A dynamic and probabilistic orienteering problem

E. Angelelli
C. Archietti
C. Filippi
M. Vindigni

WPDEM 1/2019

A dynamic and probabilistic orienteering problem

Enrico Angelelli (1) *Claudia Archetti* (1,2) *Carlo Filippi* (1) *Michele Vindigni* (1)

(1) *Department of Economics and Management, University of Brescia, Italy,* {`enrico.angelelli, claudia.archetti, carlo.filippi, michele.vindigni`}`@unibs.it`

(2) *Department of Information Systems, Decision Sciences and Statistics, ESSEC Business School, France*

Abstract

We consider an online version of the orienteering problem, where stochastic service requests arise during a first time interval from customers located on the nodes of a graph. Every request must be accepted/rejected in real time. Later, a vehicle must visit the accepted customers during a second time interval. Each accepted request implies a prize, depending on the customer, and a service cost, depending on the routing decisions. Moreover, an accepted request implies a reduction of the routing time available for possible future requests. Each acceptance/rejection decision is made to maximize the expected profit, i.e., the difference between expected prices and expected service costs.

We derive analytical expressions for the exact computation of the optimal policy. Since an exact policy computation is intractable, we design and test several heuristic approaches, including static approximation, simple greedy (non-anticipatory) methods, Sample Average Approximation (SAA) of the objective function using Monte Carlo sampling of future events. We perform extensive computational tests on the proposed algorithms and discuss the pros and cons of the different methods on the specific problem.

Keywords: Routing, Heuristics, Random requests, Dynamic vehicle routing, Orienteering problem.

1 Introduction

Dynamic and stochastic routing problems have received increasing attention in recent years. On the academic side, efficient algorithmic approaches and increased computational power allow to face the complexity of stochastic models [Psaraftis et al., 2016]; on the practical side, e-commerce and technological advances require flexible and real-time decision making [Ulmer et al., 2018]. Moreover, time-constrained deliveries have been one of the fastest growing segments of the delivery business [Campbell and Thomas, 2008], while it is argued that selective vehicle routing problems are more appropriate than the conventional routing problems in handling uncertainty with limited resources [Allahviranloo et al., 2014].

We study a time-constrained, selective, dynamic and stochastic routing problem. Consider a transportation company that collects and manages requests for long-haul transportation from numerous customers. Usually, shipments from a certain geographical zone are collected by a vehicle and delivered to a carrier's facility. Here, shipments are consolidated according to their destination and sent to other facilities by long-haul transportation [Pillac et al., 2013].

In particular, we consider the manager of a specific facility who has to organize the picking-up of goods for the next-day overnight shipments. At the beginning of the day, the manager has a portfolio of *mandatory requests* and can figure out a portfolio of *spot requests*. Mandatory requests arise from customers with a consolidated relationship of mutual trust with the carrier and use the carrier as a preferential one. These customers typically have a contract for a continuous and regular service. It is understood that all these requests are acquired, constitute an obligation for the carrier and provide a certain revenue. Spot requests arise dynamically on an e-market from occasional customers. A spot request may be accepted or rejected, but the decision must be taken in real time, or after a short negotiation. If a spot request is accepted, then it becomes binding on both parties.

We consider a decision problem encompassing two successive days. During the first day, the manager receives spot requests and, for each request, he/she has the opportunity to either accept or reject the corresponding shipment. During the second day, the customers associated with mandatory requests and accepted spot requests must be visited by a vehicle to pick-up goods. An accepted request implies a *prize*, i.e., the revenue obtained from the shipment service, and a *operational cost*, i.e., the cost of the deviation in vehicle's route to reach the customer and picking-up the goods. Moreover, accepting a request reduces the vehicle's *time budget* [Ulmer et al., 2018], and thus may prevent the acceptance of future and maybe more profitable requests. Indeed, if a vehicle returns to the facility too late on the second day, it may delay all the scheduled overnight shipments. Consequently, any vehicle must return to the facility within a sharp deadline. The effect of an accepted request on the delivery phase following the overnight shipment depends on the package destination. We assume that the manager cannot control this effect, and thus we do not consider it in this study.

Our aim is to design and implement a service policy that helps the manager to decide about the acceptance or rejection of a spot request according to a criterion of maximum expected profit, where the profit is the algebraic sum of prize and operational cost. Notice that the acceptance/rejection decision is constrained by and affects routing decisions that will be implemented at a later time. An integrated decision process,

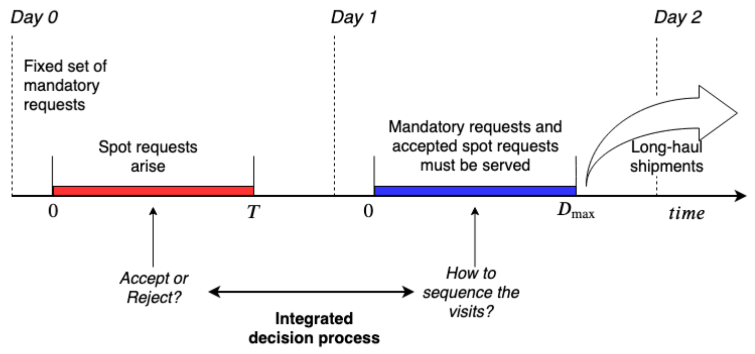
Day 0
Day 1
Day 2
Fixed set of mandatory requests
Spot requests arise
Mandatory requests and accepted spot requests must be served
Long-haul shipments
0
T
0
$D_{\max}$
time
Accept or Reject?
How to sequence the visits?
Integrated decision process

Figure 1: Graphical illustration of the decision process.

possibly anticipatory of future events, is then required.

For simplicity, we assume that the manager has a single vehicle for the picking-up service, and that the spatial location of both mandatory and spot requests is known in advance. Thus, we model the problem on a complete directed graph where nodes correspond to requests. There are two types of nodes: mandatory and spot. Mandatory nodes must be visited, spot nodes may be visited if they originate a request. A distinguished feature of our problem is that the decision process about acceptance/rejection of spot requests takes place before the actual routing is fulfilled. We thus work with two successive time intervals: a decision time interval $(0,T)$, placed in day 0, and a routing time interval $(0,D_{\max})$, placed in day 1, see Figure 1. During the decision time interval, spot requests arise randomly according to a known, time-dependent, probability law and the decision maker must accept/reject them in real time; during the routing time interval, a vehicle starts from the facility, visits the mandatory nodes and the spot nodes that have been accepted, collects the parcels, and return to the facility within the time limit. We assume that the traveling and service times are deterministic. Moreover, we assume that the quantity of goods associated with each request is comparatively small, so that the vehicle may be regarded as uncapacitated. Note that this latter assumption is consistent with many practical settings. Indeed, last-mile pickup and delivery of light parcels is characterized by tightness of customer service times or maximum working time for drivers, while capacity is rarely binding.

In summary, we study a dynamic and stochastic extension of the Orienteering Problem (OP) [Gunawan et al., 2016], where requests for service arise randomly one at a time along a given decision time interval according to a known probability law, and the acceptance/rejection of every single request must be made in real time. The objective is the maximization of the expected profit, where the profit is given by the difference between the expected total prize (including possible future requests) and the expected traveling cost (including the visits to possible future requests).

The main contributions of this paper can be summarized as follows:

- We introduce a new problem, namely the Dynamic and Probabilistic Orienteering Problem (DPOP), that generalizes the well-known OP by introducing dynamic acceptance/rejection of probabilistic customers. Differently from most dynamic routing problems considered in the literature (see Section 2), in DPOP online decisions have to be taken before the vehicle leaves the depot, so that the whole route can be revised after each acceptance/rejection decision. The objective to be maximized is the expected profit.
- We formally analyze the problem, fully characterizing the distribution of the next event and defining the optimal decision policy accordingly.
- We develop several heuristic approaches, including static approximation, simple greedy (non-anticipatory) methods, Sample Average Approximation (SAA) of the objective function using Monte Carlo sampling of future events.
- We perform extensive computational tests on the proposed algorithms and discuss the pros and cons of the different methods on the specific problem.

The paper is organized as follows. After revising the most relevant literature in Section 2, we formalize the decision problem in Section 3, where the distribution of the next event is characterized and an analytic recursive formula for the evaluation of the optimal strategy is obtained. In Section 4, we propose several approximated algorithmic approaches, whose computational performance is evaluated and discussed in Section 5. Finally, some conclusions are drawn in Section 6.

2 Literature review

The DPOP is a particular dynamic and stochastic vehicle routing problem (DSVRP) which generalizes the Orienteering Problem (OP). Specifically, the stochastic feature refers to the presence of random requests from a set of potential customers, while the dynamic feature refers to the fact that random requests reveal over time and need immediate response. Moreover, a specific feature of the DPOP is that every acceptance/rejection decision is taken before the vehicle actually starts the tour. As a consequence, the order of visit of the accepted customers can be modified thoroughly after any acceptance decision. On one hand, this feature makes the problem very different from most dynamic routing problems, where requests arise while the vehicle is on the road. On the other hand, the same feature is shared by routing problems related to *attended home delivery* (AHD) services, where, however, the main issue is the management of delivery time windows. In this section, we review the contributions on stochastic and dynamic OPs and on DSVRPs with stochastic requests mostly related to our problem.

Concerning stochastic orienteering problems, the reference closest to the present work is by Angelelli et al. [2017], who study the Probabilistic OP (POP). The problem is formulated on a directed graph, where each customer is available for visit only with a certain probability. In a first stage, a node subset has to be selected and a corresponding a priori path has to be determined such that the server can visit all customers in the subset and reach the destination without exceeding a time budget. In a second stage, after the list of available customers is revealed, the server follows the a priori path by skipping the absent nodes. The POP consists in determining a first-stage solution that maximizes the expected profit of the second-stage path, where the expected profit is the difference between the expected total prize and the expected total cost. A branch-and-cut exact approach and several matheuristic methods are proposed. In fact, the POP is a static variant of the problem we study in this paper, where the probability of request from a customer is fixed and an a priori route based on all possible requests is evaluated. In our DPOP, the probability of a request is a function of time, and once a request arises, we have to decide immediately whether to accept it or not.

