to Rolling-greedy's 29 s in the large-scale scenario with $(N,T) = (50,24)$. Therefore, DGTA-RL proves to be a more effective approach for solving the DTSP-TDS in various scenarios, particularly as the problem complexity increases.

(2) Rolling-opt achieves superior performance to DGTA-RL in the two small-scale scenarios with $(N,T) = (10,6)$ and $(N,T) = (10,12)$, because the Rolling-opt can obtain a good initial tour in the planning stage based on expected travel times. However, the performance of DGTA-RL is still close to that of Rolling-opt with the 2% gap. Rolling-opt's solving time is cumulative, encompassing both the planning and operating stages. The RL approaches do not need to optimize the initial tour in the planning stage but make sequential decisions in the operating stage. Therefore, the Rolling-opt takes the longest solving time to solve the DTSP-TDS. "Imp2." shows that the DGTA-RL's solutions are more efficient than Rolling-opt. For example, in the large-scale scenario with $(N,T) = (50,24)$, the solution of DGTA-RL outperforms the Rolling-opt algorithm by 18% in the average tour duration and also decreases the solving time from 24,847 to 19.

(3) AMDRGM-RL shows a superior performance with a shorter tour duration than NN-RL and is more efficient with a shorter solving time than Rolling-opt. DGTA-RL is more efficient than AMDRGM-RL. DGTA-RL outperforms AMDRGM-RL over all scenarios, with a notable 86% improvement in the large-scale scenario with $(N,T) = (50,24)$. The solving times of AMDRGM-RL and DGTA-RL are short in the small-scale scenario with $(N,T) = (10,6)$, within 2 s. In comparison, DGTA-RL's solving time increases slower than AMDRGM-RL and DGTA-RL is faster than AMDRGM-RL over all scenarios. For example, it takes DGTA-RL only 19 s in the large-scale scenario with $(N,T) = (50,24)$, while it takes AMDRGM-RL 33 s.

(4) Due to the simple model, NN-RL cannot provide a high-quality solution for the DTSP-TDS. It is considerably worse than the GmiraTS[E] based solely on expected travel times. For example, the average tour duration of NN-RL is 315.14 in the large-scale

Table 4
The ablation study with different components.

Component		GTA-RL	GTA-D-RL		GTA-T-RL		DGTA-RL	
Dynamic encoder			✓				✓	
Temporal pointer					✓		✓	
N	T	Ave.Dur.	Ave.Dur.	Imp.	Ave.Dur.	Imp.	Ave.Dur.	Imp.
10	6	84.97	80.34	5.5%	81.94	3.6%	76.48	10.0%
	12	85.02	79.81	6.1%	82.03	3.5%	77.20	9.2%
	24	85.04	80.11	5.8%	81.67	4.0%	76.89	9.6%
20	6	116.87	106.49	8.9%	110.32	5.6%	102.61	12.2%
	12	115.64	106.41	8.0%	111.59	3.5%	101.62	12.1%
	24	116.05	105.78	8.8%	110.35	4.9%	102.45	11.7%
30	6	144.17	116.90	18.9%	134.35	6.8%	112.26	22.1%
	12	140.65	121.49	13.6%	134.87	4.1%	114.05	18.9%
	24	140.47	117.96	16.0%	133.61	4.9%	112.66	19.8%
40	6	163.82	129.98	20.7%	153.22	6.5%	125.94	23.1%
	12	167.67	131.57	21.5%	160.22	4.4%	124.48	25.8%
	24	167.39	132.65	20.8%	152.84	8.7%	125.18	25.2%
50	6	204.69	138.21	32.5%	200.60	2.0%	132.17	35.4%
	12	224.32	140.05	37.6%	202.40	9.8%	132.45	41.0%
	24	220.17	139.79	36.5%	185.93	15.6%	130.10	40.9%

Note: Imp. represents the relative improvement ratio on Ave.Dur. compared to GTA-RL.
For example: Imp. = (GTA-RL - DGTA-D-RL)/GTA-RL × 100% in the GTA-D-RL column.

scenario with $(N, T) = (50, 24)$, while the average tour duration of GmiraTS[E] is 159.15. In contrast, DGTA-RL performs much better than NN-RL, with improvement ratio increasing from 29% in the small-scale scenario with $(N, T) = (10, 6)$ to 142% in the large-scale scenario with $(N, T) = (50, 24)$. In addition, DGTA-RL's solving time is also shorter, demonstrating its excellent time efficiency and solving ability.

