

Reinforcement Learning Applied to the Dynamic Capacitated Profitable Tour Problem with Stochastic Requests

Marvin Caspar(✉) and Oliver Wendt

Business Information Systems and Operations Research (BISOR),
University of Kaiserslautern-Landau, Postfach 3049, 67653 Kaiserslautern, Germany
{marvin.caspar,wendt}@wiwi.uni-kl.de
https://wiwi.rptu.de/fgs/bisor

Abstract. This paper investigates the *Dynamic Capacitated Profitable Tour Problem with Stochastic Requests* (DCPTPSR), a variant of the *Traveling Salesman Problem* (TSP) with profits. In the DCPTPSR, online decisions must be made for accepting and scheduling requests over a finite number of periods. Requests follow a discrete-time stochastic process, and each request is characterized by a location, demand, and prize. Accepted requests must be served on a TSP tour such that the collected prize minus the transportation costs becomes maximal. The DCPTPSR has practical applications in food delivery and less-than-truckload transportation, where requests arrive in an online fashion and immediate decisions about acceptance and scheduling must be made. We model the DCPTPSR by a *Markov Decision Process* (MDP) and propose a *Stochastic Dynamic Programming* (SDP) algorithm for solving the problem to optimality. Addressing the computational challenges involved in SDP, we present a framework that integrates *Reinforcement Learning* (RL) as an alternative solution method. We perform an extensive numerical study where instances with up to incoming 25 requests can be solved by SDP while our RL approach can be used to adequately solve instances with even up to 100 incoming requests. Particularly, the performance of the RL approach is very close to the optimal policy by SDP and outperforms both the first come first serve heuristic and the first accept traveling salesman algorithm. The latter algorithm accepts requests if the available capacity enables it and fulfills these demands in an optimal TSP tour afterward. Especially instances with scarce capacity show considerable potential for savings in request acceptance and transportation scheduling decisions if both decisions are made simultaneously.

Keywords: Online Optimization · Profitable Tour Problem · Temporal Difference Learning · Stochastic Dynamic Programming

1 Introduction

Selecting from a sequential stream of requests those that maximize the expected total profit contribution is the main focus of revenue management (see [21]). The

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024
O. Gervasi et al. (Eds.): ICCSA 2024, LNCS 14813, pp. 346–363, 2024.
https://doi.org/10.1007/978-3-031-64605-8_24

TSP with profits extends the well-known TSP by introducing a decision-making component for selecting requests and scheduling a tour to maximize profit [11]. However, in practical applications, e.g., less-than-truckload transportation (see [26]) or the scheduling of airline operations (see [35]), or in food delivery (see [39]) requests may not be known a priori and arrive in an online fashion. In such dynamic problems with stochastic incoming requests, the decision maker must immediately decide which requests to accept while anticipating the optimization problem (e.g., a TSP) that results from accepted requests. RL with the integration of neural networks enables the processing of huge amounts of information and can provide a generic and flexible framework for such request acceptance decision-making under uncertainty (see [7]).

In this paper, we address the research topic of online optimization in the capacitated profitable tour problem and make the following contributions. We provide a formulation of the DCPTPSR by an MDP and determine different request acceptance policies to maximize the expected profit over a time horizon. Since SDP cannot determine an optimal policy for all possible DCPTPSR instances in a reasonable time due to finite computation resources, inspired by related works and their success (as exemplified by [7,10]), we propose an approach that leverages reinforcement learning to outperform different straightforward greedy approaches. For managerial insights, we perform a sensitivity analysis on the DCPTPSR concerning the number of incoming requests and available capacity to determine the benefit of simultaneous compared to subsequent request acceptance and tour scheduling decisions.

The remainder of this paper is organized as follows. In Sect. 2, we provide a brief overview of results in the areas of revenue management, tour scheduling, and reinforcement learning. Furthermore, we state relevant literature that addresses request acceptance decisions under uncertainty in the background of tour problems. Section 3 shows a precise problem description and provides a Markov decision process formulation for the DCPTPSR and discusses the huge complexity of such a stochastic dynamic decision problem. In Sect. 4, we see how SDP and our RL approach can be used to maximize the expected profit in the DCPTPSR over a time horizon. Section 5 presents detailed results of our computational experiments where both solution methods are benchmarked with a first come first serve heuristic and a first accept traveling salesman algorithm. Finally, a conclusion and open questions are discussed in Sect. 6.

2 State of the Art: Related Research Fields

2.1 Revenue Management and Routing Problems with Profits

Revenue management is the process of allocating capacity to the right kind of customer at the right price to maximize sales or revenue [18]. Products that successively lose their ability to provide utility over time or decay entirely, called perishable products (see [32]) should be considered separately to maximize profit.

Especially airlines and hotels need to decide about request acceptance or rejection and how to allocate this potential demand in case of request acceptance to fixed capacity to maximize profit (see [21]).

Request acceptance decisions about continuous and independent arrival processes can be modeled as a *Markov Decision Process* (MDP) and solved to optimality by *Stochastic Dynamic Programming* (SDP) via a complete or implicit enumeration of the solution space [1]. Such dynamic decision problems under uncertainty can be represented by a tree structure where the edges represent the decisions, called actions $(a_1, a_2, ..., a_{|T|} \in A)$, and the nodes denote the system states $(s_1, s_2, ..., s_{|T|} \in S)$ called state space over a finite time horizon T with $|T| \in \mathbb{N}_{>0}$ discrete periods. For optimizing decisions, we define the reward $(R_1, R_2, ..., R_{|T|} \in R)$ that values a decision taken at a specific state over T. Furthermore, let us assume that there is no uncertainty about the number of requests that will be received over a given time horizon T and each request gets a backward-counting index $t \in T$ stating that after the request in period t, another $t-1$ requests over $t-1$ periods will arrive.

In the following example, let us consider six discrete capacity units, which will provide a service at a future point in time. Capacity units can be sold individually or bundled, i.e., the remaining capacity always takes one of the seven values of the state space $S := \{0, 1, 2, 3, 4, 5, 6\}$. Exactly one of three different types of requests $i \in 1, 2, 3$ can arrive in a period $t \in T$, which are specified by an (arriving probability, requested quantity d_i, revenue r_i) with $F_1 = (0.5, 1, 1), F_2 = (0.3, 2, 4)$, and $F_3 = (0.2, 3, 9)$. Now the task is to either accept or reject the stochastic incoming request in a period $t \in T$ to maximize the reward over T.