Other stochastic variants of the OP are proposed by İlhan et al. [2008], who discuss the OP with Stochastic Profits (OPSP); Campbell et al. [2011], who study the OP with Stochastic Travel and Service times (OPSTS); Evers et al. [2014], who consider the OP with Stochastic Weights (OPSW), where weights are associated with travel costs, travel time (including service time) or fuel consumption on arcs; Zhang et al. [2014], who study the Stochastic OP with Time Windows (SOPTW), where an uncertain waiting time at a customer results from a possible queue of competitors arrived earlier. Notice that all the OP mentioned so far are static problems. Zhang et al. [2018] consider a dynamic variant of the SOPTW, where the server decides dynamically the next customer to visit, allowing the visit to a customer previously abandoned. The decision strategy is identified by an approximate dynamic programming approach based on rollout algorithms. We refer to Gunawan et al. [2016] for additional references to stochastic extensions of the OP.

Concerning AHD problems, Campbell and Savelsbergh [2005] are the first to face grocery delivery services by optimization techniques. Their setting is similar to ours: requests may arise from known points on a plane and according to known probabilities, that reduce over time; the objective is the same as the DPOP, i.e., expected profit. However, there are important differences: requests are associated with narrow time windows, there are multiple vehicles and capacity constraints. Concerning the algorithmic approach, Campbell and Savelsbergh [2005] propose an insertion heuristic to determine the feasibility of a request and different strategies to evaluate the expected profit associated with a decision. When a new request occurs at time t, their insertion heuristic considers a static problem where the prize associated with every not-yet-materialized request is adjusted according to its probability of occurrence computed at time t. Hence, Campbell and Savelsbergh [2005] face a problem more general than the DPOP, but they analyze a single algorithmic approach. They consider neither simpler greedy heuristics, nor Sample Average Approximation methods, as we do in our study. Moreover, no theoretical analysis is provided in [Campbell and Savelsbergh, 2005], whereas we provide an analytical description of the stochastic process embedded in our optimization problem. The successive studies on AHD problems are mainly focused on time interval management and incentive schemes to induce customer to choose the best time window (from a routing viewpoint). See, e.g., Campbell and Savelsbergh [2006] and Agatz et al. [2011].

Azi et al. [2012] consider a dynamic VRP where vehicles perform multiple routes during their workday. Requests arrive according to a Poisson process and must be accepted or rejected in real time. An accepted request is inserted in one of the routes that will be executed later. The authors adapt an ALNS algorithm developed for the static version of the problem to the dynamic version. The expected gain obtained from the acceptance of a new request is evaluated by inserting it on a number of request scenarios, generated at the start of the algorithm. As in the DPOP, every vehicle starts from the depot with a fixed sequence of customers to visit and the objective is the expected profit. Differently from DPOP, the decision interval and the routing interval are overlapped, and multiple vehicles start asynchronously over time.

Ulmer and Thomas [2019] consider the capacitated customer acceptance problem with stochastic requests (CAPSR), which is very similar to DPOP. As in the DPOP, they consider a single vehicle and there are no time windows associated with requests. However, the vehicle is capacitated, the objective is the expected revenue (no costs are considered), and a (fixed) service time is spent at each visited customer. They also assume that requests arise according to a Poisson process and that prize and demand associated with each request are unknown before the request arises. Finally, the CAPSR is not the focus of the paper, primarily devoted to the development of a general-purpose "meso-parametric" value function approximation.

Concerning general DSVRPs, a first distinction depends on whether stochastic information about possible future requests is available and used in the decision process or not. In the former case, we have anticipatory approaches, in the latter, not anticipatory approaches [Ulmer et al., 2018]. Notice that the approach used in this paper is anticipatory, since decisions take into account the possible future requests. Anticipatory approaches are usually based on sampling. More specifically, future customer requests are sampled on the demand space or the graph nodes, according to the assumed probability distribution. This allows to better evaluate the impact of current decisions, but requires a significant computational effort. Sampling approaches for DSVRPs with random requests are proposed by Bent and Hentenryck [2004], Flatberg et al. [2007] and Sungur et al. [2010]. Short-term sampling is used by Ghiani et al. [2009] for a dynamic pickup and delivery problem. Sampling based on historical data is used by Hvattum et al. [2006] to solve a real-world case study. Ghiani et al. [2012] propose an anticipatory insertion waiting approach, based on the concept of center of gravity of the potential future requests. Finally, Ulmer et al. [2018] design an anticipatory time budgeting heuristic, drawing on methods of approximate dynamic programming. Notice that, among the cited works, only [Ghiani et al., 2012] consider random requests from nodes on a graph as in our work, whereas all other papers consider a probabilistic distribution of requests on a continuous 2-dimensional space. We refer to Ritzinger et al. [2016] and to Psaraftis et al. [2016] for recent surveys on DSVRPs.

3 Problem description

Let $V = \{1, 2, \ldots, n\}$ be the set of n potential request nodes and let $V_M \subseteq V$ be the set of mandatory request nodes. Mandatory requests arise from regular customers and must be satisfied. The remaining potential request nodes $V_S = V \setminus V_M$ correspond to spot requests. We will call them *optional* request nodes from now on. In general, there is no one-to-one correspondence between nodes in V and actual customers. More precisely, a regular customer r (originating a mandatory request) might sometimes place a spot request. In this case, two distinct nodes $u \in V_M$ and $v \in V_S$ would be associated with r. Moreover, an optional request node $w \in V_S$ may be used to represent a group of spot customers located in a geographical unit.

Consider first a deterministic setting. Assume that we have fixed a subset W of request nodes to serve, $V_M \subseteq W \subseteq V$, with all requests from nodes in W confirmed. Consider a directed graph $G = (\overline{V}, A)$, with node set $\overline{V} = V \cup \{0, n+1\}$, where 0 is the *origin* and $n+1$ is the *destination* (possibly coinciding with the origin), and arc set $A = \{(i, j) : i \in V \cup \{0\}, j \in V \cup \{n+1\}\}$. Let p_i be the prize associated with node $i \in V$, and let $\tau_{i,j}$ be the traveling time associated with arc $(i, j) \in A$. We assume also a traveling cost proportional to the traveling time, so that traveling through arc (i, j) implies a cost $C \cdot \tau_{i,j}$, where C is a properly defined positive constant. A vehicle must start from the origin, visit all nodes in W according to some order, and then end at the destination.

Let $\tau_{\min}(W)$ denote the minimum duration of the server's path, i.e., the duration of a minimum cost elementary path from 0 to $n+1$ visiting all nodes in W. Then, set W is *feasible* if and only if $\tau_{\min}(W) \leq D_{\max}$. Moreover, $C \cdot \tau_{\min}(W)$ is the minimum possible cost of the server's path. In this deterministic setting, the profit associated with W is the difference between the total collected prize and the minimum traveling cost:

$$P(W) = \sum_{i \in W} p_i - C \cdot \tau_{\min}(W).$$

Moving to a stochastic context, we assume that requests are placed by optional customers during a time window $(0, T)$. A random variable T_i, representing the time in which the service request is advanced, is associated with every node $i \in V$. For mandatory nodes, the distribution is degenerate: $T_i = 0$ with probability 1 for all $i \in V_M$. For all $i \in V_S$, we have:

$$T_i = \begin{cases} \overline{T}_i & \text{with probability } \theta_i, \\ T+1 & \text{with probability } 1-\theta_i, \end{cases}$$

where $\overline{T}_i$ is a continuous random variable with known Cumulative Density Function (CDF), denoted as $\overline{F}_i$, and known Probability Density Function (PDF), denoted $\overline{f}_i$. Thus, we have

$$\overline{F}_i(t) = \begin{cases} 0 & \text{if } t \leq 0, \\ \int_0^t \overline{f}_i(x)dx & \text{if } 0 < t \leq T, \\ 1 & \text{if } t > T. \end{cases}$$

For all $i \in V_S$, θ_i is the probability that a request arises from i during the time horizon $(0, T)$; $T+1$ is a conventional point in time where "late" or "no-show" requests are placed. The CDF of T_i can thus be written as

$$F_i(t) = \theta_i \overline{F}_i(t) + (1-\theta_i)I(t), \qquad (1)$$

where $I(t)$ is an indicator function defined as

$$I(t) = \begin{cases} 1 & \text{if } t \geq T+1, \\ 0 & \text{otherwise.} \end{cases}$$

We assume that requests arise independently among nodes.

We are facing the following decision problem. At time $t \in (0,T)$, an optional request may arise from a node $i \in V_S$. In real time, we have to decide whether to accept the request or not. The decision criterion is the maximization of the expected profit.

At time $t \in (0,T)$, we have a set $R(t)$ of nodes whose requests have been *received*, $V_M \subseteq R(t) \subseteq V$, and a set of nodes $W(t)$ whose requests have been *received and accepted*, $V_M \subseteq W(t) \subseteq R(t)$. We assume that $\tau_{\min}(W(t)) \leq D_{\max}$. During the time interval $(0,T)$, a *proper event* is the arrival of a request from a node in V_S, the *final event* is the achievement of the time horizon T. We have at most $h = |V_S|$ proper events, and consequently at most h decisions to make. Moreover, exactly one set $W(t)$ is associated with each $R(t)$. Thus, we introduce a counter k of events, with $1 \leq k \leq h+1$, and describe the state of the system at time $t \in (0,T)$ by the triple (t^k, R^k, W^k), where t^k is the time of occurrence, R^k is the set of nodes that have already originated a request, W^k is the set of nodes whose request has been accepted. The initial state is $(t^0, R^0, W^0) = (0, V_M, V_M)$.

