EURO Journal on Transportation and Logistics 12 (2023) 100112

Contents lists available at ScienceDirect

EURO Journal on Transportation and Logistics

journal homepage: www.sciencedirect.com/journal/euro-journal-on-transportation-and-logistics

The data-driven time-dependent orienteering problem with soft time windows

Edison Avraham, Tal Raviv *

Department of Industrial Engineering, Tel Aviv University, Tel Aviv, Israel

ARTICLE INFO

Keywords:
Time-dependent
Orienteering
Transportation
Vehicle routing

ABSTRACT

In this paper, we study an extension of the orienteering problem where travel times are random and time-dependent and service times are random. Additionally, the service at each selected customer is subject to a soft time window; that is, violation of the window is allowed but subject to a penalty that increases in the delay. A solution is a tour determined before the vehicle departs from the depot. The objective is to maximize the sum of the collected prizes net of the expected penalty. The randomness of the travel and service times is modeled by a set of scenarios based on historical data that can be collected from public geographical information services. We present an exact solution method for the problem based on a branch-and-bound algorithm enhanced by a local search procedure at the nodes. A numerical experiment demonstrates the merits of the proposed solution approach. This study is the first to consider an orienteering problem with stochastic travel times and soft time windows, which are more relevant than hard time windows in stochastic settings.

1. Introduction

When solving routing and scheduling problems, the planner often does not have sufficient resources or does not wish to serve all the customers. Instead, the planner can prioritize customers by assigning a value ("prize") to each one of them. The orienteering problem is to find a tour that visits a subset of the customers and satisfies some operational constraints with the objective of maximizing the sum of the values of the visited customers.

Orienteering problems arise in settings such as field service operations, where some tasks are of high priority (e.g., customers who need their equipment to be repaired) while others are of relatively low priority (e.g., customers who need to be visited by a technician for regular maintenance operation). With the increasing traffic congestion in many urban areas worldwide, the travel times of mobile personnel have become highly dependent on the time of the journey and the specific itinerary. Moreover, in a congested network, the variability of travel time is high since it is sensitive to various events, such as road accidents and weather conditions. Therefore, there is an increasing need to model and solve routing problems that consider time dependency and stochasticity in the context of travel times.

In this paper, we introduce and study the data-driven time-dependent orienteering problem with soft time windows (DD-TD-OP-STW). This is an extension of the orienteering problem where the selected customers are visited subject to soft time windows. That is, each customer is characterized by a time window. If the vehicle arrives at the customer before the beginning of the time window, it waits at the customer location to start the service when the window opens. If the vehicle arrives at the customer after the end of the time window, it serves the customer and incurs a penalty that increases with the extent of the delay. The travel times are random, and their distribution is determined by the departure time from each customer. In practice, the travel time between a pair of locations when departing at a given time is correlated with the travel time between other nearby origins and destinations at the same departure time as well as with these travel times when departing slightly earlier or later. These interdependencies may significantly affect the optimal route and the subset of selected customers. The objective is to determine a route a priori that maximizes the sum of the collected prizes net of the expected delay penalties due to time window violations.

We note that when travel and service times are stochastic, a route that is certain to respect the time windows of all customers can be highly inefficient. Soft time windows facilitate a way to model the tradeoff between the low probability of being late to the customer and the efficiency merits of a tight schedule.

An effective approach for modeling the stochastic nature of travel

* Corresponding author.
E-mail address: talraviv@tauex.tau.ac.il (T. Raviv).

https://doi.org/10.1016/j.ejtl.2023.100112
Received 20 June 2022; Received in revised form 23 May 2023; Accepted 7 June 2023
Available online 9 June 2023

times can consider a set of representative scenarios and evaluate the average value of the solutions over all cases. Such an approach captures the intricate interdependencies discussed above. The scenarios may be based on historical information collected by geographical information services.

The rest of this paper is organized as follows. In Section 2, we review the relevant literature and identify the gap addressed by this study. In Section 3, we provide a formal definition and a mathematical model of the DD-TD-OP-STW. In Section 4, we present a branch-and-bound (B&B) algorithm to solve the problem and extend it by integrating a local search (LS) heuristic into the algorithm. In Section 5, we present the results of a numerical experiment and demonstrate the merits of the B&B algorithm and, in particular, of its hybrid version. In Section 6, we draw conclusions and suggest directions for future research.

2. Literature review

Orienteering problems have been the focus of a variety of studies in previous decades. For a comprehensive review of many of the previous studies, we refer the reader to Vansteenwegen et al. (2011) and Gunawan et al. (2016). Here, we limit our discussion to some specific variants of the problem: Section 2.1 discusses time-dependent orienteering problems (TD-OPs); Section 2.2 discusses stochastic orienteering problems (S-OPs); Section 2.3 discusses stochastic and time-dependent orienteering problems (S-TD-OPs); and Section 2.4 identifies a gap in the literature and states the contributions of this study.

2.1. Time-dependent orienteering problems

In time-dependent vehicle routing problems, the travel time from a given location i to a given location j depends on the time at which the vehicle departs from location i. Travel time is commonly modeled in a way that satisfies the FIFO property; that is, Vehicle A that departs from location i later than vehicle B cannot arrive at j earlier than Vehicle B.

To the best of our knowledge, Fomin and Lingas (2002) were the first to consider the TD-OP and presented an approximation algorithm for it. Verbeeck et al. (2014) studied a similar problem and formulated it as a mixed-integer program (MIP). They devised an ant colony system (ACS) algorithm to solve the problem and demonstrated its effectiveness by a numerical experiment. Gunawan et al. (2014) presented a different integer linear program (ILP) formulation and several local search (LS)-based heuristics to solve the TD-OP problem. They conducted a numerical experiment that illustrated the merits of the algorithms when compared with a solution obtained by solving their ILP using CPLEX. Mei et al. (2016) studied a multiobjective variant of the TD-OP where each customer is associated with several types of prizes and the objectives are to maximize the sum of each one of these types. They devised several heuristic algorithms for the problem.

Garcia et al. (2010) studied the time-dependent orienteering problem with hard time windows (TD-OP-TW) and presented an iterated local search (ILS) algorithm for its solution. Abbaspour and Samadzadegan (2011) devised a genetic algorithm to solve a variant of the TD-OP-TW. Verbeeck et al. (2017) devised an ACS algorithm for the TD-OP-TW. Khodadadian et al. (2022) studied a variant of the TD-OP-TW and solved it using a variable neighborhood search (VNS) heuristic.

Peng et al. (2019) solved the TD-OP-TW where both travel times and profits are time dependent. The problem is formulated as an MIP. A dynamic programming (DP) approach is incorporated into an ILS algorithm for the solution of the problem. They showed that their method is competitive with the solution obtained from CPLEX and outperforms several heuristic algorithms. Peng et al. (2020) devised an exact DP algorithm for the TD-OP-TW.

Li (2012) studied the time-dependent team orienteering problem (TD-TOP), where multiple routes are to be determined. They formulated the problem as an MIP and solved it using a DP algorithm. Garcia et al. (2013) studied the time-dependent team orienteering problem with hard time windows (TD-TOP-TW). They devised ILS-based heuristics for the solution of the problem and numerically showed that their approach performs well. Gavalas et al. (2014) and Gavalas et al. (2015) devised two new cluster-based heuristic algorithms to solve a variant of the TD-TOP-TW.

2.2. Stochastic orienteering problems

The stochasticity of the travel and service times in orienteering problems can be addressed either by applying a static (off-line) solution, where the route is determined a priori before the route is executed, or by a dynamic (on-line) policy, in which the route is determined during the execution as information is gradually revealed.

To the best of our knowledge, Teng et al. (2004) were the first to consider an orienteering problem with stochastic travel and service times (S-OP). They formulated the problem as a two-stage stochastic program, and an integer L-shaped solution method was devised. Tang and Miller-Hooks (2005) considered an orienteering problem where the service time at each customer is an independent random variable drawn from a discrete distribution and the travel times and prizes are deterministic. They devised an exact branch-and-cut (B&C) algorithm as well as a heuristic algorithm.