To find an optimal decision, not only the type of requests and the remaining available capacity are relevant at each period t. In case of rejecting the request at period t, the value $V_{t-1}(cap)$ of the remaining capacity cap available for the $t-1$ remaining requests, and the value $V_{t-1}(cap - d_i)$ of cap reduced by the capacity requirement d_i of the current request type i in period t in case of acceptance becomes essential. A request in period t should be accepted if:

$$R_t + V_{t-1}(cap - d_i) \geq V_{t-1}(cap)$$

and is otherwise rejected. Accordingly, in each period t, given a remaining capacity cap such a decision must be made. The optimal decision for the period $t = 1$ and thus $V_1(cap)$ can be calculated since no further requests are received and requests smaller than or equal to the remaining capacity cap are accepted. Based on this, it is also possible to determine an optimal decision for period $t = 2$. Now in period $t = 2$, the question of saving capacity at period $t = 1$ arises for the first time. Let $Pr(cap, a, cap')$ represent the transition probability from remaining capacity cap to a remaining capacity cap' under an action a (accept or reject). The expected reward at period $t = 2$ depends on the expected generated revenue $r_2(cap, a)$ at this period (either 0 or the revenue of the currently accepted request) and the value of the residual capacity $V_1(cap')$ resulting from period $t = 1$, with cap' indicating the new residual capacity for the next period:

$$V_2(cap) := \max_a \{r_2(cap, a) + \sum_{cap'} Pr(cap, a, cap')V_1(cap')]\}$$

Consequently, for each possible state $cap \in S$, we need to find an optimal action a to maximize the reward, respectively the overall revenue.

Assuming that such requests come from different locations, it becomes necessary to simultaneously plan a tour to satisfy the demand of these accepted requests. The *Traveling Salesman Problem* (TSP) is the NP-hard problem of finding the shortest or cost-optimal Hamiltonian cycle in a graph, that starts at the depot, travels through all accepted requests, and ends at the depot.

In the class *Traveling Salesman Problem with Profits* (TSPPs) (refer to [11]) as in the *Vehicle Routing Problem with Profits* (VRPPs) (refer to [30]), we need to decide which customers to serve and how to sequence these accepted visits in an optimal tour under various possible objectives. The *Orienteering Problem* (OP) is about maximizing profit under an upper bound of given tour costs. Acceptance and rejection decisions also occur in the *Prize-Collecting TSP* (PCTSP), where we collect a prize for visiting a location and pay a penalty for unvisited locations under the objective of maximizing profit minus travel costs under a minimum total collected profit. Maximizing a weighted objective function of distance by subtracting traveling costs from the total collected profit is pursued in the *Profitable Tour Problem* (PTP) which constitutes the basis for our DCPTPSR. The authors in [30] stated, that most existing literature treats the single-vehicle case for the PTP, whereby a current overview of PTP works can be found in [9]. A recent and comprehensive survey of literature that integrates revenue and demand management with tour scheduling decisions is available in [12].

2.2 Reinforcement Learning

Problems in revenue management and tour scheduling problems under uncertainty with large instances usually cannot be solved efficiently by classical optimization as stochastic dynamic programming due to their complexity and resulting computational effort [28]. Reinforcement learning endeavors to maximize the cumulative generated reward $R^a_{ss'} = \mathbb{E}\{R_{t+1}|a_t = a, s_t = s, s_{t+1} = s'\}$ based on each period t as a result of selecting the action a_t which maximizes the expected reward at time t, given state s_t. With probability $Pr^a_{ss'} = Pr\{s_{t+1} = s'|s_t = s, a_t = a\}$ under the assumption of an MDP, the system transitions from state s to state s'. The transition probabilities only depend on the current state and action, not on process history. To determine optimal actions, a complete enumeration of the state space or a heuristic that allows the exclusion of suboptimal regions becomes necessary. Let $\gamma \in [0, 1]$ be the discount rate and an admissible way to find optimal results in this context is the Bellman equation (see [5]):

$$V^\pi(s) = \mathbb{E}_\pi\{R_t|s_t = s\} = \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k \ R_{t+k+1}|s_t = s \right\} \tag{1}$$

A decision policy π defines a probability distribution over all possible actions in a state s. The state value $V^{\pi}(s)$ is defined as the expected sum value of the possible discounted state value $V^{\pi}(s')$ of the next state s' and the immediate expected reward in period $t+1$ following policy π. Unlike the state value $V^{\pi}(s)$, the action value $Q^{\pi}(s,a)$ of a policy π is directly related to the action a chosen in s:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^{k}\ R_{t+k+1}|s_t = s, a_t = a\right\}$$

The unchosen paths according to the policy π are not included in calculating $Q^{\pi}(s,a)$. However, at each following decision level, the Eq. (1) is used to determine the state value for the next state s'. To find a policy that maximizes $V^{\pi}(s)$, we define optimal state and action values:

$$V^{\pi^{\star}}(s) = \max_{a}\{r(s,a) + \gamma\ V^{\pi^{\star}}(s')\}$$

$$Q^{\pi^{\star}}(s,a) = \mathbb{E}\{R_{t+1} + \gamma\ V^{\pi^{\star}}(s_t+1)|s_t = s, a_t = a\}$$

A specific decision policy π, that is followed until the end of the episode, usually does not lead to an optimal solution. Varying this policy π to π' in state s can result in an improvement of $V^{\pi}(s)$, which also holds for applying the new policy π' in all further decision situations. The so-called policy improvement theorem (see [28]) proves that an iterative improvement of policy π to π' is possible if the value function $V^{\pi}(s)$ of policy π in node s is known and, a selection of the optimal policy for the next node in each visited state s is possible:

$$\pi'(s) = \max_{a} Q^{\pi}(s,a) = \max_{a}\sum_{s'} Pr^{a}_{ss'}(R^{a}_{ss'} + \gamma\ V^{\pi}(s'))$$

$$V^{\pi'}(s) = \sum_{s'} Pr^{a}_{ss'}(R^{a}_{ss'} + \gamma\ V^{\pi}(s'))$$

Once a policy π is improved to π' using $V^{\pi}(s)$, a new value function $V^{\pi'}(s)$ can be determined, which may be used later to improve π' to π'' and so on. This constant alternation of evaluation and improvement can approximate the optimal policy $\pi^{\star}$ (see [28]).

To learn an appropriate policy for revenue management and tour scheduling problems, there are numerous different reinforcement learning algorithms with a recent review in the field of transportation in [10].