When a proper event occurs, a service policy π decides whether to accept the new request or not; when the final event occurs, π decides the order of node visits in the final set W^k. A proper event is identified by the ordered pair (t,i), where t is the time of occurrence and i is the node asking for service. Let $\pi(t^k, R^k, W^k, t, i) \in \{\emptyset, \{i\}\}$ denote the output of the service policy when the proper event (t,i) occurs with system at state (t^k, R^k, W^k), with $t > t^k$. Then, when a proper event (t,i) occurs, the system evolves as follows:

$$(t^k, R^k, W^k) \longrightarrow (t^{k+1} = t, R^{k+1} = R^k \cup \{i\}, W^{k+1} = W^k \cup \pi(t^k, R^k, W^k, t, i)).$$

We denote as $EP_\pi(t^k, R^k, W^k, t, i)$ the expected profit produced by the service policy π when the proper event (t,i) occurs with system at state (t^k, R^k, W^k).

The final event is identified by the ordered pair $(T, \emptyset)$. When the final event occurs, it is easy to define the maximum gain that can be obtained by any policy:

$$EP_\pi(t^k, R^k, W^k, T, \emptyset) = \sum_{i \in W^k} p_i - C \cdot \tau_{\min}(W^k). \tag{2}$$

For proper events, the expected gain function can be defined only characterizing the distribution of the next event. Assume that the proper event (t^{k+1}, i^{k+1}) occurs with system at state (t^k, R^k, W^k), with $t^{k+1} > t^k$.

Notice that the next event will occur at time

$$t_{\min}(t^{k+1}, R^{k+1}) = \min\Big\{\min\{T_j : j \in V \backslash R^{k+1}, T_j > t^{k+1}\}; T\Big\}$$

where $R^{k+1} = R^k \cup \{i^{k+1}\}$. If $t_{\min}(t^{k+1}, R^{k+1}) = T$, then the next event will be the final event, and formula (2) can be used to evaluate the expected profit associated with the acceptance or rejection of i^{k+1}. If $t_{\min}(t^{k+1}, R^{k+1}) < T$, then the next event will be proper. In this case, to evaluate the expected profit associated with the acceptance or rejection of i^{k+1}, we need to characterize the probability distribution of the next event conditioned to the current system state. This is done in the following theorem, whose proof is given in Appendix A.

Theorem 1 *Assume that the proper event (t^{k+1}, i^{k+1}) occurs with system at state (t^k, R^k, W^k). Then, the probability that the next event will be proper and will involve node $i \in V \backslash R^{k+1}$ is*

$$p_{\min}(i|t^{k+1}, R^{k+1}) = \int_{t^{k+1}}^{T} \left[\prod_{j \in V \backslash (R^{k+1} \cup \{i\})} \left(\frac{1 - \theta_j \overline{F}_j(t)}{1 - \theta_j \overline{F}_j(t^{k+1})} \right) \right] \frac{\theta_i \overline{f}_i(t)}{1 - \theta_j \overline{F}_i(t^{k+1})} dt \tag{3}$$

Moreover, the probability that the next event will be the final event is

$$p_{\min}(\emptyset|t^{k+1}, R^{k+1}) = \prod_{j \in V \backslash R^{k+1}} \left(\frac{1 - \theta_j}{1 - \theta_j \overline{F}_j(t^{k+1})} \right). \tag{4}$$

To simplify notation, for all $i \in V \backslash R^{k+1}$, let us denote by $\phi_{i|t^{k+1}, R^{k+1}}(t)$ the function integrated in (3), i.e.,

$$\phi_{i|t^{k+1}, R^{k+1}}(t) = \left[\prod_{j \in V \backslash (R^{k+1} \cup \{i\})} \left(\frac{1 - \theta_j \overline{F}_j(t)}{1 - \theta_j \overline{F}_j(t^{k+1})} \right) \right] \frac{\theta_i \overline{f}_i(t)}{1 - \theta_j \overline{F}_i(t^{k+1})}.$$

Then, the expected profit function of the optimal policy $\pi^\star$ can be expressed recursively as follows:

$$EP_{\pi^\star}(t^k, R^k, W^k, t^{k+1}, i^{k+1}) =$$

$$\max\Big\{\textstyle\sum_{i\in V\setminus R^{k+1}} \displaystyle\int_{t^{k+1}}^{T} EP_{\pi^\star}\left(t^{k+1}, R^{k+1}, W^k \cup \{i^{k+1}\}, t, i\right) \phi_{i|t^{k+1},R^k}(t)dt$$

$$+ EP_{\pi^\star}\left(t^{k+1}, R^{k+1}, W^k \cup \{i^{k+1}\}, T, \emptyset\right) p_{\min}(\emptyset|t^{k+1}, R^{k+1});$$

$$\textstyle\sum_{i\in V\setminus R^{k+1}} \displaystyle\int_{t^{k+1}}^{T} EP_{\pi^\star}\left(t^{k+1}, R^{k+1}, W^k, t, i\right)\ \phi_{i|t^{k+1},R^k}(t)dt$$

$$+ EP_{\pi^\star}\left(t^{k+1}, R^{k+1}, W^k, T, \emptyset\right) p_{\min}(\emptyset|t^{k+1}, R^{k+1})\Big\}$$

where the request from i^{k+1} is accepted or not according to the maximization of the expected value of the gain function.

Except for trivial cases, an exact evaluation of the above formula is intractable. Hence, we have to consider and evaluate suboptimal policies.

4 Solution approaches

Each time an event occurs, we should maximize the recursive expected profit function $EP_{\pi^\star}$. As evaluating this function is in general computationally prohibitive, we have to rely on heuristic approximations.

In general, one may think about two classes of solution approaches:

- *Static approaches*. The idea in this case is to develop a solution approach that is applied only once, at time 0, with the information available at time 0. This means that the solution will determine a priori which are the requests for which the service will be provided if requested and those for which the service will not be provided in any case. Then, the final solution is the one where the requests served are those that have actually arisen among the selected ones.

- *Dynamic approaches*. The idea in this case is to reoptimize the problem whenever an event occurs, i.e., whatever is the decision rule, it is applied each time a new request is placed taking into account what are the requests for which a decision has already been taken and what is the information on remaining requests at the time the reoptimization is run.

4.1 Static approach

We propose one algorithm that exploits the information available at time 0, solves a problem, and use the corresponding solution to take future decisions. Thus, this approach does not take into account the dynamic nature of the DPOP. In particular, the problem solved at time 0 corresponds to the Probabilistic Orienteering Problem (POP) studied in [Angelelli et al., 2017]. In fact, the POP can be seen as the 'static' version of the DPOP where each customer is associated with a probability of placing a request which does not depend on time and the decision is taken at time 0 and never modified. The objective is to determine a route that does not exceed $D_{\max}$ and maximizes the expected value of the difference between the collected profit and the traveling cost.

The POP is an extremely difficult problem, as shown in [Angelelli et al., 2017], for which only small instances can be solved to optimality, Thus, we opted for a heuristic solution for the POP which works as follows. For each customer i, we multiply the prize p_i by θ_i. This way, we obtain a deterministic problem which is solved by adapting the heuristic algorithm presented in [Chao et al., 1996] for the OP, where the visit to customers in V_M is imposed and the duration of the route does not exceed D_{max}.

The solution determines the subset of requests for which the service will be provided in case they appear. If a request that has not been selected by the POP solution appears, then the service will be denied. In the following, we call this algorithm `OP-One-Shot`.

4.2 Dynamic approaches

We propose two types of dynamic approaches for the solution of the DPOP. The idea behind each of the approach is to recompute an approximated value of EP_π each time a proper event (request from an optional customer) occurs. The goal is to determine whether to accept or not the new request. Notice that, considering time as a continuous variable, the occurrence of two simultaneous requests has probability zero. Thus, a common feature of the two approaches is that they take a decision on one request at a time. Another common feature is that, before any decision, the probabilities of the future possible optional requests are recomputed.

Recall that when the $(k+1)$-th request arises at time t^{k+1} from node i^{k+1}, we have a set R^k of requests already received and a set $W^k \subseteq R^k$ of requests accepted. In addition, $R^{k+1} = R^k \cup \{i^{k+1}\}$. The probability θ_i^{k+1} that a request will arise from a node $i \in V \backslash R^{k+1}$ can be computed using the CDF $F_{i|t^{k+1}}$ of the random variable T_i, given that no request has occurred from i in $[0, t^{k+1}]$. For all $t \in (t^{k+1}, T)$, using (1) we have:

$$F_{i|t^{k+1}}(t) = \frac{\mathbb{P}(t^{k+1} < T_i \leq t)}{\mathbb{P}(T_i > t^{k+1})} = \frac{F_i(t) - F_i(t^{k+1})}{1 - F_i(t^{k+1})} = \frac{\theta_i(\overline{F}_i(t) - \overline{F}_i(t^{k+1}))}{1 - \theta_i \overline{F}_i(t^{k+1})}. \tag{5}$$

Thus,

$$\theta_i^{k+1} = F_{i|t^{k+1}}(T) = \frac{\theta_i(1 - \overline{F}_i(t^{k+1}))}{1 - \theta_i \overline{F}_i(t^{k+1})}. \tag{6}$$

The dynamic approaches are the following:

1. *Myopic approaches.* These approaches take a decision by analyzing the information available at the time the decision is taken.