The gap between the GmiraTS[E] and GmiraTS[R] (or MIP) shows the impact of stochastic travel times on optimizing the tour duration. It is difficult to deal with stochastic travel times in the DTSP-TDS; even the state-of-the-art AMDRGM-RL cannot perform better than GmiraTS[E]. However, DGTA-RL can outperform GmiraTS[E] over all scenarios. MIP can provide an exact lower bound for the deterministic variant, but due to the complexity of the problem and the time limitations, it cannot provide the optimal solution for all scenarios. GmiraTS[R] is an alternative benchmark to MIP, capable of providing a comparable solution in all scenarios. Since GmiraTS is a non-optimal heuristic algorithm, GmiraTS[R] cannot obtain the optimal solution with the shortest tour duration over all scenarios. In the large-scale scenario with $(N, T) = (50, 24)$, DGTA-RL(130.10) outperforms GmiraTS[R](136.22), although GmiraTS know the actual travel time. The comprehensive comparison illustrates DGTA-RL's outstanding ability to navigate the complexities of stochastic travel times in DTSP-TDS. Compared with the benchmark approaches (Rolling-greedy, Rolling-opt, NN-RL, AMDRGM-RL, and GTA-RL), DGTA-RL obtains the shortest tour duration in the scenario with more than 20 nodes. DGTA-RL can quickly provide high-quality solutions over all scenarios. This highlights DGTA-RL's excellent performance and high efficient to solve DTSP-TDS.

6.5. Ablation study

This section conducts an ablation study to explain the function of each novel component in the DGTA model. The ablation study evaluates the performance of the original GTA-RL (proposed by Gunarathna et al. (2022)), GTA-D-RL (GTA-RL with dynamic encoder), GTA-T-RL (GTA-RL with temporal pointer), and DGTA-RL (GTA-RL with dynamic encoder and temporal pointer). The ablation study is conducted in various scenarios with different numbers of nodes ($N = 10, 20, 30, 40, 50$) and time steps ($T = 6, 12, 24$), while normal distribution and gamma distribution remain the same, specifically $(\sigma, \beta) = (15, 1.0)$.

As shown in Table 4, it is evident that DGTA-RL consistently achieves the shortest tour duration over all scenarios, demonstrating its superior performance. In the small-scale scenario with $(N, T) = (10, 6)$, DGTA-RL improves the average tour duration from 84.97 (GTA-RL) to 76.48, marking a 10% improvement. In the larger scale scenario, the advantage of DGTA-RL becomes even more pronounced. In particular, the average duration of DGTA-RL is 132.45 in the scenario with $(N, T) = (50, 12)$, which is lower than GTA-RL (224.32), achieving a significant improvement of 41.0%.

GTA-D-RL and GTA-T-RL individually offer some improvements over the original GTA-RL, but their performance varies with different scales of problem. In scenarios with 10 nodes, the improvement ranges from 5.5% to 6.1%. However, in scenarios with 50 nodes, the improvement increases to between 32.5% and 37.6%, indicating that the dynamic encoder effectively handles larger node counts. GTA-T-RL shows lower improvements than GTA-D-RL in all scenarios. For example, in small-scale scenarios, GTA-T-RL provides minor improvements (2.0%). In the large-scale scenario with $(N, T) = (50, 24)$, GTA-T-RL improves the average duration by 15.6% compared to GTA, which is still lower than the rate of improvement of GTA-D-RL (36. 5%). In the same scenario, DGTA-RL, which incorporates both components, achieves a 40.9% improvement, surpassing the improvements of GTA-D-RL and GTA-T-RL. The comparisons of GTA-D-RL and GTA-T-RL highlight that each component contributes to their individual improvements.

The DGTA-RL, which integrates both the dynamic encoder and the temporal pointer, leverages the improvements of both components while mitigating their individual weaknesses. The synergistic effect of combining these components results in significant

Table 5
The generalization analysis in different scenarios.