Ilhan et al. (2008) studied the orienteering problem with stochastic profits; that is, each location is characterized by a normally distributed random profit. The objective is to maximize the probability of collecting more than a prespecified target profit level. They presented an exact algorithm as well as a genetic heuristic to solve the problem.

Campbell et al. (2011) studied a variant of the S-OP where the customers are to be visited before a predefined deadline. The planner decides in advance on the subset of customers to be visited and on the route (a sequence of customers). For each of the visited customers, arriving earlier than the deadline results in a prize, while arriving later than the deadline implies a penalty. They suggested a DP method for solving some special cases of the problem, whereas a VNS heuristic was presented for the general case. Papapanagiotou et al. (2014) and Papapanagiotou et al. (2015) studied a problem similar to that of Campbell et al. (2011) and devised several Monte Carlo sampling-based methods for the evaluation of the objective function. Dolinskaya et al. (2018) extended the work of Campbell et al. (2011) by adding the possibility of dynamically determining the path between each pair of consecutive customers as the travel times in the road network are realized. Their solution method incorporates DP elements for the determination of paths with a VNS algorithm inspired by Campbell et al. (2011). Panadero and Juan (2020) studied the stochastic team orienteering (S-TOP) problem and solved it using a VNS framework that applies simulation techniques.

Evers et al. (2014) studied an orienteering problem with stochastic parameters that can represent travel time, service time or customer demand. They formulated the problem as a two-stage stochastic program and devised two solution approaches, namely, sample average approximation (SAA) and a local search heuristic.

Gupta et al. (2015) studied an S-OP with stochastic service times and deterministic travel times. They considered static and dynamic versions of the problem and presented approximation algorithms for the two versions. Bian and Liu (2018) studied a dynamic S-OP and suggested a multiple plan approach (MPA) strategy.

Zhang et al. (2014) solved an orienteering problem with time windows and stochastic service times where the agent can skip a customer after arriving at his location (and observing the service time). The route is determined in advance, but the skipping decisions are made dynamically. Additionally, they devised a VNS heuristic inspired by Campbell et al. (2011). Zhang et al. (2018) extended Zhang et al. (2014) by allowing fully dynamic routing decisions and presented an approximate dynamic programming algorithm (ADP) for this problem.

Angelelli et al. (2017) solved a variant of the static orienteering

problem with stochastic customers, where the route is determined a priori, but each customer may not be available with some known probability. They formulated the problem as a stochastic MIP and presented a B&C as well as heuristic algorithms for the problem. Chou et al. (2021) solved a similar problem by means of a tabu search (TS) heuristic and used Monte Carlo sampling to evaluate the solutions. Angelelli et al. (2021) solved a dynamic version of the problem in which some service requests are mandatory and known in advance and some service requests may arrive over time and can be accepted or rejected. All the mandatory and accepted requests are served on the following day.

Song et al. (2020) study a two-stage stochastic team orienteering problem with time windows (S-TOP-TW). In the first stage, a mandatory set of customers is assigned to vehicles. In the second stage, a new set of optional customers is revealed, and each such customer may be added to an existing route but without reallocating mandatory customers to vehicles. They formulated the problem as a two-stage stochastic integer program and solved the first stage using a multiple scenario approach (MSA) and the second stage using a branch-and-price approach. Karunakaran et al. (2019) studied a version of the S-TOP-TW where routes are dynamically determined and presented a hyper heuristic approach to solve the problem.

2.3. Stochastic and time-dependent orienteering problems

Lau et al. (2012) solved a variant of the orienteering problem where travel times are both stochastic and time-dependent (S-TD-OP) using a local search algorithm. They presented two approaches to calculate arrival time distributions – a sampling approach and a matrix-based approach. Varakantham and Kumar (2013) also studied the S-TD-OP and formulated it as a stochastic optimization program and as a deterministic program using SAA. The deterministic program can be solved by commercial solvers such as CPLEX. In a numerical experiment, they showed that the proposed method outperformed the LS algorithm of Lau et al. (2012). Varakantham et al. (2018) studied the S-TD-OP and presented two deterministic formulations of the problem. The first relied upon the SAA technique, and the second relied on a heuristic approximation of the parameters of the model. They solved the problem using an LS heuristic and CPLEX.

Verbeeck et al. (2016) studied the stochastic time-dependent orienteering problem with time windows (S-TD-OP-TW). The travel times were assumed to follow a normal distribution, while service times were assumed to be deterministic. The lower bounds of the hard time windows required the authors to devise an estimation algorithm to calculate the arrival times. The problem was solved using an ant colony algorithm. Numerical experiments showed the merit of considering time dependency and stochasticity compared to a deterministic time-independent case. Liao and Zheng (2018) studied a similar problem and devised a heuristic solution approach for it.

Çelik (2021) studied the team orienteering problem with stochastic and time-dependent travel times (S-TD-TOP). The problem was formulated as a two-stage stochastic MIP, and an L-shaped algorithm was devised for its solution.

2.4. Summary

In this section, we summarize the three streams of literature on the stochastic and time-dependent orienteering problem reviewed earlier.

Type of orienteering problem	
Time dependent	Fomin and Lingas (2002), Verbeeck et al. (2014) Gunawan et al. (2014), Mei et al. (2016), Garcia et al. (2010), Abbaspour and Samadzadegan (2011), Verbeeck et al. (2017), Khodadadian et al. (2022), Peng et al. (2019), Peng et al. (2020), Li (2012), Garcia et al. (2013), Gavalas et al. (2014), Gavalas et al. (2015)
Stochastic	Teng et al. (2004), Tang and Miller-Hooks (2005), Ilhan et al. (2008), Campbell et al. (2011), Papapanagiotou et al. (2014), Papapanagiotou et al. (2015), Dolinskaya et al. (2018), Panadero and Juan (2020), Evers et al. (2014), Gupta et al. (2015), Bian and Liu (2018), Zhang et al. (2014)). Zhang et al. (2018), Angelelli et al. (2017), Chou et al. (2021), Angelelli et al. (2021), Song et al. (2020), Karunakaran et al. (2019)
Stochastic & time dependent	Lau et al. (2012), Varakantham and Kumar (2013), Varakantham et al. (2018), Verbeeck et al. (2016), Liao and Zheng (2018), Çelik (2021)

2.5. Contribution of this study

This study is the first to consider an orienteering problem with stochastic travel times and soft time windows. We note that in stochastic settings, time windows with soft upper bounds may be more practical than those with hard upper bounds since it can be very restrictive to avoid late service under any possible scenario.

Previous studies on stochastic and time dependent orienteering problems overlooked the interdependencies between the travel times of spatially or temporally close journeys. In this study, we model the stochasticity of the travel times and service times using a set of scenarios based on historical data. Such a modeling approach enables capturing the intricate interdependencies that appears naturally in practice.

We present an exact solution method based on hybridization of a B&B and LS algorithms. A numerical experiment demonstrates the effectiveness of this method and the advantage of applying LS at some of the B&B nodes. Notably, while this approach is novel for the orienteering literature, it is used by previous authors for scheduling and traveling salesperson problems; see, for example, Jiang et al. (2014).

3. Problem definition

The data-driven time-dependent orienteering problem with soft time windows (DD-TD-OP-STW) is stated as follows: A set of N customers, each bearing a prize, may or may not be served during a single working day by a single vehicle that departs and returns to a depot. The travel times among the locations and the service times at each customer are uncertain. Moreover, the travel time is time dependent, i.e., affected by the departure time. The uncertainty is modeled using a set of scenarios based on historically collected data. Each scenario specifies the service time at each customer and the travel time between each pair of locations in each time interval of the planning horizon. The planner assumes that each of these scenarios can occur with equal probability. Each customer has a time window for service beginning. If the vehicle arrives at a customer before the beginning of the time window, it waits until the window opens; if it arrives at the customer after the end of the time window, a penalty is incurred. The penalty is nondecreasing in the extent of the lateness.