2. *Monte Carlo (MC) approaches.* These approaches use MC simulation to predict the policy expected performance over accept/reject decision. In particular, after arrival of request i, a number of scenarios about future possible request arrivals are generated. For each of them the value obtained by a policy is evaluated under both rejection and acceptance of request i.

We now detail the different approaches belonging to the two classes.

4.2.1 Myopic approaches

A myopic approach takes a decision on the proper event (t^{k+1}, i^{k+1}) on the basis of the information available at time t^{k+1}.

OP-Multi-Shot. `OP-Multi-Shot` consists in solving a POP each time a request arises. More precisely, when the event (t^{k+1}, i^{k+1}) occurs, we solve a POP instance where: (a) the nodes in W^k are mandatory; (b) node i^{k+1} is optional but has probability 1; (c) nodes $i \in V \backslash R^{k+1}$ are optional with probability θ_i^{k+1}, see (6). We use the same solution algorithm used in `OP-One-Shot`. Request from i^{k+1} is accepted if and only if it is served in the POP solution.

Greedy algorithms. Greedy algorithms take a decision on the basis of simple rules. We defined four rules, which generated four greedy algorithms. For any path $\mathscr{P}$ from origin 0 to destination $n+1$, let $d(\mathscr{P})$ be the duration of $\mathscr{P}$, and let $\Delta_{i,\mathscr{P}}$ be the cheapest insertion time of inserting request node i in path $\mathscr{P}$. Let

$\mathscr{P}^k$ be the path built at time t^k to visit request nodes in W^k. Initially, $\mathscr{P}^0$ is the minimum cost elementary path from 0 to $n+1$ visiting all nodes in $W^0 = V_M$. Then, the four greedy algorithms are the following:

1. `Profitable Greedy`: Request i^{k+1} is accepted if $p_{i^{k+1}} - C \cdot \Delta_{i^{k+1},\mathscr{P}^k} \geq 0$ and $d(\mathscr{P}^k) + \Delta_{i^{k+1},\mathscr{P}^k} \leq D_{max}$. The idea is that the request of customer i^{k+1} is accepted only if it provides a positive net benefit on the current solution represented by $\mathscr{P}^k$. This rule can be seen as a 'conservative' rule as the acceptance is limited to requests that provide an 'immediate' net benefit, i.e., a benefit that does not depend on what may happen in the future.

2. `Feasible Greedy`: Request i^{k+1} is accepted if $d(\mathscr{P}^k) + \Delta_{i^{k+1},\mathscr{P}^k} \leq D_{max}$. This rule may be seen as an 'optimistic' rule as it accepts all requests that can be feasibly satisfied, even in the case they do not provide an immediate net benefit, in the hope of advantageously combine them with future accepted requests.

3. `Profitable Look-ahead Greedy`: This rule uses some heuristic evaluations on the expected effects of accepting or rejecting the request from node i^{k+1}. The basic idea is to accept a request not immediately profitable if it might produce synergistic effects with other requests that have not yet materialized. To explain how the rule works, we need some additional notation.

   - Let $\mathscr{P}^{k+1}$ be the tour obtained from $\mathscr{P}^k$ when request i^{k+1} is inserted through cheapest insertion. Notice that $d(\mathscr{P}^{k+1}) = d(\mathscr{P}^k) + \Delta_{i^{k+1},\mathscr{P}^k}$. In the following definitions, $\mathscr{P}$ can be either $\mathscr{P}^k$ or $\mathscr{P}^{k+1}$.
   - $I(\mathscr{P}) = \{i \in V \backslash R^{k+1} : p_i - C \cdot \Delta_{i,\mathscr{P}} \geq 0\}$ is the set of 'interesting' possible future requests, i.e., requests that would provide an increase in the objective function if they were accepted and inserted in path $\mathscr{P}$ by cheapest insertion.
   - $AEP(I(\mathscr{P})) = \frac{1}{|I(\mathscr{P})|} \left(\sum_{i \in I(\mathscr{P})} \theta_i^{k+1} (p_i - C \cdot \Delta_{i,\mathscr{P}}) \right)$ is an estimated expected profit from a request in $I(\mathscr{P})$.
   - $N_{call}(I(\mathscr{P})) = \sum_{i \in I(\mathscr{P})} \theta_i^{k+1}$ is an estimate of the number of customers in $I(\mathscr{P})$ that will place a request.
   - $\tau_{\min}(V)$ is the duration of the minimum cost elementary path from 0 to $n+1$ visiting all nodes in V.
   - $UC = \frac{\tau_{\min}(V)}{|V|}$ is the estimated time needed to visit one request in V.

- $N_{serve}(I(\mathscr{P})) = \frac{D_{\max} - d(\mathscr{P})}{UC}$ is an estimate of the number of requests in $I(\mathscr{P})$ that may be served given the residual time availability.
- $P(I(\mathscr{P})) = AEP(I(\mathscr{P})) \cdot \min\{N_{call}(I(\mathscr{P})), N_{serve}(I(\mathscr{P}))\}$ is an estimate of the profit that may be obtained from requests in $I(\mathscr{P})$.

The `Profitable Look-ahead Greedy` works as follows:

(a) If $d(\mathscr{P}^k) + \Delta_{i^{k+1},\mathscr{P}^k} > D_{max}$ then discard i^{k+1} and stop.

(b) If $p_{i^{k+1}} - C \cdot \Delta_{i^{k+1},\mathscr{P}^k} \geq 0$ then accept $i^{k+1}$and stop.

(c) If $p_{i^{k+1}} - C \cdot \Delta_{i^{k+1},\mathscr{P}^k} + P(I(\mathscr{P}^{k+1})) \geq P(I(\mathscr{P}^k))$ then accept i^{k+1}, else discard it.

4. `Feasible Look-ahead Greedy`: This greedy heuristic works as the `Profitable Look-ahead Greedy` except for point (b) which is ignored. The rationale is to give more emphasis to future synergistic effects than to current local positive effects.

4.2.2 Monte Carlo (MC) approaches

At each proper event (t^{k+1}, i^{k+1}) MC approaches generate scenarios for requests that may still appear. We present two approaches. Both of them are based on the following four phases:

1. *Feasibility check*: if $d(\mathscr{P}^k) + \Delta_{i^{k+1},\mathscr{P}^k} > D_{max}$, then i^{k+1} is rejected and the procedure stops.

2. *Monte Carlo time sampling*: It randomly generates a set $\mathscr{S}$ of scenarios where, for each request in $V \backslash R^{k+1}$, the time at which the request arises is determined. More precisely, each scenario s is a sequence of events $(t_{i,s}, i)$ where $i \in V \backslash R^{k+1}$ and $t^{k+1} < t_{i,s} < T$. Note that an arrival time is generated for every $i \in V \backslash R^{k+1}$, but only requests with arrival time less than T are kept in scenario s.

3. *Evaluation*: For each scenario $s \in \mathscr{S}$, it solves two distinct problems: one in which the service to i^{k+1} is forced and one in which the service to i^{k+1} is forbidden. Let $z_{in}(s)$ and $z_{out}(s)$ be the value of the solutions obtained for scenario s when forcing or forbidding the service of i^{k+1}, respectively.

4. *Decision*: If $\sum_{s \in \mathscr{S}} z_{in}(s) \geq \sum_{s \in \mathscr{S}} z_{out}(s)$, then i^{k+1} is served otherwise the service is declined. This means that i^{k+1} is served if the expected profit of accepting i^{k+1} is higher than the one of rejecting its service.

The two approaches that we propose are the following.

MC Greedy. In `MC Greedy` approach, Phase 3 (*Evaluation*) is implemented as follows: given all decisions taken up to time t^{k+1}, a greedy algorithm is used to decide on requests in scenario s. Thus, as we have four myopic greedy algorithms, we obtained four `MC Greedy` algorithms. We call each algorithm with the name of the corresponding myopic greedy algorithm preceded by `MC`.

MC OP-Multi-Shot. In `MC OP-Multi-Shot`, Phase 3 (Evaluation) is implemented as follows: given all decisions taken up to time t^{k+1}, a new deterministic static problem is generated using the events in scenario s. In particular, all request in W^k are set mandatory with their respective prizes, request i^{k+1} is either mandatory or forbidden, according to which problem is solved, and requests in current scenario s are considered as deterministic requests with a prize equal to the expected prize (prize multiplied by the probability at time t^{k+1}). We solved the deterministic problem by adapting the heuristic algorithm presented in [Chao et al., 1996] for the OP.

5 Computational tests

In this section, we present the computational tests that we made in order to verify the efficacy of the solution algorithms presented in Section 4 and to identify whether the performance of the different solution approaches depends on problem characteristics. We first describe the instances on which tests have been performed in Section 5.1 and then show and comment the results in Section 5.2.

5.1 Test instances

As the DPOP has not been studied before in the literature, there are no benchmark instances to work with. We adapted the benchmark instances for the Probabilistic Orienteering Problem (POP) introduced in [Angelelli et al., 2017] and modified them to take into account the characteristics of the DPOP and to study the impact of its peculiarities on solution algorithms. To ease readability, we start by describing the instances generated in [Angelelli et al., 2017] for the POP.

Coordinates for customers and the depot are taken from TSP benchmark instances provided in the TSPLIB95 library available at the following url: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95. We assume unit speed for the server, that is $C = 1$, so that time and distance coincide. We selected the subset of instances with a number of vertices ranging from 14 to 52, for a total of 16 graphs. In every instance, the first node is chosen as both the origin and the destination.

The remaining problem characteristics are set as follows:

- The value of $D_{\max}$ is chosen so that all the mandatory nodes and an increasing fraction of the optional nodes can be visited. We set $D_{\max} = \tau_{\min}(M) + \omega(\tau_{\min}(V) - \tau_{\min}(M))$ where $\tau_{\min}(S)$ is the duration (length) of the minimum cost elementary path from origin 0 to destination $n+1$ visiting all nodes in S. As done in [Angelelli et al., 2017], we tested three values of ω: 1/4, 1/2 and 3/4.