In this work, we focus on *Temporal Difference* (TD) learning (see [28]) that learns value functions $V^{\pi}(s)$ of states gradually from current estimations and adjusts predictions not only when the reward is received, but after each action. TD learning has already been applied to revenue management for the dynamic pricing of information services (see [27]) and the VRP with time windows and stochastic customers (see [17]) or the multiple depot VRP (see [4]). Recently the authors in [20] have employed TD learning to address the online vehicle dispatching problem to serve stochastic on-demand customers efficiently.

2.3 Related Literature: Uncertainty in Tour Problems with Request Acceptance Decisions

This paper addresses the dynamic capacitated profitable tour problem with stochastic requests wherein the challenge is to simultaneously decide to accept or reject stochastic arriving requests with unknown location, demand, and revenue and how to schedule them in an optimal tour in case of acceptance over a finite number of periods. Deciding about request acceptance under uncertainty in the background of tour problems has already been addressed several times in literature.

Within the domain of TSPPs and VRPPs, stochastic variants are relatively limited, primarily centered around the OP. The authors in [29] are the first ones that integrate stochastic service times in the OP. Other contributions consider stochastic travel times in the OP (e.g., [8,31]). The authors in [15] formulate an OP with stochastic profits, where the objective function becomes to maximize the probability that the collected profit over all accepted customers exceeds a predefined target. The work [36] addresses a PCTSP with stochastic customers, where customer presence is uncertain but their locations are a priori known. The paper [37] treats a probabilistic PTP, where each customer requests a service with a given probability. Recently [2] introduced a probabilistic PTP where customers show up with a known probability after the tour is planned, while the objective becomes to maximize the expected net profit.

The online TSP with rejection options was treated by [16], where rejecting requests leads to a penalty. The authors regard a basic online version where requests can be accepted at any time after arrival and a real-time version where a decision about incoming requests must be made immediately. In [38] the authors extend this TSP with rejection options by time windows and build a framework to solve the problem by combining reinforcement learning with a greedy heuristic to outperform a tabu search heuristic. Recently [19] modeled the integrated demand-management and online tour scheduling problem as an MDP. Here the provider decides on a subset to offer in response to an incoming customer request and also selects a price for each delivery option with a delivery deadline by a fleet of vehicles.

The TSP can be reformulated as a single machine scheduling problem when interpreting distances as sequence-dependent setup times (see [13]). In *Order Acceptance and Scheduling* (OAS) problems, there is a trade-off between gaining revenue and diverting capacity from other orders by simultaneously scheduling them on machines. Uncertainty in OAS literature is usually not modeled as a sequential stochastic decision process but rather as a combinatorial or *Mixed-Integer Linear Programming* (MILP) optimization problem over a given set of orders with uncertainty relating to their attributes (see [23]). The authors in [23] consider uncertainty in processing and service durations in a stochastic OAS where a set of resources must process a set of orders. Each order with a due date is processed on one resource with a fixed cost, while tardy delivery of orders causes penalties. On the one hand, the authors formulate a deterministic version as a MILP model and also give a formulation with constraint programming. On

the other hand, they regard a two-stage stochastic model, a robust model, a polyhedral-interval robust model, and a box robust model. As a notable exception [34] does present a model aggregating uncertainty about future requests that follow a Poisson distribution and present an MDP formulation. However, several differences still prohibit direct comparison with our model. Most notably their use of fixed product classes with fixed revenue and orders that have due dates and setup times do not reduce the profit by incurring any cost.

Concerning this state of research, we formulate to the best of our knowledge for the first time (inspired by the work in [33]) the *Dynamic Capacitated Profitable Tour Problem with Stochastic Requests* (DCPTPSR), where stochastic incoming requests must be accepted or rejected and an optimal tour through the locations of these accepted requests must be scheduled.

3 The Dynamic Capacitated Profitable Tour Problem with Stochastic Requests (DCPTPSR) Formulation

3.1 Problem Description

There is a depot location (l_0) hosting a vehicle with a maximum capacity of $C \in \mathbb{N}$ units over a finite time horizon $T = \{0, ..., T_d\} \subset \mathbb{N}$. Let $T_d \in \mathbb{N}_{>0}$ be the number of decision periods. In every decision period $t \in T \setminus \{0\}$ exactly one random request $[p, d, l]_i \in N$ with $i \in \{1, ..., |N|\}$ out of a request pool N arrives that demands a discrete positive quantity $d_i \in \{1, ..., C\}$ of the capacity C and yield a positive prize p_i (called revenue in the area revenue management) in case of acceptance. The same requests can occur several times in different periods and represent a discrete-time stochastic process. For each of these T_d stochastic incoming requests, we must dynamically decide whether to accept or reject the specific request and must plan a tour for the vehicle that visits all accepted request locations starting and ending at the depot. After rejecting or accepting such a request, there is no further possibility to alter the decision. Thus, the state space of the previous revenue management problem from Sect. 2.1 extends by a $(|N| + 1)$-dimensional bit vector of the depot location l_0 and the request locations $l_i (i \in \{1, ..., |N|\})$. A cost-optimal tour through all these accepted requested locations (1 bits in vector *loc*) under consideration of the residual capacity must be determined to achieve a maximum contribution margin (refer to the traveling salesman problem in [13]). This contribution margin is the sum of all accepted requests' prizes minus the costs incurred by the tour that starts and ends at the depot l_0 depending on the distance between the locations.

Let us extend and visualize the example from Sect. 2.1 to the DCPTPSR by adding a depot (l_0) and eight potential locations for customer requests ($|N| = 8$), which are arranged as shown in Fig. 1.

Accepting request type F_2 from location l_5 would result in a negative contribution margin since the new optimal tour would become $\{l_0, l_2, l_5, l_0\}$ and increase transportation costs by $\sqrt{18} \approx 4.24$, which would not be compensated by the additional prize of $p_5 = 4$ from accepting this request from location l_5.