Scenario			Rolling-opt		DGTA-RL[S]			DGTA-RL[M]			GmiraTS[E]	GmiraTS[R]
σ	N	β	Ave.Dur.	Time	Ave.Dur.	Imp.	Time	Ave.Dur.	Imp.	Time	Ave.Dur.	Ave.Dur.
5	30	1.0	49.87	5 205	38.83	22%	11	**38.31**	23%	8	57.06	38.03
10	30	1.0	104.78	5 202	98.06	6%	9	**93.06**	11%	8	110.73	89.29
15	30	1.0	148.73	5 135	146.37	2%	10	**140.92**	5%	8	153.46	136.60
20	30	1.0	167.96	5 128	166.65	1%	10	**162.51**	3%	8	172.82	156.66
25	30	1.0	178.14	5 132	177.84	0%	14	**174.43**	2%	8	183.08	166.31
15	10	1.0	**72.85**	5 38	81.51	−12%	6	74.58	−2%	2	78.70	66.38
15	20	1.0	123.24	2 614	123.11	0%	5	**118.03**	4%		128.22	110.45
15	30	1.0	149.65	5 231	147.05	2%	10	**141.26**	6%	8	154.30	137.92
15	40	1.0	165.89	8 929	159.44	4%	13	**153.83**	7%	11	170.94	154.03
15	50	1.0	180.25	13 885	168.19	7%	20	**163.54**	9%	16	185.42	166.70
15	30	0.6	151.42	5 327	152.45	−1%	13	**147.01**	3%	8	154.44	142.20
15	30	0.8	150.27	5 321	149.80	0%	12	**144.28**	4%	8	154.40	139.59
15	30	1.0	149.59	5 378	147.32	2%	11	**141.50**	5%	8	154.18	137.67
15	30	1.2	148.70	5 358	144.69	3%	10	**138.32**	7%	8	154.19	135.63
15	30	1.4	147.19	5 153	141.45	4%	11	**135.33**	8%	8	153.58	133.57

Note: Imp. represents the relative improvement ratio on Ave.Dur. compared to the Rolling-opt.
For example: Imp. = (DGTA-RL[S] - Rolling-opt)/Rolling-opt × 100% in the DGTA-RL[S] column.

improvements over all scenarios. DGTA-RL outperforms both GTA-D-RL and GTA-T-RL, with improvements ranging from 9.2% to 41.0% over the original GTA-RL, highlighting its adaptability to various scenarios. For example, in the small-scale scenario with $(N, T) = (10, 6)$, DGTA-RL improves performance by 10.0%, compared to the average duration of GTA-RL (84.97). This improvement is significantly higher than the improvements by GTA-D-RL (5.5%) and GTA-T-RL (3.6%) in the same scenario. From a model perspective, the dynamic encoder and the temporal pointer play different roles in decision-making. There is a lot of information in the spatial–temporal representation, so inputting all of them into the decoder may make the model inefficient. The dynamic encoder selects and encodes the related representations to enable efficient solution generation. It prioritizes the current time and its next potential node, thus disregarding the other representations that may not significantly influence decision making. The temporal pointer emphasizes temporal representations, taking into account both future and past expected travel times to guide current decisions.

In summary, the ablation study underscores the superior performance of DGTA-RL in solving the DTSP-TDS. The GTA-D-RL and GTA-T-RL provide consistent improvements that increase as the problem scale increases, and the combination of both components in DGTA-RL leads to superior performance.

6.6. Generalization study

This section investigates the generalization capability of DGTA-RL across various scenarios and compares its performance with Rolling-opt (as shown in Table 5). We examine two types of generalization: single-scenario generalization, where a policy trained on one scenario is tested across different scenarios, and multi-scenario generalization, where a policy trained on multiple scenarios is evaluated for its ability to find a route for a diverse set of scenarios. In single-scenario generalization study, we train the DGTA model in the single scenario with $(\sigma, N, \beta) = (15, 30, 1.0)$, and its performances in other scenarios with are indicated as DGTA-RL[S]. In the multi-scenario generalization study, we train the DGTA model in the generalized scenario with $\sigma \in [5, 15]$, $N \in [10, 50]$, $\beta \in [0.6, 1.4]$, and its performance in various scenarios is indicated as DGTA-RL[M]. The time intervals remains as 12 for all scenarios in the generalization study.