The goal is to select a subset of the customers and a sequence to visit them to maximize the total sum of prizes collected at the visited customers net of the expected penalty due to late arrivals. The vehicle returns to the depot at the end of the tour, but the return time is not considered since there is no time window associated with the depot. However, the trip from the last-served customer to the depot may or may not be implemented in practice. This model is suitable for many organizations in the field service industry, where the length of the working day is implicitly defined by the closing time of the latest time window.

Next, we formulate the problem as a nonlinear mathematical model.

3.1. Mathematical model

3.1.1. Parameters

$\{1, \ldots n\}$	Indices of the customers
0	Index of the depot. We refer to $\{0, \ldots, n\}$ as the set of *locations*
$[a_i, b_i]$	Time window for the beginning of service for customer i
K	Number of scenarios
s_{ik}	The service time of customer i in scenario k
p_i	The prize that can be collected if customer i is visited
$t_{i,j,k}(\tau)$	Travel time between location i and location j at departure time τ for the k^{th} scenario. We assume that the travel times in each scenario follow the FIFO property. Moreover, we assume that the triangular inequality holds in the time-dependent setting. That is, $t_{i,j,k}(\tau) \leq t_{i,l,k}(\tau) + t_{l,j,k}(\tau + t_{i,l,k}(\tau)) \forall i,j,l = 0, \ldots, n, k = 1, \ldots, K, \tau$.
$G_i(o)$	Penalty function for late start service at the customer's location i. The exact shape of the penalty function is an input of this model and should be determined by the service level agreement between the provider and the customers. We assume that $G_i(o)$ is a nondecreasing positive function of o and $G_i(0) \equiv 0$.

3.1.2. Decision variables

x_{ij}	Binary variable that equals 1 if customer j is visited immediately after customer i
u_{ik}	The start of service time at location i in realization k, where $u_{0k} \equiv 0$. If customer i is not visited the value of u_{ik} is zero.
o_{ik}	Lateness at customer i in realization k

$$maximize \sum_{i=1}^{n} p_i \sum_{j=0}^{n} x_{ij} - \frac{1}{K} \sum_{i=1}^{n} \sum_{k=1}^{K} G_i(o_{ik}) \quad (1)$$

subject to

$$\sum_{j=0}^{n} x_{ij} = \sum_{j=0}^{n} x_{ji} \forall i = 0, \ldots, n \quad (2)$$

$$\sum_{j=0}^{n} x_{ij} \leq 1 \forall i = 0, \ldots, n \quad (3)$$

$$u_{jk} \geq \left(u_{ik} + s_{ik} + t_{i,j,k}(u_{ik} + s_{ik})\right) x_{ij} \forall i = 0, \ldots, n; j = 1, \ldots, n, k = 1 \ldots .K \quad (4)$$

$$a_i \sum_{j=0}^{n} x_{ij} \leq u_{ik} \forall i = 1, \ldots, n, k = 1 \ldots .K \quad (5)$$

$$b_i \sum_{j=0}^{n} x_{ij} + o_{ik} \geq u_{ik} \forall i = 1, \ldots, n, k = 1 \ldots .K \quad (6)$$

$$u_{0k} = 0 \; k = 1, \ldots K \quad (7)$$

$$x_{ij} \in \{0, 1\} \forall i, j = 0, \ldots, n \quad (8)$$

$$o_{ik} \geq 0 \forall i = 0, \ldots, n; \; k = 1, \ldots K \quad (9)$$

The model maximizes the overall collected prizes net of the expected penalty costs, that is, the expected net profit. The first term of the objective function is the collected prizes, which are not affected by the scenarios. The second term is the expected penalty cost calculated based on the assumption that each of the K scenarios has the same probability of occurrence. This assumption is reasonable when the scenarios are constructed from actual traffic data over a sample of several days. Constraint (2) maintains vehicle flow conservation. Constraint (3) stipulates that each location is visited at most once. Constraint (4) relates the start service times to the routing variables. Together with (2), it also eliminates subtours and stipulates that the resulting tour starts at the depot (node 0). Constraint (5) enforces hard lower bounds on the start service times of customers, while constraint (6) relates the lateness variables to the start service times. Note that for unvisited customers, the model sets $u_{ik} = 0$ and due to the optimization, $o_{ik} = 0$. Constraint (7) initializes the departure time from the depot, and Constraints (8) and (9) define the domains of the decision variables.

In the appendix, we present a linearized version of the mathematical model (1)–(9). We note that this model can be solved for only tiny instances using a commercial solver; this is the motivation for the solution methods presented in Section 4.

Note that any optimal solution to the DD-TD-OP-STW problem cannot contain customers with a negative expected net profit. If such a customer exists in the solution, removing her from the tour strongly increases the expected net profit related to her (to zero) and cannot increase the penalty of all the following customers. Indeed, due to the triangle inequality, the start service time at all these customers is not postponed.

4. Methodology

In this section, our approaches to solving the DD-TD-OP-STW are presented. First, we discuss an exact B&B algorithm that can solve instances of up to 30 customers. Then, we present an approach that hybridizes the B&B and a local search heuristic. The hybrid approach can produce near-optimal solutions in a short time. The concepts presented in this section are demonstrated graphically in the electronic appendix of this paper.

4.1. B&B algorithm for the DD-TD-OP-STW

B&B algorithms are commonly used to solve combinatorial optimization problems, in particular, single vehicle routing problems (see Laporte and Martello (1990) and Ramesh et al. (1992) for orienteering problems). Our exact B&B algorithm is described in this section.

Pseudocode of the algorithm is presented in Fig. 1.

In lines 1–4, we define the properties of the root node of the B&B tree. Each node contains a sequence of the selected customers in the node (*Sequence*), a set of customers that may (or may not) be visited later in the tour (*Eligible*), and the upper bound of the node (*UB*). Customers that are not included in both *Sequence* and *Eligible* will not be visited in any of the descendants of the current node. In the list L we store the open nodes of the B&B tree.

At the root node, *Sequence* is initialized as a list containing the depot only and the set *Eligible* as the list of customers. In line 2, *Eligible* is refined using the function *CalcPotentialSuccessors* (described below), which eliminates the customers who are not worth visiting given the current values of *Sequence* and *Eligible*. The upper bound for the root node, which is the initial global upper bound, is calculated using the function *CalcUB* (see detailed description below).

Next, in line 5, an initial lower bound (*GlobalLB*), as well as an incumbent solution (*Incumbent*), are obtained using the function *CalcLB* that applies a heuristic method to obtain a feasible solution that extends the current sequence. Later in this section, we describe several implementations of this function.

In each iteration of the main loop (lines 6–24), one node from list L is removed and stored as $(Sequence, Eligible, UB)$ (line 7). If the upper bound of the current node is greater than the global lower bound, the node is processed (line 8); otherwise, it is discarded. In line 9, the algorithm constructs a set of customers (*PotentialImmediateSuccessors*) that may immediately follow the last customer in the current tour (*Sequence*) in any optimal solution. This is carried out by the function *CalcPotentialImmediateSuccessors* (see below). The algorithm branches on the customers in this set only.