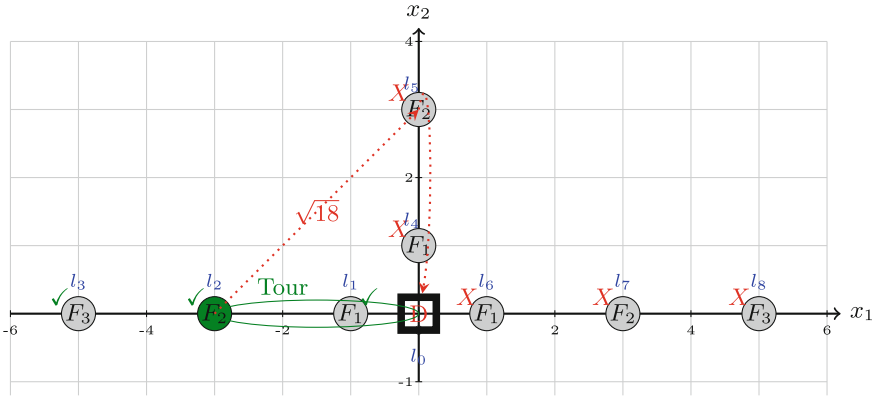


Fig. 1. State after acceptance of request type F_2 at l_2 with a remaining capacity of three capacity units. The current optimal tour is $\{l_0, l_2, l_0\}$.

Similarly, requests from locations in other directions than the already accepted location l_2 would also result in negative contribution margins (see X in Fig. 1). Here it is obvious that if the number of future requests yet to arrive is sufficiently high and the residual capacity is larger than three, the optimal strategy waits for a request type F_3 from the same cardinal direction. Only requests from the same direction can reduce the additional transportation costs to zero and maximize the total contribution margin.

3.2 Problem Formulation

In the following, we formulate the DCPTPSR as a MDP.
States: We denote *cap* as the residual capacity and *loc* as a binary vector with length $|N|+1$ stating if a location must be visited or not over any period $t \in T$. The state space becomes the combination of all possible residual capacities *cap* and location vectors *loc* by $s_t = (cap, loc)$, with

- $t \in \{T_d, ..., 1\}$ is the down-counting index of periods that indicates the number of remaining requests, respectively, decision periods.
- *cap* states the current available residual capacity.
- *loc* is a binary vector with length $|N|+1$. An entry of 1 at position i in *loc* shows that the location of request i must be visited in the tour. The first entry of *loc* is always 1 ($loc[0] = 1$) since it states the depot location where the tour must start and end

There is the initial state $s_{T_d} = (C, [1, 0, ...0])$ with start capacity C and where the first request will arrive. A terminal state where no more request arrives is given by s_0.
Actions: At each decision period $t \in \{1, ..., T_d\}$, a random request $[p, d, l]_i \in N$ arrives in the current state s_t and determines the actual action space A_s. This set A_s contains the request acceptance action $a_p \in \{0, 1\}$:

- $A0$: Requests can always be rejected. In case of rejection ($a_p = 0$), the residual capacity and accepted request locations for the next state s_{t-1} remain unchanged, without receiving any prize ($s_{t-1} = s_t$).
- $A1$: The incoming request $[p, d, l]_i \in N$ in decision period t can only be accepted ($a_p = 1$), if the residual capacity in the next state s_{t-1} holds $cap = cap - d_i \geq 0$ for this period $t - 1$.

Transitions: Between two states s_t and s_{t-1} with $t \in T \backslash \{0\}$ a transition occurs. If a request $[p, d, l]_i \in N$ arrives in period t, a decision a_p becomes necessary. The state transition principle for the DCPTPSR becomes:

- $1 \leq t \leq T_d$: If the current random request $[p, d, l]_i \in N$ with $i \in \{1, ..., |N|\}$ is accepted ($a_p = 1$) in s_t, the next state becomes $s_{t-1} = (cap - d_i, loc)$ with $loc[i] = 1$ since this accepted request location l_i must be part of the tour.
- $1 \leq t \leq T_d$: If the current random request $[p, d, l]_i \in N$ is rejected ($a_p = 0$) in state s_t, the next state stays the same with $s_{t-1} = s_t$.
- $t = 0$: No further requests will arrive and s_0 is the terminal state.

Rewards: The action-dependent rewards accrue to request acceptance decisions in every decision period $t \in T \setminus \{0\}$:

- $R_t^{A0}(s_t, a_p) = 0 \quad \forall\, a_p = 0,\, t \in T \setminus \{0\}$: the current request $[p, d, l]_i \in N$ in period t is rejected and no immediate reward follows with the same residual capacity and accepted request locations as in the period before.
- $R_t^{A1}(s_t, a_p) = p_i - \Delta_{TC}(loc, loc|\{loc[i] = 1\}) \quad \forall\, a_p = 1,\, t \in T \setminus \{0\}$: the current request $[p, d, l]_i \in N$ in period t is accepted and an immediate reward by the actual request prize p_i minus the additional transportation cost $\Delta_{TC}(loc, loc|\{loc[i] = 1\})$ due to this additional request location follows.

Policy: A policy $\pi(s_t) : S \longrightarrow A_s$ is a function that maps each state to an action. In the DCPTPSR the objective is to find the best action in each state to maximize the cumulative reward over the time horizon T.
Objective Function: Thus the objective function of the DCPTPSR becomes:

$$\max_{\pi(s_t)} \mathbb{E} \left\{ \sum_{t \in T \setminus \{0\}} R_t(s_t, \pi(s_t)) \right\}$$

and an appropriate policy must be found.
Value Function: To evaluate individual states a value function $V_t(cap, loc)$ represents the objective function value that can be expected at each decision period t. Let $Pr([p, d, l]_i)$ represent the probability that request $[p, d, l]_i \in N$ arrives in period t, the calculation of $V_t(cap, loc)$ can be expressed by:

$$V_t(cap,\, loc) = \begin{cases} -\infty & \forall cap \notin (\mathbb{R}_0^+), t \in T, \\ 0 & \forall cap \in (\mathbb{R}_0^+), t = 0, \\ \sum\limits_{[p,d,l]_i \in N} Pr([p, d, l]_i) \max\limits_{a_p \in \{0,1\}} \big\{ a_p \cdot p_i - \Delta_{TC}(loc, loc|\{loc[i] = 1\}) \\ \qquad + V_{t-1}(cap - a_p \cdot d_i,\, loc|\{loc[i] = 1\}) \big\} & \text{else} \end{cases}$$

with $\Delta_{TC}(loc, loc|\{loc[i] = 1\})$ determining the non-negative value difference between an optimal tour through loc and loc with $loc[i] = 1$.