Since rolling-opt has demonstrated superior performance in solving DTSP-TDS in Section 6.4, the generalization capability is compared with rolling-opt. "Imp." in Table 5 is the improvement rates of DGTA-RL[S] and DGTA-RL[M]'s average duration, compared with Rolling-opt. As the value of σ increases, their improvement rates decrease, possibly because the multi-attention component in the model can amplify features of clustered nodes, differentiating features of nearby nodes. As the node number increases, the gap between DGTA-RL[M] and Rolling-opt widens. For example, DGTA-RL[M] outperforms Rolling-opt by 9% in the scenario with $(\sigma, N, \beta) = (15, 50, 1.0)$. This difference arises due to the expanding solution space for Rolling-opt with more nodes, making it more challenging to find an optimal solution within a limited time. In contrast, DGTA-RL[M] demonstrates less sensitivity to the number of nodes and consistently provides high-quality solutions for instances with more than 20 nodes. The improvement rates for DGTA-RL[S] and DGTA-RL[M] rise as β increases, demonstrating DGTA-RL's strong generalization capability in effectively addressing uncertainty in travel times.

It can be found that, except for the small-scale scenario with 10 nodes, DGTA-RL[M] outperforms Rolling-opt across all scenarios. Due to training with a broader range of scenarios, DGTA-RL[M] demonstrates significant improvements over DGTA-RL[S] across all scenarios. DGTA-RL[S] can outperforms Rolling-opt when evaluating the trained model in the most of the scenarios. Additionally, DGTA-RL demonstrates high efficiency compared to Rolling-opt, which is important for practical applications. The sum of the training time and solving time for the DGTA-RL is 3559 s in the scenario with $(\sigma, N, \beta) = (15, 30, 1.0)$, which is even shorter than Rolling-opt's total solving time (5231 s) in the same scenario. In practical applications with limited time and scenario, the DGTA

model trained in a single scenario can provide efficient solutions for DTSP-TDS in other scenarios.

The single-scenario generalization study demonstrates that the implementation of DGTA-RL is not limited to known scenarios but exhibits a notable generalization ability to adapt to some new scenarios. This highlights the model's generalization capacity to overcome the constraints of its training data. Meanwhile, the multi-scenario generalization study highlights the model's effectiveness across diverse scenarios, showing that a single model trained on multiple scenarios can replace models trained individually for each scenario. This approach enhances efficiency by reducing the need for scenario-specific models, thus lowering training costs. These two generalized performances provide a comprehensive study of the model's performance and generalization capability across various scenarios.

7. Conclusion

This paper investigates the dynamic traveling salesman problem with time-dependent and stochastic travel times in a dynamic environment. To handle the high-dimensional solution space, we model the problem as a sequential decision-making process using a Markov Decision Process and propose a novel DGTA-RL approach. The DGTA-RL approach incorporates the DGTA model, effectively encoding the state of time-dependent and stochastic travel times. We evaluate the DGTA-RL in various scenarios. DGTA-RL can efficiently handle time-dependent and deterministic travel times, generating solutions similar to the GmiraTS but with significantly shorter solving times. Moreover, DGTA-RL consistently outperforms other benchmark approaches in the DTSP-TDS, showing its potential to deal with time-dependent and stochastic travel times effectively.

These findings highlight the importance of incorporating stochastic elements in the model to achieve better performance. Furthermore, the trained DGTA model exhibits generalization capability to various scenarios with varying customers' locations and uncertainty levels of travel times without requiring additional training time. DGTA-RL is fast and flexible compared to Rolling-opt, producing competitive online decisions. It offers efficiency and generalization capabilities, making it a valuable tool for addressing complex logistics optimization challenges. Our work contributes to advancing the field of DTSP-TDS, with potential applications in various industries.

Despite the promising opportunities, there are some limitations to consider in the future. Firstly, the DGTA-RL approach assumes known and fixed customer, which may not be realistic in some real-world cases. Future research could focus on addressing the stochastic customers. Additionally, the DGTA-RL approach currently focuses on the DTSP-TDS, but other variants of vehicle routing problems could benefit from similar reinforcement learning approaches. Exploring these variants and developing novel reinforcement learning approaches with deep neural networks could be a promising direction for future research.

CRediT authorship contribution statement

Dawei Chen: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Data curation, Conceptualization. **Christina Imdahl:** Writing – review & editing, Validation, Supervision, Project administration, Investigation, Formal analysis, Conceptualization. **David Lai:** Writing – review & editing, Supervision, Investigation, Conceptualization. **Tom Van Woensel:** Writing – review & editing, Supervision, Resources, Project administration.

Acknowledgments

This work has made use of resources and expertise provided by SURF Experimental Technologies Platform, which is part of the SURF cooperative in the Netherlands (No.EINF-9576). Dawei Chen acknowledges financial support from the China Scholarship Council (No. 202106370026).