- The percentage of mandatory customers with respect to the total number of customers in V has been set equal to two values: 0% and 25%.

- The values of prizes p_i for $i \in V$ has been generated according to four rules:

  P_1. All prizes are equal to 1.

  P_2. The prize of customer i is set to $1 + ((7141 \cdot i + 73) \mod 100)$ as done in [Fischetti et al., 1998].

  P_3. The prize of customer i is set to $1 + \lceil \frac{99 \cdot d_{0i}}{\max_{j \in V} d_{0j}} \rceil$ where d_{ij} is the distance between node i and node j.

  P_4. The prize of customer i is set to d_{0i}.

  To obtain a balanced trade-off between prizes and costs, in each instance the prizes are normalized as follows. First, they are scaled so that their sum is twice $C \cdot \tau_{\min}(V)$; second, each prize is rounded to the nearest integer.

- The probability of appearance of an optional request from a node $i \in V \backslash V_M$, i.e., θ_i, is generated in two ways:

  - Constant for all customers. We tested three different situations: low probability 0.25 (in the following F_1), medium probability 0.50 (in the following F_2), high probability 0.75 (in the following F_3).
  - Randomly generated in the interval $[0.25, 0.75]$ (in the following F_3).

Concerning the probability distributions, we assume that $\overline{T}_i = U[0,T]$ for all $i \in V \backslash V_M$, where $U[0,T]$ is the continuous uniform distribution taking values between 0 and T. In this way, the conditional probabilities $\theta_i^{k+1} = F_{i|t^{k+1}}(T)$ are easily computed as $\theta_i^{k+1} = \frac{\theta_i(T - t^{k+1})}{T - \theta_i t^{k+1}}$.

In summary, we have 96 parameter combination. For each TSPLIB instance and parameter combination, we generated 1000 random realizations. So, we tested in total 1,536,000 instances.

To guarantee a fair comparison of results, the scenarios used in the dynamic approaches have been generated once for each instance and used by all the approaches.

5.2 Computational results

In this section, we present the computational results obtained by the solution algorithms presented in Section 4. We first focus (Section 5.2.1) on the analysis of the greedy algorithms presented in Section 4.2.1 to compare the performance of the four rules we propose to determine the acceptance of a request. Then, in Section 5.2.2, we compare the myopic greedy algorithms with the `MC Greedy` ones to verify whether the introduction of the MC simulation improves the performance of the approaches. Finally, in Section 5.2.3, we show the results related to all approaches presented in Section 4.

The objective values returned by all algorithms are compared with those returned by a `Post-exact` algorithm, which provides the optimal solution of the problem using full “a posteriori” information about customer requests. More precisely, the `Post-exact` algorithm solves a POP over the set of (mandatory and optional) nodes arising requests in $(0,T)$, as this information is available at time T. The visit on mandatory nodes is imposed, while the probability of optional nodes is set to one. The value of the optimal solution of the resulting deterministic problem provides an upper bound on the value of the DPOP.

5.2.1 Myopic greedy algorithms

We first focus on the analysis of the myopic greedy algorithms presented in Section 4.2.1. The reason is that greedy algorithms are based on simple rules and are likely to be applied in practical applications where no information is available on future events or when no sophisticated solution approach is available. Thus, it is relevant to determine whether there is a dominance between them and/or whether the performance depends on problem characteristics.

First, we analyze the average solution quality over all tested instances. The behavior is illustrated in Figure 2, where the average solution value of the four greedy algorithms is illustrated and compared with the average solution value of the `Post-exact` algorithm.

The figure shows that `Profitable Look-ahead Greedy` and `Feasible Look-ahead`

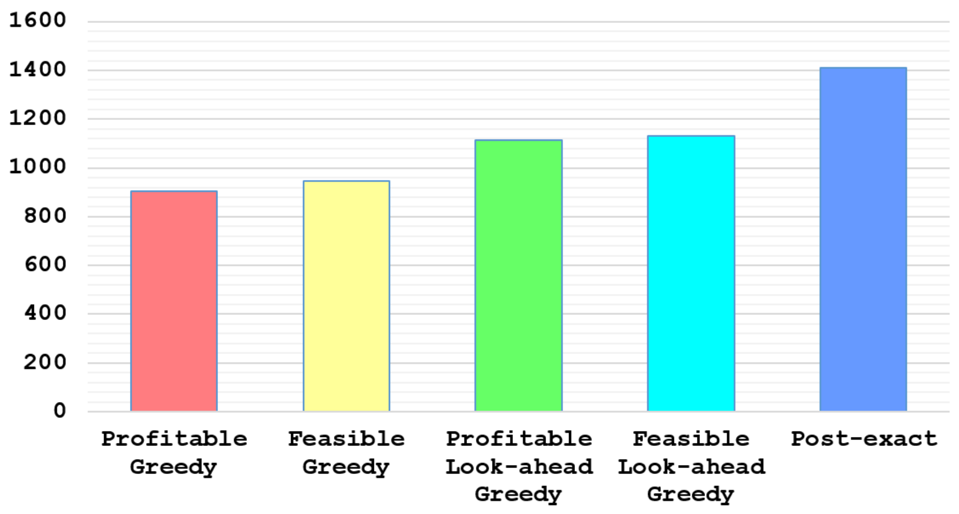

Figure 2: Myopic greedy algorithms and `Post-exact`: average solution values.

`Greedy` outperform both `Profitable Greedy` and `Feasible Greedy`, with the latter being slightly better than the former. In addition, despite being based on basic rules, their gap with respect to the bound provided by `Post-exact` is reasonable.

A more detailed analysis of the behavior of myopic greedy algorithms on the basis of problem characteristics is provided in Table I. The table reports, for each greedy algorithm, the percentage gap of the average solution value over all instances with respect to the average solution value obtained by `Post-exact`. The gap is calculated as:

$$\frac{avg(\texttt{myopic Greedy}) - avg(\texttt{Post-exact})}{avg(\texttt{Post-exact})}$$

where $avg(A)$ is the average solution value over all instances obtained by algorithm A and `myopic` refers to any one of the four algorithmic variants considered here. The results are classified by: values of ω, percentage of mandatory customers (called M from now on), rule for generating the probability of a request from an optional node and rule for generating customers' profits. The last line of the table reports the average over all instances.

The results confirm that `Feasible Look-ahead Greedy` is the best myopic greedy algorithm with an average percentage gap which is slightly below 20%, while the worst is the `Profitable Greedy` with an average percentage gap of almost 36%. The reason for the bad behavior of `Profitable Greedy` is related to the fact that it is based on a too conservative rule which prevents accepting requests that do not

	`Profitable Greedy`	`Feasible Greedy`	`Profitable Look-ahead Greedy`	`Feasible Look-ahead Greedy`
$\omega = 1/4$	33.54	45.83	29.25	23.06
$\omega = 1/2$	36.79	37.11	20.94	20.33
$\omega = 3/4$	36.40	20.56	15.64	17.11
$M = 0\%$	71.73	51.67	34.42	35.58
$M = 25\%$	12.33	20.69	12.24	9.45
F_1	34.04	28.02	22.24	21.25
F_2	21.52	50.18	19.80	19.96
F_3	34.30	28.79	23.46	23.02
F_4	40.52	35.10	19.03	16.93
P_1	34.55	29.23	18.62	18.61
P_2	27.21	36.56	15.49	13.34
P_3	41.79	32.70	25.70	24.37
P_4	41.74	32.84	25.69	24.35
All	**35.81**	**32.93**	**21.00**	**19.78**

Table I: Performance of myopic greedy algorithms with respect to `Post-exact`

seem to be convenient immediately but may turn out to be convenient when combined with future requests. This is particularly evident when looking at the results classified by value of M: when $M = 0\%$, then the gap of `Profitable Greedy` is huge, almost 72%. In fact, in this case, as no request is mandatory, the cost related to the insertion of the first customer placing a request may turn out to be very high as the route is empty at the beginning, due to the absence of mandatory nodes. This induces `Profitable Greedy` to accept very few (if any) requests. When instead $M = 25\%$, the insertion cost may be lower and this favor the acceptance of requests. In addition, in the latter case, part of the solution, the one that serves the mandatory nodes, is in common with the solution of `Post-exact`, meaning that these customers are necessarily served by both solutions, and this reduces the gap. This trend is confirmed for all myopic greedy algorithms: the gap with respect to `Post-exact` is much larger for instances where $M = 0\%$ than for instances where $M = 25\%$ for the same reason explained above. Focusing on `Feasible Greedy`, we notice that it is much more sensitive with respect to `Profitable Greedy` to the value of ω, with the gap that increases when the value of ω decreases. This is explained by the fact that when the value of ω is small, then only few requests may be feasibly accepted, thus acceptance of early arrived requests prevents from serving future, possibly more profitable, ones. We also notice that `Feasible Greedy` is very much sensitive to the rules of generating the probability of customers' requests, providing a large gap (more than 50%) when all nodes have a probability of 50%. The reason may be the same mentioned above: when all nodes have the same probability of placing a request, an early acceptance may prevent the acceptance of (probable) convenient future requests. `Profitable Look-ahead Greedy` and `Feasible Look-ahead Greedy` show a very similar behavior. They always outperform the other two heuristics, with `Feasible Look-ahead Greedy` being dominant except for the cases where $\omega = 3/4$, $M = 0\%$ and F_2. Finally, we notice a quite remarkable variance of the results on the basis of the rules used to generate the profits. In particular, classes P_3 and P_4 are the ones for which all myopic greedy algorithms provide the worst results. This may be related to the fact that, when profits are related to distances, greedy rules based on the evaluation of functions that depend on the comparison between profits and distances may face more troubles in establishing what is the best choice.