Thus an exact computation of $V_t(cap, loc)$ requires the best acceptance decision in each decision period under the assumption that the optimal tour is determined through all accepted request locations. In each decision period t, the number of possible states is a function of the number $|N|$ of requests in the request pool and all possible tours through the accepted requests. Since one of the $|N|$ different requests can arrive and must be accepted or rejected in every decision period and these accepted request locations can be arranged in up to $T_d!$ possibilities within a tour, there are up to $(|N| \cdot 2 \cdot T_d!)^{T_d}$ possible states until a terminal state after the last decision period is reached. The astronomical complexity of such a dynamic stochastic decision problem becomes apparent since already a request pool of $|N| = 10$ and 10 decision periods ($T_d = 10$) leads up to $(10 \cdot 2 \cdot 10!)^{10} \approx 4.05 \cdot 10^{78}$ possible states until a terminal state. To solve such a complex problem as the DCPTPSR, we will show an optimal and two approximate solution methods in the following section.

4 Solution Methods for the DCPTPSR

In order to solve the DCPTPSR, we propose a stochastic dynamic programming algorithm and a temporal difference learning algorithm.

4.1 Stochastic Dynamic Programming

As already discussed in Sect. 2.1, dynamic and stochastic decision problems usually can be modeled by an MDP and solved to optimality by SDP. One concrete technique to solve such a value equation $V_t(cap, loc)$ of the DCPTPSR is an iterative algorithm known as SDP value iteration. Value iteration solves the value function $V_t(\cdot)$ by setting up a recurrence relation for which the Bellman equation (1) is a fixed point, while this iteration converges to the optimal value function $V_t^\star(\cdot)$ for arbitrary start values $V_0(\cdot)$ (see [6]). The value iteration iterates over all periods t, and all possible requests in N that can arrive in each period t. For all possible states and all possible arriving requests in a decision period t, we must check if the corresponding request can be accepted. If the acceptance is possible, the value $V_t(cap, loc)$ must be updated according to the higher value increase between accepting and rejecting the current request i. Let $\Delta_{TC}(loc', loc)$ be a function that returns the additional cost of the optimal TSP tour with location vector loc' (including the location of this new request location) compared to the optimal TSP tour with location vector loc. Acceptance of the request i leads to a value increase of $Pr(loc, a_p, loc')(p_i + V_{t-1}(cap', loc') - \Delta_{TC}(loc', loc))$ with new residual capacity cap' and new location vector loc'. In the case of rejection, $V_t(cap, loc)$ is increased by $Pr(loc, a_p, loc)V_{t-1}(cap, loc)$, since cap and loc remain unchanged. Algorithm 1 provides a generic description of how to solve the value equation $V_t(cap, loc)$ by such an SDP value iteration.

Algorithm 1: A Stochastic Dynamic Programming Algorithm as value iteration for the DCPTPSR

Function `iterative_sdp`$(T, S_{|T|+1}, N)$**:**
  Initialize $V_t(cap, loc)$ arbitrarily for every period $t \in T$ and state (cap, loc)
  for *each period* $t \in T$ **do**
    for *each possible request* $[p, d, l]_i \in N$ **do**
      for *each possible state* (cap, loc) **do**
        if $cap - q \geq 0$ **then**
          Update loc' according to l_i
          $V_t(cap, loc) += max\{V_{t-1}(cap, loc), p_i + V_{t-1}(cap - d_i, loc') - \Delta_{TC}(loc', loc)\}Pr(loc, a_p, loc')$
        else
          $V_t(cap, loc) += V_{t-1}(cap, loc)Pr(loc, a_p, loc)$
  return $V_t^\star(cap, loc)$

Such problems as the DCPTPSR usually cannot be solved efficiently by SDP (see Algorithm 1) due to their complexity and resulting computational effort. For this reason, approximate solution methods for the DCPTPSR become inevitable.

4.2 Temporal Difference Learning for the DCPTPSR

As part of reinforcement learning, TD learning can learn directly without a model of the environment by updating an existing estimate in every time step without waiting for a final outcome (see [28]). Unlike the state value $V^\pi(s)$, the state action value $Q^\pi(s, a)$ of a policy π is directly related to the action a chosen in state s and a simple one step TD learning (called TD(0)) makes the update using the estimate $Q(s_{t+1}, a_{t+1})$, receiving reward R_{t+1} and learning rate α:

$$Q^\pi(s_t, a_t) = Q(s_t, a_t) + \alpha \left(R_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right) \tag{2}$$

Algorithm 2: Temporal Difference Algorithm TD(0) (see [28])

Initialize $Q(s, a)$ arbitrarily
for *each episode* **do**
  Initialize s
  Choose a from s using policy derived from Q (e.g., ϵ-greedy)
  for *each step (period) of this episode* **do**
    Take action a, observe reward R, and next state s'
    Choose $a' \in A(s')$ using a policy derived from Q (e.g., ϵ-greedy)
    $Q(s, a) = Q(s, a) + \alpha(\mathrm{R} + \gamma Q(s', a') - Q(s, a))$
    $s = s', a = a'$
  until s is the terminal state
return Q

With probability $\epsilon \in (0, 1]$, we explore by selecting a random action from the action space, and with probability $1 - \epsilon$ we select a greedy action every time a request arrives. The corresponding policy iteration algorithm for the *Temporal Difference* TD(0) procedure estimating Q is stated in Algorithm 2:

For our DCPTPSR model, we set the discount rate to $\gamma = 1$ and after each incoming request, we must take an action, while in case of acceptance, we need to calculate the cost of integrating this request at an optimal position in the TSP tour. After observing the reward and the new state, we update the state action function according to the TD(0) rule (2) and iterate forward until the terminal state is reached.

For higher numbers of requests and capacity, several obstacles e.g., memory limits or residual value function information compression prohibit the explicit representation of the residual value function. In this context, reinforcement learning in connection with neural networks is worth highlighting and can lead to human-level performance and even beyond [22,28]. Thus, we investigate how the TD approach (Algorithm 2) performs in the DCPTPSR when we use an *Artificial Neural Network* (ANN), respectively a multi-layer perception (see [24]) with three layers. Hereby, input neurons represent the current available residual capacity *cap* and the location vector *loc* and there is a time neuron specifying the remaining number of requests. A single output neuron represents the remaining value determined by the values of the input neurons and the corresponding neurons in the hidden layer. Each time a request arrives, the ANN determines the residual value for two different (cap, loc) combinations, one for acceptance and one for rejection. The difference between these outputs determines the acceptance or rejection decision for this request, considering its immediate reward.

5 Computational Experiments and Results

This section compares the performance of the TD algorithm with SDP and two algorithms in several different experimental environments.