In the inner loop (lines 10–22), the algorithm creates the new potential nodes to be inserted into the tree (L). Each such node consists of a sequence that extends the current sequence by one of the customers in *PotentialImmediateSuccessors* (line 11). The new sequence is stored in *NewSequnce*. In line 12, an updated set of potential successors related to

```
1  Sequence ← [0],    Eligible = {1, ..., n};
2  Eligible = CalcPotentialSuccessors(Sequence, Eligible);
3  GlobalUB = CalcUB(Sequence, Eligible);
4  L ← [(Sequence, Eligible, GlobalUB)];
5  GlobalLB, Incumbent = CalcLB(Sequence, Eligible);
6  while L ≠ ∅ do
7      Remove a node from L and store as (Sequence, Eligible, UB);
8      if UB > GlobalLB then
9          PotentialImmediateSuccessors = CalcPotentialImmediateSuccessors(Sequence, Eligible);
10         for i ∈ PotentialImmediateSuccessors do
11             NewSequnce = [Sequence, i];
12             NewEligible = CalcPotentialSuccessors(NewSequnce, Eligible \ {i});
13             NewLB, CandSolution = CalcLB(NewSequnce, NewEligible);
14             NewUB = CalcUB(NewSequnce, NewEligible);
15             if NewUB > GlobalLB then
16                 L.append((NewSequnce,NewEligible,NewUB));
17             end
18             if NewLB > GlobalLB then
19                 GlobalLB = NewLB;
20                 Incumbent = CandSolution;
21             end
22         end
23     end
24 end
```

Fig. 1. Pseudocode of the B&B algorithm.

the new sequence (*NewEligible*) is calculated using the function *CalcPotentialSuccessors*. In line 13, the algorithm applies the function *CalcLB* to obtain a heuristic solution that extends *NewSequnce* and returns its value as a lower bound. In line 14, a new upper bound for the node is calculated using the function *CalcUB*.

If the upper bound of the new sequence is greater than the global lower bound (line 15), we create a new entry in L (line 16). The new entry contains the newly created sequence, new set of eligible customers, and the upper bound. If the lower bound of the sequence is greater than the best known lower bound (Line 18), the global lower bound and the incumbent solution are updated as their respective values for the node (lines 19–20). The process ends with an optimal incumbent solution when the list L is empty, i.e., all the nodes of the B&B tree have been explored or discarded. However, other stopping criteria may apply to obtain near-optimal solutions.

L is implemented as a priority queue with the upper bound of each entry as its key. That is, in each iteration, the node with the largest upper bound is processed. Next, we describe the functions used by the algorithms in detail.

All the functions obtain two arguments: *Sequence* and *Eligible*. The value of *Sequence* is an ordered set that describes the subroute already constructed in the current node. *Eligible* is a subset of customers not included in *Sequence* that includes all the customers not yet eliminated from the solution in the node. With respect to a given *Sequence*, we denote the last customer by *i*. The service completion time of customer *i* in scenario *k* is denoted by $t_k^{'}$.

4.1.1. CalcpotentialSuccessors (sequence, eligible)

The function returns a subset of the set *Eligible* consisting of customers who may be visited in an optimal solution after visiting the customers in *Sequence*. A customer is potentially profitable if their prize is greater than a lower bound on the expected delay penalty. Recall that to calculate the expected profit, we need to evaluate all the scenarios considering the time windows and the time-dependent travel times. Specifically, the function returns the set

$$\{j \in Eligible |\ p_j > \frac{1}{K}\ \sum_{k=1}^{K} G_j\left(\max\left(0, t_k^{'} + t_{i,j,k}\left(t_k^{'}\right) - b_j\right)\right)$$

The travel times satisfy the triangular inequality, and the delay penalties are nondecreasing in the start service time. Therefore, if it is not profitable to visit a customer immediately after customer *i*, it is also not profitable to do so at any later leg of the tour. Hence, if a customer is not in the eligible set of a particular node in the B&B tree, it is also not in the eligible set of any of its descendent nodes. Thus, when a new node is created, the *eligible* set of its parent is refined using the *CalcPotentialSuccessors* function.

4.1.2. CalcLB(Sequence, eligible)

The function returns a solution that starts with *Sequence* and may contain some of the customers from *Eligible* along with the value of the solution. This value constitutes a lower bound for solutions that consist of *Sequence* possibly followed by some of the customers in *Eligible*.

The function extends *Sequence* by greedily adding customers from *Eligible*. In each iteration, the customer with the highest expected net profit is added at the end of the current sequence. The process continues until it is not possible to add a customer with positive expected net profit. The expected net profit of a customer is calculated as the prize net of the average penalty over all the scenarios. For this purpose, the function maintains the start service time at each customer in each scenario.

4.1.3. CalcUB(Sequence, eligible)

The upper bound of a node given *Sequnce* and *Eligible* is obtained as the sum of the total prizes net of penalties collected when visiting the customers in *Sequnce* first and an upper bound on the net profit that can be collected from the customers in *Eligible*. The latter term is obtained as the solution of a maximum weight matching problem in a bipartite graph (U, V, E) where $U = \{i\} \cup Eligible$ (recall that i is the last customer in *Sequence*), and $V = Eligible$. The vertices in U represent possible customer origins in the future route of the vehicle, and the vertices of V represent possible customer destinations.

To define the edges set, E, we first introduce the parameter $\delta_{j_1 j_2 k}$ that represents an upper bound on the net profit that can be gained by visiting customer $j_2 \in V$ after customer $j_1 \in U$ in scenario k in a solution that extends *Sequence*.

For $j_1 = i$, we define $\delta_{i j_2 k} = p_{j_2} - G_{j_2}(max\{t_k^{'} + t_{ij_2k}(t_k^{'}) - b_{j_2}, 0\})$.

For $j_1 \neq i$, we define $\delta_{j_1 j_2 k} = p_{j_2} - G_{j_2}(max\{max(a_{j_1}, t_k^{'} + t_{i j_1 k}(t_k^{'})) + s_{j_1 k} + t_{j_1 j_2 k}(max(a_{j_1}, t_k^{'} + t_{i j_1 k}(t_k^{'})) + s_{j_1 k}) - b_{j_2}, 0\})$.

Note that due to the triangle inequality and the FIFO property, the arguments of the penalty function $G_{j_2}(\cdot)$ in the calculation of $\delta_{j_1 j_2 k}$ are lower bounds on the start service time at customer j_2 (if preceded by j_1). Furthermore, since the penalty, $G_{j_2}(\cdot)$, is a nondecreasing function of the start service time at the customer, $\delta_{j_1 j_2 k}$ is an upper bound on the net profit from j_2 when preceded by j_1.

The set of edges in (U, V, E) is obtained as follows:

$$E = \left\{ (j_1, j_2) \middle| j_1 \in U, j \in V, j_1 \neq j_2, \sum_{k=1}^{K} \delta_{j_1 j_2 k} > 0 \right\}$$

This set defines all the ordered pairs of remaining customers (j_1, j_2) for which it is still possible to obtain a positive expected profit by visiting j_2 later than j_1.

The weight of each edge (j_1, j_2) in (U, V, E) is

$$\frac{1}{K} \sum_{k=1}^{K} \delta_{j_1 j_2 k},$$

which equals the upper bound on the expected net profit from visiting j_2.

The upper bound on the expected net profit that can be collected from the remaining customers equals the value of the maximum weight matching of (U, V, E). Indeed, each visited customer from *Eligible* $= V$ must be preceded by one of the customers from U. Any subroute starting at customer i that visits some of the remaining customers in *Eligble* can be mapped to a solution of the matching problem with a value equal to or greater than the net profit that can be obtained from visiting the subroute. Finally, the function returns the value of the maximum weight matching problem plus the expected profit from *Sequence* as an upper bound on the maximal value of solutions that begins with *Sequence*.

4.1.4. CalcpotentialImmediateSuccessors(Sequence, eligible)

This function returns a set of customers that may follow customer i immediately in an optimal solution. This set is a subset of the set returned by *CalcPotentialSuccessors* because it applies additional reasoning to further eliminate customers from being immediately added to the route.

The creation of the set of potential immediate successors relies on the concept of a *local precedence relation* introduced in Avraham and Raviv (2020) and explained here for completeness.