5.2.2 Myopic versus MC greedy algorithms

In this section, we analyze the benefits of introducing Monte Carlo simulation on the greedy algorithms evaluated above. The aim is to verify whether the introduction of Monte Carlo to simulate the evolution of future events helps in providing better quality solutions. Thus, we compare the performance of myopic versus MC greedy algorithms. In particular, for `MC Greedy`, we generated 100 scenarios for Step 2 of the procedure described in Section 4.2.2.

Table II shows the results. In particular, the table reports, for each `MC greedy` algorithm, the percentage gap with respect to the `Post-exact` solution and the percentage improvement with respect to the corresponding myopic greedy algorithm ('impr.'). Table II shows that the improvement due to the introduction of Monte Carlo simulation is high, especially when focusing on `Profitable Greedy` and `Feasible Greedy`. This can be safely attributed to the fact that these two heuristics are those which provide the worst results in their myopic version, and thus there is a much wider space for improvement in their case. Overall, MC provides big advantages to all greedy algorithms by remarkably smoothing the gaps with respect to `Post-exact`. In fact, focusing on `Profitable Look-ahead Greedy` and `Feasible Look-ahead Greedy`, we see that the gap is always lower than 12% except for one case ($\omega = 1/4$).

In Table III, we show the average computational time over all instances for all greedy approaches: the four myopic approaches and the four MC approaches with 100 scenarios. Times are reported in milliseconds (ms). The table shows that, even if the effort due to scenario evaluation is substantial, still computational times are absolutely acceptable. In fact, while they are at most 26ms for myopic approaches, the highest average computational time for MC approaches is 305ms for `MC Feasible Look-ahead Greedy`

	`MC Profitable Greedy`		`MC Feasible Greedy`		`MC Profitable Look-ahead Greedy`		`MC Feasible Look-ahead Greedy`	
	gap	impr.	gap	impr.	gap	impr.	gap	impr.
$\omega = 1/4$	13.34	20.19	15.04	30.79	12.48	16.77	12.11	10.95
$\omega = 1/2$	9.34	27.45	9.60	27.51	8.21	12.73	7.63	12.70
$\omega = 3/4$	5.19	31.21	7.08	13.47	4.74	10.90	4.63	12.48
$M = 0\%$	12.87	58.85	15.24	36.43	10.87	23.55	10.65	24.93
$M = 25\%$	6.06	6.27	6.59	14.09	6.05	6.19	5.62	3.83
F_1	7.94	26.10	8.53	19.50	7.34	14.90	7.14	14.11
F_2	11.93	9.59	21.00	29.18	11.57	8.23	11.70	8.26
F_3	8.27	26.03	8.89	19.90	7.66	15.80	7.51	15.51
F_4	8.93	31.58	9.49	25.62	7.82	11.21	7.18	9.75
P_1	8.18	26.37	8.96	20.27	7.34	11.28	7.09	11.52
P_2	7.74	19.47	9.11	27.45	7.11	8.38	6.76	6.58
P_3	9.71	32.08	11.13	21.58	8.82	16.88	8.42	15.95
P_4	9.68	32.06	11.16	21.68	8.81	16.89	8.41	15.94
All	**8.76**	**27.05**	**10.01**	**22.92**	**7.96**	**13.05**	**7.61**	**12.17**

Table II: MC versus myopic greedy algorithms

	Myopic	MC
`Profitable Greedy`	20	103
`Feasible Greedy`	26	110
`Profitable Look-ahead Greedy`	24	140
`Feasible Look-ahead Greedy`	21	305

Table III: Computational time of greedy algorithms (ms)

which is still extremely fast.

In Table III, we note a counter-intuitive fact: while in the myopic version `Profitable Look-ahead Greedy` is slightly slower than `Feasible Look-ahead Greedy`, in the Monte Carlo version the former algorithm is much faster than the latter. We explain this phenomenon as follows. Under the same conditions, a request accepted by `Feasible Look-ahead Greedy` is accepted also by `Profitable Look-ahead Greedy`, but not vice versa. Hence, `Profitable Look-ahead Greedy` accept more requests at the beginning of the decision time window $(0,T)$ and might reduce more quickly the time budget for future requests. The smaller time budget imply that the simulations stop earlier, as no simulation is run if a node cannot be feasibly inserted in the current path. This produces sensible time savings.

Given that `Feasible Look-ahead Greedy` provides a good compromise between solution quality and computational time and is the best greedy algorithm, in the following analysis we focus on it and do not

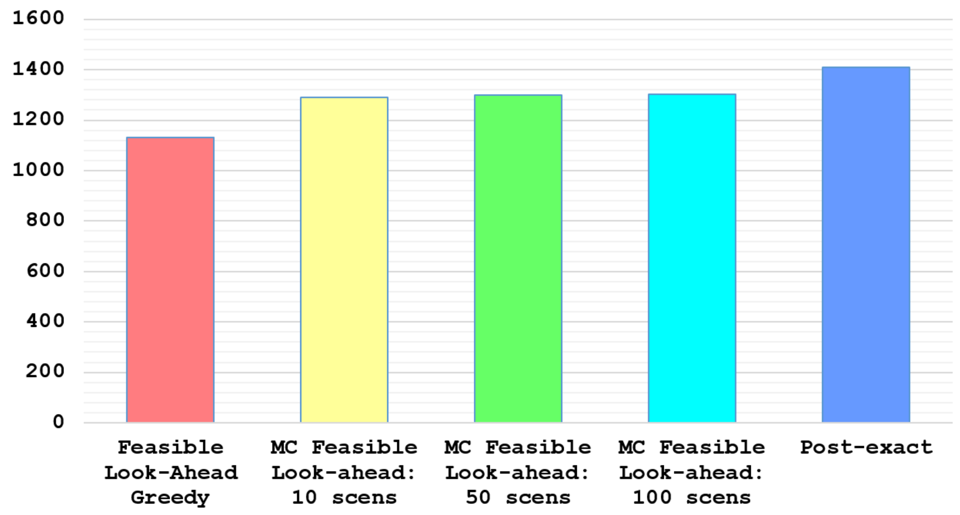

Figure 3: Comparison of `Feasible Look-ahead Greedy` and `MC Feasible Look-ahead Greedy` with different number of scenarios (`scens`).

provide further analysis related to the other greedy approaches.

One crucial decision in MC greedy approaches is the number of scenarios generated. Thus, we now analyse the performance of the `MC Feasible Look-ahead Greedy` on the basis of this parameter. In Figure 3 we report the average solution value of the `Feasible Look-ahead Greedy`, `MC Feasible Look-ahead Greedy` with 10, 50, and 100 scenarios and the `Post-exact` solution. What is interesting to notice is that the advantage to the `MC Feasible Look-ahead Greedy` with respect to `Feasible Look-ahead Greedy` is achieved already with 10 scenarios.

We remark that a similar analysis has been performed also on the other greedy algorithms and the results showed a similar trend.

5.2.3 *Comparison of static and dynamic approaches*

In this section, we analyze the performance of all approaches illustrated in Section 4. As mentioned above, from the class of greedy algorithms, we choose the `Feasible Look-ahead Greedy` as a good compromise between solution quality and computational time. Thus, in the following we illustrate the results related to the following solution approaches:

- `OP-One-Shot`.
- `OP-Multi-Shot`.

- `Feasible Look-ahead Greedy`.
- `MC Feasible Look-ahead Greedy` with 100 scenarios. Even if we noticed in the previous section that the results with 10 scenarios are similar to the ones with 100 scenarios, given the extremely low computation time of `MC Feasible Look-ahead Greedy` with 100 scenarios, we kept this setting as it is the one that provides the best results.
- `MC OP-Multi-Shot` with 10 scenarios. In this case, we choose a number of scenarios equal to 10 as `MC OP-Multi-Shot` is much more time consuming than `MC Feasible Look-ahead Greedy`. We will analyse more in depth this issue in the following.

First, we focus on the comparison between `MC OP-Multi-Shot` and `MC Feasible Look-ahead Greedy`. In particular, we analyze the average solution quality to determine which approach is performing better. In Figure 4, we show the average solution value, over all instances, of `MC OP-Multi-Shot` and `MC Feasible Look-ahead Greedy` with 10 and 100 scenarios respectively. The figure shows that the best approach among the three is `MC Feasible Look-ahead Greedy` with 100 scenarios, but the difference is very limited. In addition, `MC Feasible Look-ahead Greedy` with 10 scenarios performs better than `MC OP-Multi-Shot`, which has the same number of scenarios. This may be due to the fact that, even if `MC feasible Look-ahead Greedy` is based on a simple decision rule, the advantage of taking into account the time at which events occur overcomes the more sophisticated decision rule applied in `MC OP-Multi-Shot`.

Extending the comparison to the remaining approaches listed above, we obtain the results presented in Table IV. The table reports the gap with respect to the solution of the `Post-exact` algorithm categorized as done in the previous tables.

Table IV shows that the best approach is `MC Feasible Look-ahead Greedy` which systematically outperforms all other approaches on all instance classes. The worst approach is `OP-One-Shot`. This was expected as `OP-One-Shot` is a static approach which determines a solution at time 0 and decisions are based on this solution without taking into account updated information. In addition, `OP-One-Shot` behaves particularly bad for the class of instances F_2, where all optional nodes have a probability equal to 0.5 of placing a request. This is the class of instances that generates the worst performance also for the other approaches, and this is due to the fact that this case generates the highest error in predicting which customers will place a request. However, this disadvantage is amplified in `OP-One-Shot` due to its static nature.