Appendix A. Multi-head attention mechanism

The multi-head attention mechanism(with M heads) compute the input representation $H^{(l)}$ to get query representations Q_m^l, key representations K_m^l, and value representations V_m^l for each head m as:

$$Q_m^l = H^{(l)} W_{q,m}^l, K_m^l = H^{(l)} W_{k,m}^l, V_m^l = H^{(l)} W_{v,m}^l \tag{25}$$

where $W_{q,m}^l, W_{k,m}^l, W_{v,m}^l \in \mathbb{R}^{D \times D^m}$ are trainable parameters. Subsequently, the query, key and value representations are used to compute the $\sqrt{D^m}$-dimensional node representations in each head, such that:

$$a_m = \text{Softmax}\left(\frac{Q_m^l \cdot \left(K_m^l\right)^T}{\sqrt{D^m}} \right) \cdot V_m^l \tag{26}$$

after which the outputs from all heads are concatenated and transformed by a learnable matrix $W_c^l \in \mathbb{R}^{D \times D}$ to obtain the advanced $\sqrt{D^m}$-dimensional node representations below:

$$\mathbf{MHA}_l \left(H^{(l)}; \mathbf{W}^l \right) = \text{concat}(a_1, a_2, \ldots, a_M) \cdot \mathbf{W}_c^l \tag{27}$$

where $\mathbf{W}^l = \left(W_{q,m}^l, W_{k,m}^l, W_{v,m}^l \right)_{m=1,\ldots,M}$ represents the trainable parameters at each attention mechanism.

Table 6
Notations.

Notation	Description
Notations for the MDP	
$\mathcal{T}$	The set of time intervals. $\mathcal{T} = \{1, 2, \dots, T\}$, where T is the total number of intervals
t	General index for time interval, $t \in \mathcal{T}$
Δt	Length of each time interval
$\mathcal{N}$	The set of nodes. $\mathcal{N} = \{0, 1, 2, \dots, N\}$, where 0 represents the depot, and the others represent customers. N represents the node number
i, j	Nodes index, $i, j \in \mathcal{N}$
x_i, y_i	The horizontal and vertical coordinates of node i
$\hat{c}_{ij}(t)$	Expected travel time between nodes i and j in time interval t
c_{ijd}	Realized travel time between nodes i and j at decision step d
D	Number of the decision steps
d	Index of the decision step
S_d	The subsequent state at decision step d
τ_d	Time at the decision step d
t_d	The time interval of the decision step d
$A_d(j)$	The arrival time at node j at the decision step d
$\mathbf{A}_d$	The vector of arrival times at the decision step d, $\mathbf{A}_d = (A_d(j))_{j \in \mathcal{N}}$
$\mathbf{V}_d$	The vector of the visited nodes
X_d^π	The next node to visit at decision step d, determined by policy π
β	The scale of the gamma distribution
Notations for the DGTA model	
θ	The parameters in the DGTA model
$\bar{\theta}$	The parameters in the baseline model
$\mathbf{G}$	Node coordinate vector
$\mathbf{G}^*$	Expanded node coordinate matrix
$\mathbf{T}$	Time interval vector
$\mathbf{T}^*$	Expanded Time interval matrix
W^E	Trainable parameter in the embedding layer
D	Dimension of the embeddings
D^m	Dimension of the query, key, and value representations for each head
$H_t^{(l,S)}$	The input of spatial attention mechanism in lth block in the dual attention layer
$\bar{\mathbf{H}}^{(l,S)}$	The output of spatial attention mechanism in lth block in the dual attention layer
$H_t^{(l,S)}$	The input of temporal attention mechanism in lth block in the dual attention layer
$\bar{\mathbf{H}}^{(l,S)}$	The output of temporal attention mechanism in lth block in the dual attention layer
$\mathbf{W}^{(l,I)}$	Trainable parameter to integrate spatial and temporal representations at the lth block
$\mathbf{H}^{(l)}$	The output of lth block in the dual attention layer
L_1	Block number in the dual attention layer
L_2	Block number of the attention layer in dynamic encoder
$\tilde{H}_d^S$	The input of the multi-head attention mechanism in the dynamic encoder
H_d^S	The spatial representation output from the dynamic encoder
C_d^S	Spatial context representation
Q_d^S	Query vector in the spatial attention mechanism at decision step d
K_d^S	Key vector in the spatial attention mechanism at decision step d
V_d^S	Value vector in the spatial attention mechanism at decision step d
Q_d^T	Query vector in the temporal attention mechanism at decision step d
K_d^T	Key vector in the temporal attention mechanism at decision step d
V_d^T	Value vector in the temporal attention mechanism at decision step d
h_d^S	Output of the spatial attention mechanism at decision step d
h_d^T	Output of the temporal attention mechanism at decision step d
$\mathbf{W}^{(l,S)}$	Trainable parameters in the spatial attention mechanism at the lth block
$\mathbf{W}^{(l,T)}$	Trainable parameters in the temporal attention mechanism at the lth block
W^S	Trainable parameter in the dynamic encoder
W^I	Trainable parameter in the integration layer

Appendix B. Notations for MDP and DGTA model

See Table 6.