For a given scenario k, we say that customer j_1 $(i, t_k^{'}, k)$-locally precedes customer j_2 if after completing service at customer i at time $t_k^{'}$ in scenario k it is possible to travel from i to j_1, complete the service at j_1, travel to j_2 and arrive there before the opening of its service window, a_{j_2}. The relation holds when the following inequality is true:

$$\max\left(t_k^{'} + t_{ij_1k}(t_k^{'}), a_{j_1}\right) + s_{j_1} + t_{j_1 j_2 k}\left(\max\left(t_k^{'} + t_{ij_1k}(t_k^{'}), a_{j_1}\right) + s_{j_1}\right) \leq a_{j_2}$$

Moreover, if the inequality holds for all scenarios $k = 1, \ldots, K$ and corresponding $t_k^{'}s$, we say that j_1 i-locally precedes customer j_2.

A given *Sequence* uniquely defines its last customer i and $t_k^{'}$ for each of the scenarios. The function returns all the customers j_2 in *Eligible* such that no other customer $j_1 \in$ *Eligible* i-locally precedes them. We quickly construct the output of the function by employing a preprocessing procedure to calculate the relation before launching the B&B algorithm. The technical details of this procedure are described in Avraham and Raviv (2020).

4.2. Memoization

We improved the performance of the B&B algorithm by applying considerations that arise from the observation of Lemma 1 below.

Let $S^{'}$ be a sequence of selected visited customers that ends with the current customer i, and let $C^{'}$ represent the average penalties over all the scenarios accumulated up to the start service at customer i when the route follows $S^{'}$. Let S'' represent an alternative sequence to $S^{'}$ that visits the same customers in a different order and ends with the same customer i. C'' denotes the accumulated average penalty in sequence S''. Given a solution (a sequence) R, we denote the start service time at a particular customer j in scenario k by $t(R, j, k)$.

Lemma 1. *If $C^{'} \leq C''$ and $t(S^{'}, i, k) \leq t(S'', i, k) \forall k = 1, \ldots, K$, then the sequence $S^{'}$ weakly dominates the sequence S''. That is, there exists an optimal sequence that does not contain S'' as a prefix.*

Proof: *Consider two valid solutions of the DD-TD-OP-STW. The first solution is the sequence obtained as a concatenation of the sequences S' and S, i.e., the route first visits the customers in $S^{'}$ and then continues to the customers of S according to the order of these sequences. We denote this solution as $S^{'} + S$. Similarly, the second solution is $S'' + S$. We prove the Lemma by showing that the expected net profit of solution $S'' + S$ is no greater than the expected net profit of $S^{'} + S$.*

Let j denote the first customer in S. For customer j, $t(S^{'} + S, j, k) = max\,(a_j, t(S^{'} + S, i, k) + s_{ik} + t_{i,j,k}(t(S^{'} + S, i, k) + s_{ik}))$ and $t(S'' + S, j, k) = max\,(a_j, t(S'' + S, i, k) + s_{ik} + t_{i,j,k}(t(S'' + S, i, k) + s_{ik}))$. Since the FIFO property in the time-dependent setting ensures that no later departure from origin i can result in an earlier arrival at destination j, $t(S^{'} + S, i, k) + s_{ik} + t_{i,j,k}(t(S^{'} + S, i, k) + s_{ik}) \leq t(S'' + S, i, k) + s_{ik} + t_{i,j,k}(t(S'' + S, i, k) + s_{ik})$; thus, $t(S^{'} + S, j, k) \leq t(S'' + S, j, k) \forall k = 1, \ldots, K$. A similar argument can be used for all the customers in S; i.e., $t(S^{'} + S, p, k) \leq t(S'' + S, p, k) \forall k = 1, \ldots, K, p \in S$.

Recall that $G_j(x)$ is nondecreasing in x. Thus, for each customer p, the penalty incurred when following the sequence $S^{'} + S$ is no greater than the penalty incurred when following the sequence $S'' + S$. Since $C^{'} \leq C''$, the total expected penalty when following the sequence $S^{'} + S$ is no greater than the expected penalty when following the sequence $S'' + S$. Moreover, since the sum of prizes is the same for both $S^{'} + S$ and $S'' + S$, the net profit of the solution $S^{'} + S$ cannot be smaller than the net profit of $S'' + S$. Therefore, there exists an optimal route that does not begin with S''. ∎

We use Lemma 1 to enhance the B&B algorithm. When the algorithm explores a node, it uses memoization to check if previous nodes dominate it in the sense of Lemma 1. Branches of dominated nodes are pruned. Non-dominated nodes are stored in a hash table, and previously stored nodes that are dominated by the current node are deleted from the table. In our preliminary numerical experiment, the memorization mechanism significantly reduced the size of the B&B tree and the running time of the algorithm.

4.3. Hybrid B&B - local search heuristic

In this section, we present a local search (LS) heuristic and integrate it into our B&B procedure. We begin with the presentation of the LS heuristic and discuss its hybridization at the end of the section. The LS heuristic is designed to improve the solutions that are generated at each node while the B&B algorithm runs and thus to find good incumbent solutions in a short time. Its input consists of three components: (1) an initial solution denoted by *CandSolution* and obtained from the greedy algorithm described in Section 4.1, (2) the sequence of selected customers in the node, denoted by *Sequnce*, that cannot be modified (*Sequnce* is a prefix of *CandSolution*), and (3) a set of customers that may (or may not) be visited later in the tour, denoted by *Eligible*. It returns a solution whose value is no worse than the value of *CandSolution*.

The heuristic begins its search with an initial current solution that is set as *CandSolution*. Next, it finds a set of solutions that can be created

from the current solution by some (simple) manipulations (the *neighborhood* of the current solution) and evaluates their values. In each iteration, the entire neighborhood is scanned; if an improved solution is found, the current solution is updated to the best found solution, and the process is repeated. Otherwise, the search ends, and the current solution is returned. Next, further details related to the components of the heuristic are described.

4.3.1. Neighborhood

Recall that any valid solution (and related lower bound) to a node with a sequence of selected customers *Sequnce* must contain *Sequnce* as a prefix. This is guaranteed for the solution received in the input of the heuristic (*CandSolution*, see Section 4.1). We denote the suffix of *CandSolution* that contains all the customers not in *Sequnce* by S'. Next, the neighborhood related to a current solution is defined to maintain this property. The neighborhood is constructed as the union of the following three sets:

Replace Solutions generated from replacing one of the customers of S' with a customer from *Eligible*.

Insert Solutions generated from inserting one of the customers from *Eligible* into S' or immediately before or after it.

Reposition Solutions generated by changing the position of one of the customers of S' to a different position within S'.

Note that while the replace and insert operations may affect both the total sum of the prizes and the total penalty, the reposition operation changes only the total penalty in the solution.

4.3.2. Improvement of generated solutions

Recall that an optimal solution of DD-TD-OP-STW cannot contain customers with a negative expected net profit. Therefore, each solution in the neighborhood of the current solution undergoes a simple iterative improvement process. In each iteration, the earliest customer in the sequence whose expected net profit is negative is identified and removed from the sequence. The process ends when all visited customers yield a positive expected net profit.

4.3.3. Memoization

Recall that evaluation of the value of a given solution (a sequence of customers) is computationally expensive since we need to calculate the average penalty cost over all scenarios. To reduce the running time required for these evaluations, we store all the encountered solutions during the local search along with their values. Thus, although each solution may be encountered many times, it is only evaluated once.

4.3.4. Integration in the B&B algorithm

The local search heuristic is embedded into the B&B algorithm to obtain better lower bounds than those generated by the greedy algorithm (described in Section 4.1). Since the local search procedure is typically computationally intense, we apply it only in the following two cases: (1) at the root node, for the sake of achieving an initial good incumbent solution, and (2) whenever the gap (in percentage) between the lower bound obtained by the simple greedy algorithm and the global lower bound (see Section 4.1) is smaller than some optimality gap, denoted as α. That is, when

$$\frac{GlobalLB}{NewLB} - 1 \leq \alpha$$

Hence, if a new best-known solution is obtained and $NewLB > GlobalLB$, the local search procedure is initiated to further improve the new solution.