5.1 Experimental Setup

All experiments were performed on an Intel(R) Core(TM) i7-10750H CPU with 2.60GHz and access to 4 cores and 16 GB of RAM. The TD and SDP algorithm implementation was done in Java 16. We implemented the other two algorithms for benchmarking in Python 3.9 and used the branch-and-cut solver Gurobi Optimizer 10.0 [14] with a time limit of 6 h using 4 threads. Furthermore, the highly optimized Google OR-tools v9.7 [25] for finding optimal TSP tours is used.

Since the DCPTPSR is to the best of our knowledge addressed for the first time and there are no suitable benchmark test instances, we extend and test instances from the Capacitated Vehicle Routing Problem Library (CVRPLIB) available at http://vrp.galgos.inf.puc-rio.br/index.php/en/. The chosen instances are taken from different sets and include up to 100 requests varying in

request location, demand, and prize while arriving requests are independent and uniformly distributed. The procedure for generating the prize p_i of request i is similar to [3] with $p_i = h \cdot d_i$, where h is an integer random number uniformly generated in the interval $[1, 10]$.

We use four different algorithms to solve these DCPTPSR test instances:

- A first come first serve heuristic accepts over the T_d decision periods every stochastic incoming request if the available capacity enables it and schedules a tour according to the sequence of these accepted requests.
- The first accept traveling salesman algorithm accepts every stochastic incoming request if the capacity enables it and determines an optimal TSP tour through all these accepted request locations.
- Stochastic dynamic programming can solve some instances to optimality in an appropriate computation time (see Algorithm 1).
- A temporal difference learning algorithm with a 3-layer ANN is trained to mimic the value function (see Algorithm 2).

Preliminary experiments with the TD learning algorithm have shown that 128 neurons in the hidden layer of the neural network and a learning rate α of 0.05 with an $\epsilon = 0.1$ greedy strategy lead to promising results for our test instances. Each instance undergoes 5,000,000 training episodes and subsequent validation in 1,000,000 episodes.

5.2 Results of the Experiments

Within the first experimental part, we consider different test scenarios with up to 25 stochastic incoming requests, where it is possible to benchmark the reinforcement learning approach with the optimal solution by SDP given our computational resources. The first column states the instance with the pool size of requests and the capacity of the vehicle. Column two states the first come first serve heuristic solution and the third column gives the first accept traveling salesman algorithm solution. In column four we see the SDP solution Z^*, while the fifth column displays the needed computation time with Algorithm 1 in seconds. The sixth and seventh columns show the solution Z_{TD} of the TD Algorithm (see Algorithm 2) and the needed computation time. In the last column (Perf.) we see the ratio of the reinforcement learning solution Z_{TD} and the SDP solution Z^*.

Table 1 shows that our TD learning approach (see Algorithm 2) strongly outperforms the FCFS heuristic and the FATS algorithm and reaches on average around 98 % of the optimal solution obtained by SDP in only around 4 % of this SDP computation time. The FATS algorithm shows an average improvement of around 26 % compared to the FCFS heuristic in these instances, which underlines the importance of a subsequent tour optimization after accepting random incoming requests. Especially instances with smaller capacity ($C = 50$) show that a significant increase in profit is possible if acceptance and transportation scheduling decisions are made simultaneously (Z^*) compared to sequentially

Table 1. Numerical results for the DCPTPSR with varying T_d.

Id	Instance ($\|N\|, C$)	FCFS	FATS	SDP		TD		
		Z_{FCFS}	Z_{FATS}	Z^*	t	Z_{TD}	t	Perf.(%)
$T_d = 5$								
1	A-n32-k5 (16,50)	15.74	48.39	118.65	7298 s	117.38	71 s	98.9
2	A-n32-k5 (16,100)	100.08	176.58	245.67	16993 s	239.88	142 s	97.6
3	E-n51-k5 (14,50)	35.33	40.48	96.43	762 s	95.14	47 s	98.9
4	E-n51-k5 (14,100)	104.96	126.43	152.46	1975 s	151.13	94 s	99.1
5	M-n101-k10 (10,50)	72.37	73.69	127.12	764 s	126.08	90 s	99.2
6	M-n101-k10 (10,100)	171.76	176.61	213.26	1827 s	209.96	207 s	98.5
7	P-n101-k4 (14,50)	98.88	116.92	231.71	1340 s	229.14	80 s	98.9
8	P-n101-k4 (14,100)	204.15	243.25	369.73	3144 s	364.65	152 s	98.6
$T_d = 10$								
9	E-n51-k5 (14,50)	37.70	44.64	130.63	1548 s	128.87	87 s	98.6
10	E-n51-k5 (14,100)	127.39	159.16	265.99	4039 s	260.57	173 s	98.0
11	M-n101-k10 (10,50)	74.02	75.30	159.41	1519 s	157.88	167 s	99.0
12	M-n101-k10 (10,100)	213.31	219.81	311.39	3683 s	308.32	403 s	99.0
13	P-n101-k4 (14,50)	85.29	124.92	292.36	2707 s	282.57	155 s	96.7
14	P-n101-k4 (14,100)	265.88	352.37	573.85	6391 s	559.62	279 s	97.5
$T_d = 25$								
15	E-n51-k5 (14,50)	38.25	46.12	165.80	3860 s	159.85	152 s	96.4
16	E-n51-k5 (14,100)	126.59	162.25	397.23	9329 s	376.95	297 s	94.9
17	P-n101-k4 (14,50)	102.20	130.99	361.16	6644 s	349.62	231 s	96.8
18	P-n101-k4 (14,100)	257.08	363.92	759.43	15782 s	718.82	502 s	94.7
Avg.	-	118.39	148.99	276.24	4978 s	268.69	185 s	97.9

(Z_{FATS}). Overall, the TD algorithm shows on average an 80 % profit increase compared to the FATS algorithm over the instances in Table 1.

To demonstrate the superiority of our TD algorithm in the DCPTPSR, we further extend the previous instances to up to 100 stochastic incoming requests and benchmark the results with the FCFS heuristic and the FATS algorithm. In analogy to the previous experiments, the last column (Imp.) of Table 2 shows the improvement in percentage between Z_{TD} from the fifth column and the solution of the FATS algorithm in column three.

Table 2 also clarifies for these instances, that our TD learning approach dramatically outperforms the FCFS heuristic and the FATS algorithm. In particular, the TD learning approach shows a significantly greater improvement over the FATS algorithm for instances with 100 stochastic incoming requests than for instances with 25 stochastic incoming requests. For these instances, the Z_{TD} solution is around 33 % better on average compared to Z_{FATS}.

Table 2. Numerical results for the DCPTPSR with varying T_d.