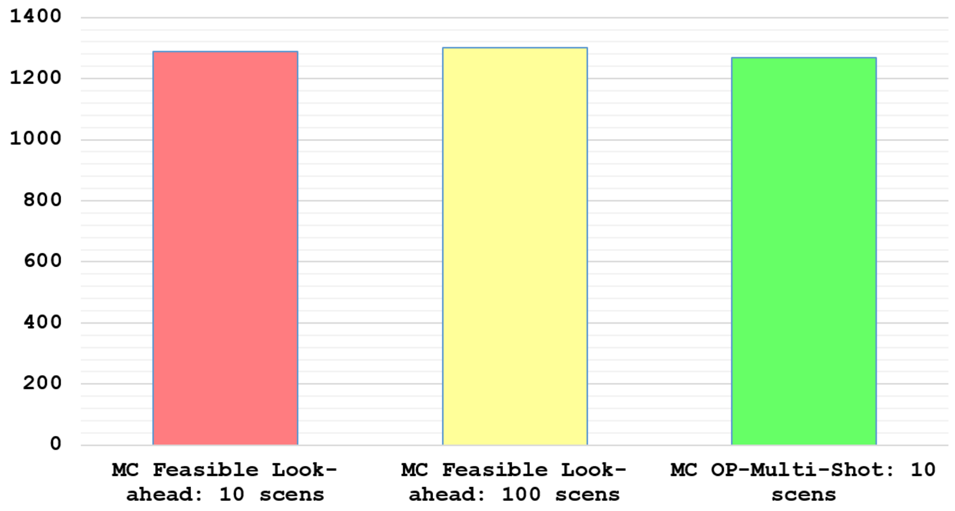

Figure 4: Comparison of `MC Feasible Look-ahead Greedy` and `MC OP-Multi-Shot`.

	`OP-One-Shot`	`OP-Multi-Shot`	`MC Feasible Look-ahead Greedy`	`MC OP-Multi-Shot`
$\omega = 1/4$	23.45	15.20	12.11	16.51
$\omega = 1/2$	21.87	14.89	7.63	10.07
$\omega = 3/4$	20.31	13.00	4.63	5.69
$M = 0\%$	32.83	25.74	10.65	13.47
$M = 25\%$	14.38	6.72	5.62	7.76
F_1	25.76	17.91	7.14	8.79
F_2	60.98	20.57	11.70	14.46
F_3	27.30	19.03	7.51	8.99
F_4	8.34	7.95	7.18	10.54
P_1	20.03	12.02	7.09	9.33
P_2	18.26	10.69	6.76	9.05
P_3	24.61	17.64	8.42	11.00
P_4	24.72	17.63	8.41	10.98
All	**21.67**	**14.24**	**7.61**	**10.02**

Table IV: Performance of static and dynamic approaches with respect to `Post-exact`

	OP-One-Shot	OP-Multi-Shot	MC Feasible Look-ahead Greedy	MC OP-Multi-Shot
$\omega = 1/4$	8	302	140	278
$\omega = 1/2$	14	503	332	680
$\omega = 3/4$	19	437	445	579
$M = 0\%$	5	291	297	663
$M = 25\%$	23	537	314	361
F_1	12	359	305	280
F_2	4	112	76	28
F_3	11	352	297	273
F_4	28	832	544	1467
P_1	16	506	321	567
P_2	14	429	305	486
P_3	13	362	298	500
P_4	13	359	298	496
All	**14**	**414**	**305**	**512**

Table V: Computational times (ms)

Concerning the comparison between `OP-Multi-Shot` and `MC OP-Multi-Shot`, we see that the latter performs better with the exception of few cases, i.e., $\omega = 1/4$, $M = 25\%$ and F_4.

Finally, in Table V we present the average computational times (in ms) of the approaches analyzed in Table IV, with the same classification of instances.

All computational times are absolutely reasonable being at most slightly above half a second. `OP-One-Shot` has a computational time of few ms. However, as it is shown in Table IV, this goes to the detriment of solution quality. Among the three other approaches, the fastest is `MC Feasible Look-ahead Greedy` which confirms to be the best both in terms of solution quality and computational time.

6 Conclusions

We have studied a dynamic and probabilistic OP, called DPOP, where decisions about acceptance/rejection of nodes must be taken one at a time, according to the probabilistic appearance of new requests. Given the set of assumptions, we have derived explicit formulas for the optimal policy according to the objective of expected profit maximization. To give practical decision methods, we have designed and implemented different heuristic techniques according to three approaches: static (i.e., using a fixed, a priori view of the future evolution), myopic (i.e., without anticipation of future events), anticipatory (i.e., based on SAA of the objective function through Monte Carlo simulation of future events). Computational tests have shown that simple anticipatory approaches are superior to myopic and static approaches.

The special feature of DPOP is the fact that acceptance/rejection decisions have to be made in real time but before the vehicle starts its route. Hence, the set of accepted nodes and the whole sequence of visits must be updated each time a new request is accepted. We believe that this feature may be relevant in several application contexts.

The present work may be extended along at least two directions. First, a richer environment may be considered, including multiple vehicles, capacity constraints, and customer time windows. This would not impact the decision framework, but it would force the adoption of more sophisticated algorithmic approaches. Second, a different management of the decision time interval $(0,T)$ could be considered. To model an online decision process, we have considered time as a continuous parameter. In many applications however, time is considered as a discrete parameter. If time is discretized in equally spaced decision epochs, multiple simultaneous service requests must be taken into account. Thus, considering time as a discrete parameter would impact both the theoretical modeling and the algorithmic approaches.

A Proof of Theorem 1

We start introducing some notations. For all $i \in V_S$ and all $t^\star \in (0,T)$, let $F_{i|t^\star}$ denote the CDF of the random variable T_i, given that no request has occurred from i in $[0,t^\star]$. For all $t \in (t^\star, T)$, we have:

$$F_{i|t^\star}(t) = \frac{\theta_i(\overline{F}_i(t) - \overline{F}_i(t^\star))}{1-\theta_i\overline{F}_i(t^\star)} \tag{7}$$

(see (5)). Moreover, we have:

$$F_{i|t^\star}(t) = \begin{cases} 0 & \text{if } t \leq t^\star, \\ \frac{\theta_i - F_i(t^\star)}{1-F_i(t^\star)} = \frac{1-\overline{F}_i(t^\star)}{1/\theta_i - \overline{F}_i(t^\star)} & \text{if } T \leq t < T+1, \\ 1 & \text{if } t \geq T+1. \end{cases}$$

Recall that $\overline{f}_i$ denotes the PDF of $\overline{T}_i$. Accordingly, let $\overline{f}_{i|t^\star}$ denote the PDF of $\overline{T}_i$, given that no request

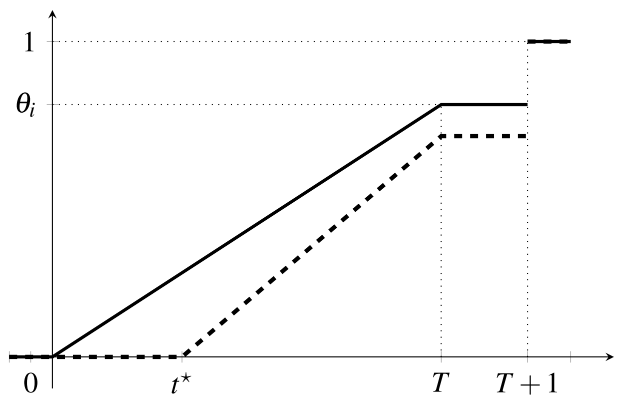

Figure 5: Example plot of F_i (solid line) and $F_{i|t^\star}$ (dashed line) in the special case where $\overline{T}_i$ is uniform.

has occurred from i in $[0,t^\star]$. By deriving (7) we have:

$$\overline{f}_{i|t^\star}(t) = \frac{\theta_i}{1-\theta_i \overline{F}_i(t^\star)} \cdot \overline{f}_i(t) \qquad \text{for all } t \in (t^\star, T) \tag{8}$$

(See Figure 5 for an illlustration.)

To prove the theorem, assume that the proper event (t^{k+1}, i^{k+1}) occurs with system at state (t^k, R^k, W^k). First, we prove that equation (3) defines the probability that the next event will be proper and will involve node $i \in V \backslash R^{k+1}$.

Let $i \in V \backslash R^{k+1}$ and let $t \in (t^{k+1}, T)$. From the independence assumption and using (7), the probability that $T_j > t$ for all $j \in V \backslash (R^{k+1} \cup \{i\})$ is

$$\prod_{j \in V \backslash (R^{k+1} \cup \{i\})} \left(1 - F_{j|t^{k+1}}(t)\right) = \prod_{j \in V \backslash (R^{k+1} \cup \{i\})} \frac{1-\theta_i \overline{F}_j(t)}{1-\theta_i \overline{F}_j(t^{k+1})}.$$

From the law of total probability, the next event will be proper and involve $i \in V \backslash R^{k+1}$ with probability

$$\int_{t^{k+1}}^{T} \left[\prod_{j \in V \backslash (R^{k+1} \cup \{i\})} \frac{1-\theta_i \overline{F}_j(t)}{1-\theta_i \overline{F}_j(t^{k+1})} \right] \overline{f}_{i|t^\star}(t) dt.$$

Thus, equation (3) follows from (8).