References

Arigliano, A., Ghiani, G., Grieco, A., Guerriero, E., Plana, I., 2019. Time-dependent asymmetric traveling salesman problem with time windows: Properties and an exact algorithm. Discrete Appl. Math. 261, 28–39.
Avraham, E., Raviv, T., 2020. The data-driven time-dependent traveling salesperson problem. Transp. Res. Part B: Methodol. 134, 25–40.

Basso, R., Kulcsár, B., Sanchez-Diaz, I., Qu, X., 2022. Dynamic stochastic electric vehicle routing with safe reinforcement learning. Transp. Res. Part E: Logist. Transp. Rev. 157, 102496.
Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S., 2016. Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940.
Binart, S., Dejax, P., Gendreau, M., Semet, F., 2016. A 2-stage method for a field service routing problem with stochastic travel and service times. Comput. Oper. Res. 65, 64–75.
Bono, G., Dibangoye, J.S., Simonin, O., Matignon, L., Pereyron, F., 2020. Solving multi-agent routing problems using deep attention mechanisms. IEEE Trans. Intell. Transp. Syst. 22 (12), 7804–7813.
Chang, T.-S., Wan, Y.-w., Ooi, W.T., 2009. A stochastic dynamic traveling salesman problem with hard time windows. European J. Oper. Res. 198 (3), 748–759.
Chen, H.-K., Hsueh, C.-F., Chang, M.-S., 2006. The real-time time-dependent vehicle routing problem. Transp. Res. Part E: Logist. Transp. Rev. 42 (5), 383–408.
Cordeau, J.-F., Ghiani, G., Guerriero, E., 2014. Analysis and branch-and-cut algorithm for the time-dependent travelling salesman problem. Transp. Sci. 48 (1), 46–58.
Ehmke, J.F., Steinert, A., Mattfeld, D.C., 2012. Advanced routing for city logistics service providers based on time-dependent travel times. J. Comput. Sci. 3 (4), 193–205.
Franceschetti, A., Demir, E., Honhon, D., Van Woensel, T., Laporte, G., Stobbe, M., 2017. A metaheuristic for the time-dependent pollution-routing problem. European J. Oper. Res. 259 (3), 972–991.
Gao, S., Huang, H., 2012. Real-time traveler information for optimal adaptive routing in stochastic time-dependent networks. Transp. Res. Part C: Emerg. Technol. 21 (1), 196–213.
Gmira, M., Gendreau, M., Lodi, A., Potvin, J.-Y., 2021a. Managing in real-time a vehicle routing plan with time-dependent travel times on a road network. Transp. Res. Part C: Emerg. Technol. 132, 103379.
Gmira, M., Gendreau, M., Lodi, A., Potvin, J.-Y., 2021b. Tabu search for the time-dependent vehicle routing problem with time windows on a road network. European J. Oper. Res. 288 (1), 129–140.
Gouveia, L., Paias, A., Ponte, M., 2023. The travelling salesman problem with positional consistency constraints: An Application to healthcare services. European J. Oper. Res. 308 (3), 960–989.
Gunarathna, U., Borovica-Gajic, R., Karunasekara, S., Tanin, E., 2022. Solving dynamic graph problems with multi-attention deep reinforcement learning. arXiv preprint arXiv:2201.04895.
Guo, W., Atasoy, B., Negenborn, R., 2022. Global synchromodal shipment matching problem with dynamic and stochastic travel times: A reinforcement learning approach. Ann. Oper. Res. 1–32.
Guo, F., Wei, Q., Wang, M., Guo, Z., Wallace, S.W., 2023. Deep attention models with dimension-reduction and gate mechanisms for solving practical time-dependent vehicle routing problems. Transp. Res. Part E: Logist. Transp. Rev. 173, 103095.
Haghani, A., Jung, S., 2005. A dynamic vehicle routing problem with time-dependent travel times. Comput. Oper. Res. 32 (11), 2959–2986.
Hashemi Doulabi, H., Pesant, G., Rousseau, L.-M., 2020. Vehicle routing problems with synchronized visits and stochastic travel and service times: Applications in healthcare. Transp. Sci. 54 (4), 1053–1072.
Hildebrandt, F.D., Thomas, B.W., Ulmer, M.W., 2022. Opportunities for reinforcement learning in stochastic dynamic vehicle routing. Comput. Oper. Res. 106071.
Huang, Z., Huang, W., Guo, F., 2019. Integrated sustainable planning of self-pickup and door-to-door delivery service with multi-type stations. Comput. Ind. Eng. 135, 412–425.