Id	Instance ($\|N\|, C$)	FCFS	FATS	SDP		TD		
		Z_{FCFS}	Z_{FATS}	Z^*	t	Z_{TD}	t	Imp.(%)
$T_d = 25$								
19	A-n32-k5 (16,100)	176.12	325.26	-	-	376.79	325 s	15.8
20	A-n32-k5 (16,300)	1027.54	1471.47	-	-	1522.91	956 s	3.5
21	M-n101-k10 (10,100)	212.69	219.45	-	-	393.42	621 s	79.2
22	M-n101-k10 (10,200)	499.79	513.48	-	-	713.26	3213 s	38.9
23	E-n51-k5 (14,300)	557.46	681.25	-	-	703.45	756 s	3.3
24	P-n101-k4 (14,300)	1418.08	1580.49	-	-	1647.24	2668 s	4.2
$T_d = 100$								
25	A-n32-k5 (16,100)	179.11	330.97	-	-	546.71	612 s	65.2
26	A-n32-k5 (16,300)	1046.89	1491.04	-	-	1975.67	1878 s	32.5
27	M-n101-k10 (10,100)	211.81	218.40	-	-	401.27	1527 s	83.7
28	M-n101-k10 (10,200)	503.88	517.86	-	-	757.75	12782 s	46.3
29	E-n51-k5 (14,100)	216.75	223.25	-	-	412.80	78 s	84.9
30	E-n51-k5 (14,300)	562.17	684.47	-	-	1306.78	1748 s	90.9
31	P-n101-k4 (14,100)	395.40	452.08	-	-	748.61	775 s	65.6
32	P-n101-k4 (14,300)	1443.72	1606.93	-	-	2201.54	4859 s	37.0
Avg.	-	570.09	736.89	-	-	979.16	2343 s	46.5

To highlight the effective optimization process of our reinforcement learning algorithm we investigate learning curves of the TD algorithm (see Algorithm 2). Figure 2 shows the training process over 5,000,000 episodes for instance with Id 16 with $T_d = 10$ and instance Id 31 with $T_d = 100$ stochastic incoming requests, by analyzing the average objective value after each 10,000 independent episodes during the training process. A local regression for each 500 points is performed in Fig. 2 to clarify the graphical illustration.

Figure 2 shows that, at the beginning of the training process, the TD algorithm is unable to perform well, as actions are chosen randomly and the neural network is initialized with random weights. However, in the left diagram of Fig. 2(a) a performance of almost 96% is already achieved after 2,000,000 training episodes, which only increases marginally with the remaining 3,000,000 training episodes. By contrast, we see in the right diagram of Fig. 2(b) where 100 stochastic requests arrive that after 5,000,000 training episodes no clear convergence behavior can be observed and further training could probably still improve the performance.

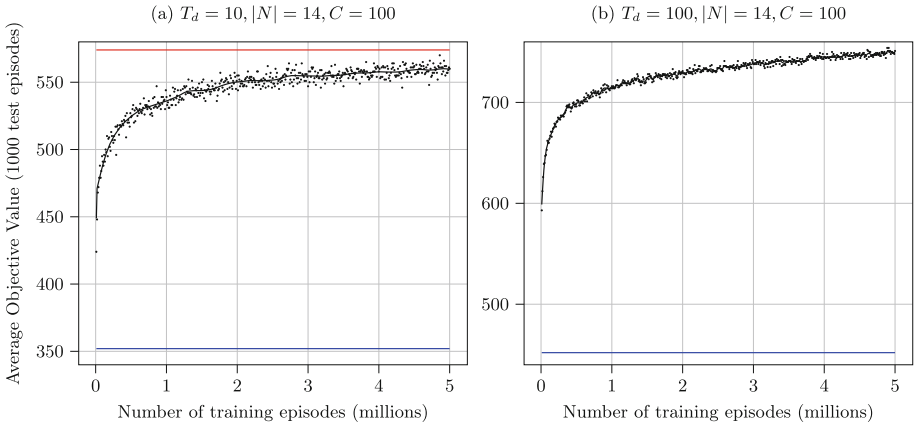


Fig. 2. Training evaluation of instance 16 (a) and 31 (b). On the x-axis, is the number of training episodes, and on the y-axis, is the average objective value over 1000 test episodes. The black dots indicate the average objective value for the TD algorithm, while the black line shows a local regression and the red line indicates the SDP solution (if found within the computational resource limit). (Color figure online)

6 Conclusion and Future Research

In this paper, we investigated the *Dynamic Capacitated Profitable Tour Problem with Stochastic Requests* (DCPTPSR) where requests characterized by a location, demand, and prize follow a discrete-time stochastic process and online decisions must be made for accepting and scheduling requests over a finite time horizon. In Sect. 2, we examined the decision-making process for accepting stochastic incoming requests within the context of revenue management and clarified the connection between tour problems with profits. Additionally, we briefly discussed reinforcement learning as a potential solution method for such problems. In Sect. 3, we described the DCPTPSR, emphasizing its objective of finding an optimal tour through all locations of the accepted requests, and contributed a Markov decision process formulation for the problem. Section 4 described how *Stochastic Dynamic Programming* (SDP) can solve the DCPTPSR to optimality. Given the inherent complexity of the problem and the computational demands associated with SDP, we proposed *Temporal Difference* (TD) learning in combination with a neural network as a solution method for the DCPTPSR. Our computational experiments in Sect. 5 demonstrated that our TD learning approach for the DCPTPSR heavily outperforms a first come first serve heuristic and the first accept traveling salesman algorithm even after relatively few training episodes. Thus we conclude that request acceptance and transportation scheduling decisions should be made simultaneously to achieve an appropriate solution quality in such a dynamic problem with stochastic incoming requests.

Building on the obtained results in the DCPTPSR, ongoing work should study the corresponding extended problem version with multiple vehicles. Fur-

thermore regarding the promising performance of TD learning in the DCPTPSR it is worthwhile to analyze if such a solution method is also useful for other similar dynamic and stochastic problems in the area of revenue management or tour scheduling problems.