Second, we prove that (4) defines the probability that the next event will be the final event. The next

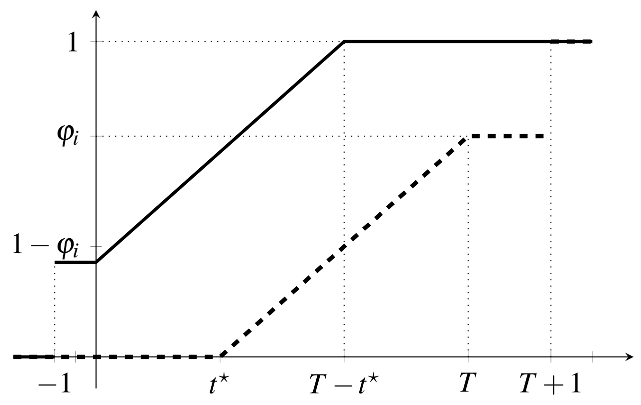

Figure 6: Example plot of $\widehat{F}_{i|T-t^\star}$ (solid line) in comparison with $F_{i|t^\star}$ (dashed line) in the special case where $\overline{T}_i$ is uniform. We use φ_i to denote $F_{i|t^\star}(T)$.

event is the final event if $t_i = T+1$ for all $i \in V \backslash R^{k+1}$. Since

$$\mathbb{P}(t_i = T+1 | t_i > t^{k+1}) = 1 - F_{j|t^{k+1}}(T),$$

we have that equation (4) follows from the independence assumption.

Finally, we prove that the probabilities computed as in (3) and (4) form indeed a probability distribution, i.e.,

$$\sum_{i \in V \backslash R^{k+1}} p_{\min}(i|t^{k+1}, R^{k+1}) + p_{\min}(\emptyset|t^{k+1}, R^{k+1}) = 1 \tag{9}$$

By using the conditioned distributions, we may write:

$$p_{\min}(i|t^{k+1}, R^{k+1}) = \int_{t^{k+1}}^{T} \left[\prod_{j \in V \backslash (R^{k+1} \cup \{i\})} \Big(1 - F_{j|t^{k+1}}(t)\Big) \right] \overline{f}_{i|t^{k+1}}(t)dt \tag{10}$$

$$p_{\min}(\emptyset|t^{k+1}, R^{k+1}) = \prod_{j \in V \backslash R^{k+1}} \Big(1 - F_{j|t^{k+1}}(T)\Big)$$

For convenience, for all $i \in V \backslash R^{k+1}$, we consider the mixed random variables $\widehat{T}_i = T - T_i$. Given a $t^\star \in (0, T)$, for all $t \in [t^\star, T]$ we define:

$$\widehat{F}_{i|T-t^\star}(T-t) = \mathbb{P}(\widehat{T}_i \leq T-t | \widehat{T}_i < T-t^\star).$$

We have:

$$\begin{aligned}
\widehat{F}_{i|T-t^\star}(T-t) &= \mathbb{P}(T-T_i \le T-t | T-T_i < T-t^\star) \\
&= \mathbb{P}(T_i \ge t | T_i > t^\star) \\
&= 1-\mathbb{P}(T_i \le t | T_i > t^\star) = 1 - F_{i|t^\star}(t)
\end{aligned}$$

Moreover, for all $t \in (t^\star, T)$,

$$\widehat{f}_{i|T-t^\star}(T-t) = \frac{d}{d(T-t)}\widehat{F}_{i|T-t^\star}(T-t) = -\frac{d}{dt}(1-F_{i|t^\star}(t)) = \overline{f}_{i|t^\star}(t)$$

(See Figure 6 for an illustration.) Thus, equation (10) can be written

$$\begin{aligned}
p_{\min}(i|t^{k+1}, R^{k+1}) &= \int_{t^{k+1}}^{T} \left[\prod_{j \in V \setminus (R^{k+1} \cup \{i\})} \left(\widehat{F}_{j|T-t^{k+1}}(T-t) \right) \right] \widehat{f}_{i|T-t^{k+1}}(T-t) dt \\
&= \int_{T}^{t^{k+1}} \left[\prod_{j \in V \setminus (R^{k+1} \cup \{i\})} \left(\widehat{F}_{j|T-t^{k+1}}(T-t) \right) \right] \widehat{f}_{i|T-t^{k+1}}(T-t) d(T-t) \\
&= \int_{0}^{T-t^{k+1}} \left[\prod_{j \in V \setminus (R^{k+1} \cup \{i\})} \left(\widehat{F}_{j|T-t^{k+1}}(x) \right) \right] \widehat{f}_{i|T-t^{k+1}}(x) dx
\end{aligned}$$

Applying integration by parts to the last expression, we obtain:

$$\begin{aligned}
p_{\min}(i|t^{k+1}, R^{k+1}) &= \left[\prod_{j \in V \setminus R^{k+1}} \widehat{F}_{j|T-t^{k+1}}(x) \right]_{0}^{T-t^{k+1}} \\
&\quad - \sum_{h \in V \setminus (R^{k+1} \cup \{i\})} \int_{0}^{T-t^{k+1}} \left[\prod_{j \in V \setminus (R^{k+1} \cup \{h\})} \left(\widehat{F}_{j|T-t^{k+1}}(x) \right) \right] \widehat{f}_{h|T-t^{k+1}}(x) dx \\
&= 1 - \prod_{j \in V \setminus R^{k+1}} \left(1 - F_{j|t^{k+1}}(T) \right) - \sum_{h \in V \setminus (R^{k+1} \cup \{i\})} p_{\min}(h|t^{k+1}, R^{k+1}) \\
&= 1 - p_{\min}(\emptyset | t^{k+1}, R^{k+1}) - \sum_{h \in V \setminus (R^{k+1} \cup \{i\})} p_{\min}(h|t^{k+1}, R^{k+1})
\end{aligned}$$

This proves equation (9) and completes the proof.

References

N. Agatz, A. Campbell, M. Fleischmann, and M. Savelsbergh. Time slot management in attended home delivery. *Transportation Science*, 45(3):435–449, 2011.

M. Allahviranloo, J.Y.J. Chow, and W.W. Recker. Selective vehicle routing problems under uncertainty without recourse. *Transportation Research Part E: Logistics and Transportation Review*, 62(0):68–88, 2014.

E. Angelelli, C. Archetti, C. Filippi, and M. Vindigni. The probabilistic orienteering problem. *Computers & Operations Research*, 81:269–281, 2017.

N. Azi, M. Gendreau, and J.-Y. Potvin. A dynamic vehicle routing problem with multiple delivery routes. *Annals of Operations Research*, 199(1):103–112, 2012.

R.W. Bent and P. Van Hentenryck. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, 52(6):977–987, 2004.

A. M. Campbell, M. Gendreau, and B.W. Thomas. The orienteering problem with stochastic travel and service times. *Annals of Operations Research*, 186(1):61–81, Jun 2011.

A.M. Campbell and M. Savelsbergh. Incentive schemes for attended home delivery services. *Transportation Science*, 40(3):327–341, 2006.

A.M. Campbell and M.W.P. Savelsbergh. Decision support for consumer direct grocery initiatives. *Transportation Science*, 39(3):313–327, 2005.

A.M. Campbell and B.W. Thomas. Probabilistic traveling salesman problem with deadlines. *Transportation Science*, 42(1):1–21, 2008.

I-M. Chao, B.L. Golden, and E.A. Wasil. A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88(3):475–489, 1996.

L. Evers, K. Glorie, S. van der Ster, A.I. Barros, and H. Monsuur. A two-stage approach to the orienteering problem with stochastic weights. *Computers & Operations Research*, 43:248–260, 2014.

M. Fischetti, J.J. Salazar González, and P. Toth. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10(2):133–148, 1998.

T. Flatberg, G. Hasle, O. Kloster, E.J. Nilssen, and A. Riise. Dynamic and stochastic vehicle routing in practice. In V. Zeimpekis, C.D. Tarantilis, G.M. Giaglis, and I. Minis, editors, *Dynamic Fleet Management: Concepts, Systems, Algorithms & Case Studies*, pages 41–63. Springer US, Boston, MA, 2007.

G. Ghiani, E. Manni, A. Quaranta, and C. Triki. Anticipatory algorithms for same-day courier dispatching. *Transportation Research Part E: Logistics and Transportation Review*, 45(1):96–106, 2009.

G. Ghiani, E. Manni, and B. W. Thomas. A comparison of anticipatory algorithms for the dynamic and stochastic traveling salesman problem. *Transportation Science*, 46(3):374–387, 2012.

A. Gunawan, H.C. Lau, and P. Vansteenwegen. Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2):315–332, 2016.

L.M. Hvattum, A. Løkketangen, and G. Laporte. Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic. *Transportation Science*, 40(4):421–438, 2006.

T. İlhan, S.M.R. Iravani, and M.S. Daskin. The orienteering problem with stochastic profits. *IIE Transactions*, 40(4):406–421, 2008.

V. Pillac, M. Gendreau, C. Guéret, and A.L. Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013.

H.N. Psaraftis, M. Wen, and C.A. Kontovas. Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):3–31, 2016.

U. Ritzinger, J. Puchinger, and R.F. Hartl. A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research*, 54(1):215–231, 2016.

I. Sungur, Y. Ren, F. Ordó nez, M. Dessouky, and H. Zhong. A model and algorithm for the courier delivery problem with uncertainty. *Transportation Science*, 44(2):193–205, 2010.

M.W. Ulmer and B.W. Thomas. Meso-parametric value function approximation for dynamic customer acceptances in delivery routing. *European Journal of Operational Research*, 2019. doi: https://doi.org/10.1016/j.ejor.2019.04.029. URL `http://www.sciencedirect.com/science/article/pii/S0377221719303637`.

M.W. Ulmer, D.C. Mattfeld, and F. Köster. Budgeting time for dynamic vehicle routing with stochastic customer requests. *Transportation Science*, 52(1):20–37, 2018.

S. Zhang, J.W. Ohlmann, and B.W. Thomas. A priori orienteering with time windows and stochastic wait times at customers. *European Journal of Operational Research*, 239(1):70–79, 2014.

S. Zhang, J.W. Ohlmann, and B.W. Thomas. Dynamic orienteering on a network of queues. *Transportation Science*, 52(3):691–706, 2018.