Ichoua, S., Gendreau, M., Potvin, J.-Y., 2003. Vehicle dispatching with time-dependent travel times. European J. Oper. Res. 144 (2), 379–396.
James, J., Yu, W., Gu, J., 2019. Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. IEEE Trans. Intell. Transp. Syst. 20 (10), 3806–3817.
Kim, S., Lewis, M.E., White, C.C., 2005. Optimal vehicle routing with real-time traffic information. IEEE Trans. Intell. Transp. Syst. 6 (2), 178–188.
Kitjacharoenchai, P., Ventresca, M., Moshref-Javadi, M., Lee, S., Tanchoco, J.M., Brunese, P.A., 2019. Multiple traveling salesman problem with drones: Mathematical model and heuristic approach. Comput. Ind. Eng. 129, 14–30.
Kool, W., Van Hoof, H., Welling, M., 2018. Attention, learn to solve routing problems!. arXiv preprint arXiv:1803.08475.
Laporte, G., Louveaux, F., Mercure, H., 1992. The vehicle routing problem with stochastic travel times. Transp. Sci. 26 (3), 161–170.
Lee, E., Cen, X., Lo, H.K., 2022. Scheduling zonal-based flexible bus service under dynamic stochastic demand and Time-dependent travel time. Transp. Res. Part E: Logist. Transp. Rev. 168, 102931.
Levering, N., Boon, M., Mandjes, M., Núñez-Queija, R., 2022. A framework for efficient dynamic routing under stochastically varying conditions. Transp. Res. Part B: Methodol. 160, 97–124.
Li, J., Ma, Y., Gao, R., Cao, Z., Lim, A., Song, W., Zhang, J., 2021a. Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem. IEEE Trans. Cybern. 52 (12), 13572–13585.
Li, J., Xin, L., Cao, Z., Lim, A., Song, W., Zhang, J., 2021b. Heterogeneous attentions for solving pickup and delivery problem via deep reinforcement learning. IEEE Trans. Intell. Transp. Syst. 23 (3), 2306–2315.
Liu, Z., Li, X., Khojandi, A., 2022. The flying sidekick traveling salesman problem with stochastic travel time: A reinforcement learning approach. Transp. Res. Part E: Logist. Transp. Rev. 164, 102816.
Mao, C., Shen, Z., 2018. A reinforcement learning framework for the adaptive routing problem in stochastic time-dependent network. Transp. Res. Part C: Emerg. Technol. 93, 179–197.
Özarık, S.S., Veelenturf, L.P., Van Woensel, T., Laporte, G., 2021. Optimizing e-commerce last-mile vehicle routing and scheduling under uncertain customer presence. Transp. Res. Part E: Logist. Transp. Rev. 148, 102263.
Pan, W., Liu, S.Q., 2023. Deep reinforcement learning for the dynamic and uncertain vehicle routing problem. Appl. Intell. 53 (1), 405–422.
Pham, D.T., Kiesmüller, G.P., 2022. Multiperiod integrated spare parts and tour planning for on-site maintenance activities with stochastic repair requests. Comput. Oper. Res. 148, 105967.
Pillac, V., Gendreau, M., Guéret, C., Medaglia, A.L., 2013. A review of dynamic vehicle routing problems. European J. Oper. Res. 225 (1), 1–11.
Pralet, C., 2023. Iterated maximum large neighborhood search for the traveling salesman problem with time windows and its time-dependent version. Comput. Oper. Res. 150, 106078.
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 J. Oper. Res. 238 (1), 18–30.
Serrano, B., Florio, A.M., Minner, S., Schiffer, M., Vidal, T., 2024. Contextual stochastic vehicle routing with time windows. arXiv preprint arXiv:2402.06968.
Sun, P., Hu, Y., Lan, J., Tian, L., Chen, M., 2019. TIDE: Time-relevant deep reinforcement learning for routing optimization. Future Gener. Comput. Syst. 99, 401–409.
Sun, P., Veelenturf, L.P., Hewitt, M., Van Woensel, T., 2020. Adaptive large neighborhood search for the time-dependent profitable pickup and delivery problem with time windows. Transp. Res. Part E: Logist. Transp. Rev. 138, 101942.
Sutskever, I., Vinyals, O., Le, Q.V., 2014. Sequence to sequence learning with neural networks. Adv. Neural Inf. Process. Syst. 27.
Tang, X., Li, M., Lin, X., He, F., 2020. Online operations of automated electric taxi fleets: An advisor-student reinforcement learning framework. Transp. Res. Part C: Emerg. Technol. 121, 102844.