Note that as the B&B algorithm runs, *GlobalLB* improves; thus, the probability that the local search procedure will yield solutions that outperform it decreases. Therefore, we gradually reduce the value of α. That is, the initial value of α is set at some value, denoted as α_0, and every Ω times the LS is applied, α is updated as $\alpha \leftarrow \alpha\omega$, where $0 < \omega < 1$ represents a reduction factor.

5. Numerical experiments

In this section, we evaluate the performance of the proposed B&B strategy and its enhancement (hybridization with LS). Since this is the first study on the data-driven time-dependent orienteering problem with soft time windows (DD-TD-OP-STW), it is not possible to compare our results with those of previous studies. Therefore, we report the solution times for a set of instances that could be solved to optimality in reasonable time consistently and the optimality gap of larger and harder instances with a time limit of 24 h.

Section 5.1 presents the problem instances in our benchmark dataset; Section 5.2 reports our results and discusses their implications.

5.1. Problem instances and setup

The set of problem instances is based on time-dependent travel time data between 60 locations in central Israel. The input consists of 40 actual scenarios of travel times between each pair of locations for each interval of 1 min during the working day. Details about the procedure used to collect and process the data from Google Maps are available in Avraham and Raviv (2020). The data are available by request from the first author.

We created problem instances with 24, 30 and 54 customers: twenty instances of each size were created. The customer service times in each of the 40 scenarios of each instance were randomly generated such that the total average time required to serve all the customers was approximately 7 h a day. Across all scenarios, the service times ranged from 3 to 57 (resp., 2–47) minutes in the 24 (resp., 30) customer instances. The values were drawn from a lognormal distribution. The range of the prizes was 8–35 (resp. 7–29) in the 24 (resp., 30) customer instances. Each customer was assigned to one of six nonoverlapping time windows of 90 min each (over a planning horizon of 9 h), such that the customers were equally divided among the intervals. The prize, p_i, for each customer was set as his average service time over all scenarios. The penalty function (for starting the service after the end of the time window) was set to $G_i(o) = p_i o^2$, where o is the lateness in hours. We selected this function because it is convex (long delays are much more costly than short ones), and the penalty is also associated with the prize, which is a proxy for the value of the customer. However, our solution methods can be applied to any nondecreasing penalty function.

In this section, we refer to the Hybrid B&B algorithm as HYBRID and to the B&B algorithm that is not enhanced with a local search heuristic at the nodes as STANDARD. We also compare the results to the solution obtained by our naïve local search heuristic applied at the root node of HYBRID. This algorithm is denoted as LS.

The algorithms were implemented as single-threaded applications in Python 2.7 and tested on an Intel i9-9900K 3.6 GHz desktop with 64 GB RAM running Ubuntu Linux 18.04 using a PyPy interpreter that implements a just-in-time compiler.

5.2. Results

Both the standard and hybrid B&B algorithms were applied to the 60 instances described above with a time limit of 24 h. Following some preliminary experiments, the tuning parameters of the hybrid algorithm were set to $\alpha_0 = 0.1$, $\Omega = 100$ and $\omega = 0.99$. In our preliminary experiment, we tested the hybrid algorithm with 24 and 30 customer instances (20 instances each) with $\Omega = 100$ and $\Omega = 200$. The $\Omega = 100$ performed better on average for both sizes, although the solution times of the 30 customer instances took much longer times than the 24 customer instances. The difference was not large but statistically significant for both problem sizes. We concluded that at least for the instance sizes that we tested, the algorithm's performance is not very

sensitive to the value of Ω. However, if the algorithm is to be implemented repeatedly in a particular setting, it would be advisable to fine tune these parameters.

Table 1a-c report the results of the experiments for the instances with 24, 30, and 54 customers. The leftmost column displays the instance number. Next, the optimality gap of the LS solution is presented as a baseline to compare the performance of the more intricate B&B algorithms. Moreover, for the standard B&B and the hybrid B&B, the following are presented: total running time, the time to encounter the best-known solution, and the optimality gap after 10 min and after 24 h. Notably, all optimality gaps were calculated based on the tightest upper bound found for each instance by the two exact algorithms after 24 h. That is, the optimality gaps were obtained as $\frac{BestUB-Sol}{Sol}$, where *BestUB* denotes the tightest upper bound and *Sol* denotes the value of the solution. The last four rows of the table present summary statistics of the information in the tables.

Table 1a shows that all instances with 24 customers were solved to optimality within a 24-h time limit. While HYBRID requires, on average, somewhat longer time to prove the optimality of a given solution, it greatly reduces the time needed to encounter an optimal solution. The average time is reduced from approximately 1000 s–123 s. This difference is statistically significant, with a p value = 0.006 in a one-sided paired *t*-test.

The two exact algorithms provide a near optimal solution after 10 min of running time for the instances with 24 customers. In fact, HYBRID always found an optimal solution in less than 10 min, while STANDARD failed to find one in some instances. Compared with our naïve LS heuristic, both B&B methods deliver significantly better solutions (p value = 0.015 when compared to STANDARD and p value < 0.0001 when compared to the HYBRID in a paired one-sided *t*-test).

Table 1b shows that when the dimensionality of the instances is increased, the advantage of HYBRID becomes more prominent. Seven of the twenty instances with 30 customers could not be solved to optimality within 24 h, or at least the two algorithms could not prove optimality. For five of these instances, HYBRID yields better solutions than STANDARD.

When observing the instances solved to optimality, we note that incorporating the local search heuristic into the B&B procedure increased the time to reach proven optimality but greatly reduced the time to encounter the optimal solution. Moreover, for these larger instances, HYBRID delivers much better solutions than STANDARD when stopped after 10 min. Interestingly, the naïve LS heuristic also outperforms STANDARD. All the above statements are statistically significant, with p value < 0.0014.

In Table 1c, to explore the limitations of our algorithm, we present the same results for the 54 instances. None of the instances were solved to optimality or near optimality. The pure B&B algorithm provided solutions with an average optimality gap of 82.50%, and our hybrid algorithm ended with an optimality gap of 16.90%. However, in all 20 cases, the hybrid algorithm found the best solution after a few minutes by running a local search at the root node. The rest of the 24 h were used only to tighten the upper bound. This finding motivates the future development of advanced local search heuristics for the problem (e.g., ALNS).

Table 2 presents an analysis of the special features of HYBRID. For each set of instances, we present the total number of local searches initiated, the number of successful local searches and the percentage of solution evaluations saved during the LS processes due to the memoization. Notably, successful local searches are defined as searches initiated at nodes that result in a solution better than the current best-known solution.

In Table 2, we observe that only a small fraction of the local searches actually find new best-known solutions. However, we observed earlier that these searches significantly shorten the time needed for the B&B procedure to find near-optimal solutions. This finding highlights the effectiveness of the hybridization of LS, and possibly other fast heuristics, with B&B procedures and calls for considering such enhancement of the B&B, especially in hard problems where the calculation of the bounds at each node is expensive, for example, when the evaluation of multiple scenarios is needed.

As expected, the algorithm initiates a greater number of searches when solving larger instances. Finally, the merits of memoization are evident. Approximately 19% of all solution evaluations during the LS runs can be saved by reusing stored results.

Table 1a
Performance of the two algorithms for instances with 24 customers.