References

1. Alstrup, J., Boas, S., Madsen, O.B., Vidal, R.V.: Booking policy for flights with two types of passengers. Eur. J. Oper. Res. **27**(3), 274–288 (1986)
2. Angelelli, E., Mansini, R., Rizzi, R.: The probabilistic profitable tour problem under a specific graph structure. arXiv (2022)
3. Archetti, C., Feillet, D., Hertz, A., Speranza, M.G.: The capacitated team orienteering and profitable tour problems. J. Oper. Res. Soc. **60**, 831–842 (2009)
4. Bdeir, A., Boeder, S., Dernedde, T., Tkachuk, K., Falkner, J.K., Schmidt-Thieme, L.: RP-DQN: an application of Q-learning to vehicle routing problems. In: Edelkamp, S., Möller, R., Rueckert, E. (eds.) KI 2021. LNCS (LNAI), vol. 12873, pp. 3–16. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-87626-5_1
5. Bellman, R.: Dynamic programming. Science **153**(3731), 34–37 (1966)
6. Bertsekas, D.: Dynamic programming and optimal control: Volume I, vol. 4. Athena scientific (2012)
7. Bondoux, N., Nguyen, A.Q., Fiig, T., Acuna-Agost, R.: Reinforcement learning applied to airline revenue management. J. Revenue Pricing Manag. **19**(5), 332–348 (2020)
8. Campbell, A.M., Gendreau, M., Thomas, B.W.: The orienteering problem with stochastic travel and service times. Ann. Oper. Res. **186**(1), 61–81 (2011)
9. Caspar, M., Schermer, D., Wendt, O.: Formulations for the split delivery capacitated profitable tour problem. In: International Conference on Computational Science and Its Applications, pp. 82–98. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-36805-9_6
10. Farazi, N.P., Zou, B., Ahamed, T., Barua, L.: Deep reinforcement learning in transportation research: a review. Transp. Res. Interdisciplinary Perspectives **11**, 100425 (2021)
11. Feillet, D., Dejax, P., Gendreau, M.: Traveling salesman problems with profits. Transp. Sci. **39**(2), 188–205 (2005)
12. Fleckenstein, D., Klein, R., Steinhardt, C.: Recent advances in integrating demand management and vehicle routing: a methodological review. Eur. J. Oper. Res. **306**(2), 499–518 (2023)
13. Gilmore, P.C., Gomory, R.E.: Sequencing a one state-variable machine: a solvable case of the traveling salesman problem. Oper. Res. **12**(5), 655–679 (1964)
14. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2023)
15. Ilhan, T., Iravani, S.M., Daskin, M.S.: The orienteering problem with stochastic profits. IIE Trans. **40**(4), 406–421 (2008)
16. Jaillet, P., Lu, X.: Online traveling salesman problems with rejection options. Networks **64**(2), 84–95 (2014)
17. Joe, W., Lau, H.C.: Deep reinforcement learning approach to solve dynamic vehicle routing problem with stochastic customers. In: Proceedings of the International Conference on Automated Planning and Scheduling, vol. 30, pp. 394–402 (2020)
18. Kimes, S.E.: Yield management: a tool for capacity-considered service firms. J. Oper. Manag. **8**(4), 348–363 (1989)

19. Klein, V., Steinhardt, C.: Dynamic demand management and online tour planning for same-day delivery. Eur. J. Oper. Res. **307**(2), 860–886 (2023)
20. Liang, E., Wen, K., Lam, W.H., Sumalee, A., Zhong, R.: An integrated reinforcement learning and centralized programming approach for online taxi dispatching. IEEE Trans. Neural Networks Learn. Syst. (2021)
21. McGill, J.I., Van Ryzin, G.J.: Revenue management: research overview and prospects. Transp. Sci. **33**(2), 233–256 (1999)
22. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
23. Naderi, B., Begen, M.A., Zhang, G.: Integrated order acceptance and resource decisions under uncertainty: Robust and stochastic approaches. SSRN (2022)
24. Pal, S.K., Mitra, S.: Multilayer perceptron, fuzzy sets, classifiaction. IEEE Trans. Neural Networks (1992)
25. Perron, L., Furnon, V.: Or-tools. google. OR-Tools v9.7 (2023)
26. Sawadsitang, S., Kaewpuang, R., Jiang, S., Niyato, D., Wang, P.: Optimal stochastic delivery planning in full-truckload and less-than-truckload delivery. In: 2017 IEEE 85th Vehicular Technology Conference (VTC Spring), pp. 1–5. IEEE (2017)
27. Schwind, M., Wendt, O.: Dynamic pricing of information products based on reinforcement learning: a yield-management approach. In: Jarke, M., Lakemeyer, G., Koehler, J. (eds.) KI 2002. LNCS (LNAI), vol. 2479, pp. 51–66. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45751-8_4
28. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (2018)
29. Tang, H., Miller-Hooks, E.: Algorithms for a stochastic selective travelling salesperson problem. J. Oper. Res. Soc. **56**(4), 439–452 (2005)
30. Toth, P., Vigo, D.: Vehicle routing: problems, methods, and applications. SIAM (2014)
31. Verbeeck, C., Vansteenwegen, P., Aghezzaf, E.H.: Solving the stochastic time-dependent orienteering problem with time windows. Eur. J. Oper. Res. **255**(3), 699–718 (2016)
32. Weatherford, L.R., Bodily, S.E.: A taxonomy and research overview of perishable-asset revenue management: yield management, overbooking, and pricing. Oper. Res. **40**(5), 831–844 (1992)
33. Wendt, O., Goeke, D.: Revenue Management für tourenplanungsprobleme. In: Gössinger, R., Zäpfel, G. (eds.) Management integrativer Leistungserstellung, Festschrift für H. Corsten., vol. 168, pp. 455–486. Duncker-Humblot GmbH (2014)
34. Xu, L., Wang, Q., Huang, S.: Dynamic order acceptance and scheduling problem with sequence-dependent setup time. Int. J. Prod. Res. **53**(19), 5797–5808 (2015)
35. Yan, S., Tang, C.H., Fu, T.C.: An airline scheduling model and solution algorithms under stochastic demands. Eur. J. Oper. Res. **190**(1), 22–39 (2008)
36. Zhang, M., Qin, J., Yu, Y., Liang, L.: Traveling salesman problems with profits and stochastic customers. Int. Trans. Oper. Res. **25**(4), 1297–1313 (2018)
37. Zhang, M., Wang, J., Liu, H.: The probabilistic profitable tour problem. Int. J. Enterprise Inf. Syst. (IJEIS) **13**(3), 51–64 (2017)
38. Zhang, R., Prokhorchuk, A., Dauwels, J.: Deep reinforcement learning for traveling salesman problem with time windows and rejections. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2020)
39. Zheng, J., et al.: Modeling stochastic service time for complex on-demand food delivery. Complex Intell. Syst. **8**(6), 4939–4953 (2022)