Taniguchi, E., Shimamoto, H., 2004. Intelligent transportation system based dynamic vehicle routing and scheduling with variable travel times. Transp. Res. Part C: Emerg. Technol. 12 (3–4), 235–250.
Taş, D., Dellaert, N., Van Woensel, T., De Kok, T., 2013. Vehicle routing problem with stochastic travel times including soft time windows and service costs. Comput. Oper. Res. 40 (1), 214–224.
Taş, D., Dellaert, N., van Woensel, T., De Kok, T., 2014a. The time-dependent vehicle routing problem with soft time windows and stochastic travel times. Transp. Res. Part C: Emerg. Technol. 48, 66–83.
Taş, D., Gendreau, M., Dellaert, N., Van Woensel, T., De Kok, A., 2014b. Vehicle routing with soft time windows and stochastic travel times: A column generation and branch-and-price solution approach. European J. Oper. Res. 236 (3), 789–799.
Ulmer, M.W., Goodson, J.C., Mattfeld, D.C., Hennig, M., 2019. Offline–online approximate dynamic programming for dynamic vehicle routing with stochastic requests. Transp. Sci. 53 (1), 185–202.
Ulmer, M.W., Goodson, J.C., Mattfeld, D.C., Thomas, B.W., 2020. On modeling stochastic dynamic vehicle routing problems. EURO J. Transp. Logist. 9 (2), 100008.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. Adv. Neural Inf. Process. Syst. 30.
Vinyals, O., Fortunato, M., Jaitly, N., 2015. Pointer networks. Adv. Neural Inf. Process. Syst. 28.
Wang, Z., Dessouky, M., Van Woensel, T., Ioannou, P., 2023. Pickup and delivery problem with hard time windows considering stochastic and time-dependent travel times. EURO J. Transp. Logist. 12, 100099.
Williams, R.J., 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Reinf. Learn. 5–32.
Xiao, Y., Konak, A., 2016. The heterogeneous green vehicle routing and scheduling problem with time-varying traffic congestion. Transp. Res. Part E: Logist. Transp. Rev. 88, 146–166.
Yan, Y., Chow, A.H., Ho, C.P., Kuo, Y.-H., Wu, Q., Ying, C., 2022. Reinforcement learning for logistics and supply chain management: Methodologies, state of the art, and future opportunities. Transp. Res. Part E: Logist. Transp. Rev. 162, 102712.
Zhang, K., He, F., Zhang, Z., Lin, X., Li, M., 2020. Multi-vehicle routing problems with soft time windows: A multi-agent reinforcement learning approach. Transp. Res. Part C: Emerg. Technol. 121, 102861.
Zhang, K., Lin, X., Li, M., 2023. Graph attention reinforcement learning with flexible matching policies for multi-depot vehicle routing problems. Phys. A 128451.
Zhang, L., Liu, Z., Yu, L., Fang, K., Yao, B., Yu, B., 2022. Routing optimization of shared autonomous electric vehicles under uncertain travel time and uncertain service time. Transp. Res. Part E: Logist. Transp. Rev. 157, 102548.
Zhang, Z., Liu, H., Zhou, M., Wang, J., 2021. Solving dynamic traveling salesman problems with deep reinforcement learning. IEEE Trans. Neural Netw. Learn. Syst..
Zhao, J., Mao, M., Zhao, X., Zou, J., 2020. A hybrid of deep reinforcement learning and local search for the vehicle routing problems. IEEE Trans. Intell. Transp. Syst. 22 (11), 7208–7218.