Inst.	LS	STANDARD				HYBRID			
	Opt. Gap	Running time (s)	Time to encounter best known solution (s)	Gap 10 m	Gap after 24 h	Running time (s)	Time to encounter best known solution (s)	Gap 10 m	Opt. Gap 24 h
1	0.00%	247.5	237.6	0.0%	0.0%	306.4	2.4	0.0%	0.0%
2	0.00%	4,894.7	31.5	0.0%	0.0%	5,016.3	85.4	0.0%	0.0%
3	0.00%	702.6	58.5	0.0%	0.0%	827.3	2.3	0.0%	0.0%
4	4.71%	314.6	83.0	0.0%	0.0%	452.5	85.3	0.0%	0.0%
5	1.87%	1,128.5	1,128.4	1.63%	0.0%	1,366.6	11.6	**0.0%**	0.0%
6	1.21%	704.3	704.3	0.81%	0.0%	864.0	227.2	**0.0%**	0.0%
7	2.21%	613.6	319.1	0.0%	0.0%	835.9	90.4	0.0%	0.0%
8	0.02%	270.6	270.5	0.0%	0.0%	384.5	7.3	0.0%	0.0%
9	2.55%	546.1	144.2	0.0%	0.0%	670.2	104.5	0.0%	0.0%
10	2.08%	3,132.0	2,227.4	2.53%	0.0%	3,597.7	178.2	**0.0%**	0.0%
11	1.49%	879.9	553.3	0.0%	0.0%	1,114.3	12.2	0.0%	0.0%
12	0.47%	5,021.3	5,021.1	1.65%	0.0%	5,705.3	163.5	**0.0%**	0.0%
13	0.31%	118.6	118.6	0.0%	0.0%	158.1	121.0	0.0%	0.0%
14	1.59%	4,076.0	4,075.6	1.46%	0.0%	4,738.6	149.1	**0.0%**	0.0%
15	0.05%	3,331.4	745.3	0.54%	0.0%	3,852.7	9.5	0.0%	0.0%
16	1.24%	3,472.9	2,895.1	2.82%	0.0%	4,078.5	463.5	**0.0%**	0.0%
17	3.56%	389.4	164.2	0.0%	0.0%	517.8	228.7	0.0%	0.0%
18	4.31%	1,944.6	357.7	0.0%	0.0%	2,211.4	32.3	0.0%	0.0%
19	0.88%	1,095.2	615.6	0.17%	0.0%	1,455.9	376.8	**0.0%**	0.0%
20	1.11%	451.3	261.9	0.0%	0.0%	616.3	108.4	0.0%	0.0%
average	1.48%	1,666.8	1,000.6	0.58%	0.0%	1,938.5	123.0	0.0%	0.0%
median	1.22%	792.1	338.4	0.00%	0.0%	989.2	97.5	0.0%	0.0%
min	0.00%	118.6	31.5	0.00%	0.0%	158.1	2.3	0.0%	0.0%
max	4.71%	5,021.3	5,021.1	2.82%	0.0%	5,705.3	463.5	0.0%	0.0%

Table 1b
Performance of the two algorithms for instances with 30 customers.

Inst.	LS	STANDARD				HYBRID			
	Opt. Gap	Running time (s)	Time to encounter best known solution (s)	Gap 10 m	Gap after 24 h	Running time (s)	Time to encounter best known solution (s)	Gap 10 m	Opt. Gap 24 h
1	1.23%	3,374	1,778	15.52%	0.0%	4,165	2,312	1.23%	0.0%
2	1.38%	11,866	4,845	24.83%	0.0%	13,968	2,850	1.38%	0.0%
3	3.77%	86,400	84,764	16.07%	5.4%	86,400	5,998	3.77%	**3.4%**
4	2.56%	18,361	18,358	35.99%	0.0%	20,955	10,416	2.56%	0.0%
5	0.00%	66,100	44,226	5.06%	0.0%	70,925	7	0.00%	0.0%
6	4.74%	7,525	7,525	20.05%	0.0%	8,767	8,767	4.74%	0.0%
7	0.56%	51,871	38,409	14.32%	0.0%	61,615	1,560	0.56%	0.0%
8	1.48%	33,057	33,053	31.84%	0.0%	36,953	36,952	1.48%	0.0%
9	1.42%	4,943	1,120	0.18%	0.0%	5,575	259	0.00%	0.0%
10	0.81%	86,400	50,680	3.00%	0.7%	86,400	53,559	0.73%	0.7%
11	1.04%	21,344	21,341	11.71%	0.0%	26,575	3,916	1.04%	0.0%
12	9.28%	86,400	19,271	19.29%	9.1%	86,400	1,478	9.28%	**8.0%**
13	1.34%	41,271	41,266	7.68%	0.0%	47,597	1,062	0.21%	0.0%
14	0.50%	48,434	6,911	19.87%	0.0%	50,591	6,792	0.50%	0.0%
15	8.46%	86,400	25,083	14.69%	6.1%	86,400	23,558	6.80%	**5.1%**
16	5.72%	86,400	85,611	8.78%	4.9%	86,400	62,329	4.69%	**4.4%**
17	2.81%	86,400	63,657	6.28%	1.0%	86,400	511	0.82%	**0.8%**
18	2.26%	60,318	45,608	14.00%	0.0%	68,316	8,269	2.26%	0.0%
19	3.07%	86,400	7,052	14.18%	1.7%	86,400	1,143	3.07%	1.7%
20	0.28%	82,762	19,657	41.43%	0.0%	84,646	8,010	0.28%	0.0%
average	2.64%	52,801	31,011	16.24%	1.4%	55,272	11,987	2.27%	**1.2%**
median	1.45%	56,095	23,212	14.50%	0.0%	64,965	4,957	1.30%	0.0%
min	0.00%	3,374	1,120	0.18%	0.0%	4,165	7	0.00%	0.0%
max	9.28%	86,400	85,611	41.43%	9.1%	86,400	62,329	9.28%	**8.0%**

Table 1c
Performance of the two algorithms for the instances with 54 customers.

Inst.	LS	STANDARD				HYBRID			
	Opt. Gap	Running time (s)	Time to find best known solution (s)	Gap 10 m	Gap after 24 h	Running time (s)	Time to find best known solution (s)	Gap 10 m	Opt. Gap 24 h
1	16.06%	86,400	68,662.42	162.01%	77.04%	86,400	106.57	16.06%	16.06%
2	15.73%	86,400	23,901.74	162.68%	57.28%	86,400	123.04	15.73%	15.73%
3	22.11%	86,400	84,917.22	99.61%	87.05%	86,400	101.30	22.11%	22.11%
4	17.37%	86,400	85,751.65	123.64%	104.67%	86,400	110.67	17.37%	17.37%
5	17.93%	86,400	79,582.35	147.59%	116.42%	86,400	117.92	17.93%	17.93%
6	14.68%	86,400	86,098.36	156.62%	58.72%	86,400	140.23	14.68%	14.68%
7	17.01%	86,400	82,721.94	141.08%	122.40%	86,400	155.95	17.01%	17.01%
8	13.60%	86,400	69,859.47	212.90%	58.07%	86,400	176.57	13.60%	13.60%
9	14.62%	86,400	79,244.39	100.18%	52.27%	86,400	145.26	14.62%	14.62%
10	15.03%	86,400	86,215.88	195.00%	67.90%	86,400	148.43	15.03%	15.03%
11	21.81%	86,400	80,104.09	138.18%	121.03%	86,400	127.40	21.81%	21.81%
12	18.62%	86,400	77,567.05	119.30%	96.28%	86,400	175.62	18.62%	18.62%
13	12.75%	86,400	74,675.68	153.40%	58.03%	86,400	161.23	12.75%	12.75%
14	20.58%	86,400	69,505.37	109.98%	96.06%	86,400	95.27	20.58%	20.58%
15	17.06%	86,400	66,404.10	177.83%	75.71%	86,400	113.11	17.06%	17.06%
16	14.68%	86,400	85,589.89	180.36%	58.02%	86,400	100.54	14.68%	14.68%
17	24.76%	86,400	61,903.11	135.46%	112.76%	86,400	102.09	24.76%	24.76%
18	13.32%	86,400	61,562.02	114.64%	98.51%	86,400	151.43	13.32%	13.32%
19	16.50%	86,400	84,551.96	135.64%	75.75%	86,400	122.40	16.50%	16.50%
20	13.74%	86,400	74,056.38	122.15%	56.01%	86,400	126.19	13.74%	13.74%
average	16.90%	86,400	74,143.75	144.41%	82.50%	86,400	130.06	16.90%	16.90%
median	16.28%	86,400	78,405.72	139.63%	76.40%	86,400	124.61	16.28%	16.28%
min	12.75%	86,400	23,901.74	99.61%	52.27%	86,400	95.27	12.75%	12.75%
max	24.76%	86,400	86,215.88	212.90%	122.40%	86,400	176.57	24.76%	24.76%

5.3. Comparison between the B&B and local search solutions

In this section, we compare the characteristics of the results pertaining to the solutions obtained via the two methods: Hybrid B&B and LS. We focus on the 24 customer instances where we can solve all the instances to optimality with the B&B method. In Table 3, we present the breakdown of the objective function value to the prizes and the lateness penalty components as well as the average lateness per visited customer. The objective function values are also presented, and optimal solutions are in bold font.

It is apparent from the table that the optimal solutions (obtained via B&B) are on average approximately 1.5% better than those obtained via LS and that the average improvement stems from both the penalty and prizes components. However, in some particular instances, the optimal solution incurs much higher expected penalties than the LS solutions. In both cases, the penalties are responsible for only 2.7–3% of the objective function, which shows that the optimal and near-optimal solutions prevent misses of the time windows most of the time.

Table 2
Local search heuristic measures.

Instance	Instances with 24 Customers			Instances with 30 Customers		
	Number of local searches initiated	Number of successful local searches	% of LS solution evaluations saved	Number of local searches initiated	Number of successful local searches	% of LS solution evaluations saved
1	1,432	1	21.2%	7,173	6	22.43%
2	9,743	1	18.4%	8,528	4	18.73%
3	9,284	1	16.4%	4,256	2	16.08%
4	8,643	7	20.8%	11,310	8	18.66%
5	5,621	5	23.2%	8,180	1	22.76%
6	1,619	4	18.3%	6,653	9	19.24%
7	5,300	4	21.4%	11,386	3	19.20%
8	2,758	2	22.6%	5,711	6	18.79%
9	1,410	4	18.8%	15,055	5	20.57%
10	4,934	6	16.5%	15,080	5	15.55%
11	6,705	3	21.6%	8,846	4	20.12%
12	6,134	2	18.9%	11,636	5	17.28%
13	556	3	21.1%	10,090	4	21.23%
14	6,980	3	17.9%	17,295	5	15.87%
15	11,190	2	15.8%	10,235	5	13.88%
16	6,593	4	17.2%	13,021	8	13.82%
17	5,628	4	21.7%	11,983	5	17.15%
18	9,987	2	21.1%	6,890	3	19.22%
19	7,139	4	17.4%	18,188	3	18.79%
20	4,190	4	20.9%	15,384	2	19.16%
average	5,792	3.3	19.6%	10,845	5	18.43%
median	5,881	3.5	19.9%	10,773	5	18.79%
min	556	1	15.8%	4,256	1	13.82%
max	11,190	7	23.2%	18,188	9	22.76%

Table 3
Breakdown of the objective function under both solution methods.

Inst.	LS solution (Heuristic)				Hybrid branch & bound (Optimal)			
	Total prizes	Total lateness penalty	Net profit (Objective function)	Average lateness (min.)	Total prizes	Total lateness penalty	Net profit (Objective function)	Average lateness (min.)
1	13,880	204.34	**13,675.66**	1.77	13,880	204.34	**13,675.66**	1.77
2	14,400	312.48	**14,087.52**	2.68	14,400	312.48	**14,087.52**	2.68
3	14,880	732.66	**14,147.34**	5.37	14,880	732.66	**14,147.34**	5.37
4	13,440	542.99	12,897.01	4.94	14,240	735.80	**13,504.20**	5.73
5	13,720	241.63	13,478.37	2.57	14,040	309.86	**13,730.14**	3.53
6	13,240	296.12	12,943.88	2.94	13,360	259.40	**13,100.60**	2.23
7	13,760	705.29	13,054.71	5.02	13,600	256.47	**13,343.53**	2.77
8	14,080	169.23	13,910.77	1.67	14,080	166.01	**13,913.99**	1.67
9	13,720	533.80	13,186.20	4.77	14,240	717.55	**13,522.45**	5.97
10	13,880	437.34	13,442.66	3.66	14,040	318.32	**13,721.68**	2.61
11	14,000	594.92	13,405.08	4.40	14,000	395.81	**13,604.19**	3.10
12	14,920	404.23	14,515.77	2.45	14,920	336.07	**14,583.93**	2.21
13	12,600	369.31	12,230.69	4.42	12,400	131.47	**12,268.53**	2.14
14	14,520	209.06	14,310.94	1.92	14,800	261.51	**14,538.49**	1.82
15	14,400	411.04	13,988.96	4.08	14,480	483.75	**13,996.25**	4.80
16	14,000	283.67	13,716.33	2.29	14,200	313.95	**13,886.05**	2.54
17	14,480	330.64	14,149.36	3.71	15,200	546.58	**14,653.42**	4.79
18	13,680	819.00	12,861.00	5.23	14,040	624.99	**13,415.01**	4.19
19	13,960	551.36	13,408.64	5.58	13,960	432.84	**13,527.16**	4.83
20	13,480	163.11	13,316.89	2.24	13,600	134.83	**13,465.17**	1.58
Average	**13,952**	**415.61**	**13536.39**	**3.59**	**14,118**	**383.73**	**13,734.27**	**3.32**

In Table 4, we provide information about the similarity of the solutions obtained via the two methods. For each instance, we present.

- The number of customers visited under the solution.
- The cardinality of the symmetric difference between the sets of visited customers. That is, the number of customers visited under one of the solutions but not under the other
- The average offset in the positions of the customers visited under the two solutions.
- The number of customers visited under the two solutions.
- The number of customers visited under the two solutions at the same position in the sequence.

We observe in the table that the solutions obtained by the two methods are very similar in nature. Both include approximately 20 customers on average. The symmetric difference is 2.55 on average, meaning that the two solutions share most of the visited customers, and the sequence is similar, with an average offset of 0.67. On average, the solutions have 18.45 customers in common, and more than 10 are in the same position in the sequence.

5.4. Comparison with simpler models

As we observed in the previous sections, the DD-TD-OP-STW problem is an intricate routing problem that takes into account both the time dependency and stochasticity of travel times. It creates a sequence of

Table 4
Similarity between the visit sequence under the two solution methods.

Inst.	Customers under optimal solution	Customers under LS solution	Symetric difference	Average offset	Customers under both solutions	Customers in the same position
1	20	20	0	0.00	20	20
2	20	20	0	0.00	20	20
3	20	20	0	0.00	20	20
4	21	19	8	0.88	16	5
5	21	20	5	1.33	18	4
6	19	19	2	0.39	18	11
7	21	21	2	0.95	20	8
8	21	21	0	0.86	21	10
9	20	19	1	1.53	19	6
10	19	18	3	0.35	17	12
11	21	20	3	1.21	19	3
12	19	19	0	0.42	19	15
13	20	20	2	0.11	19	17
14	20	19	3	0.56	18	8
15	19	19	2	0.50	18	15
16	19	18	5	0.50	16	8
17	21	19	6	1.24	17	0
18	20	19	3	1.61	18	0
19	19	19	2	0.78	18	8
20	20	20	4	0.22	18	14
Average	20	19.45	2.55	0.67	18.45	10.20

customer visits that optimally balances the expected sum of prizes and penalties. The former represents the revenue of the service provider, and the latter represents the service level.

In a soft time-window setting, any sequence of customers is a feasible solution, and even when reality is noisy and time dependent, a solution method that ignores time dependence and stochasticity yields a solution that can be used in practice. For example, travel times can be set to be deterministic and fixed over time, deterministic but time dependent or stochastic (represented by a set of scenarios) and derived from the same distribution independent of departure time. These simplifications result in three special cases of the DD-TD-OP-STW, which are denoted as OP-STW, TD-OP-STW, and DD-OP-STW. Solutions obtained via such simpler approaches are likely to incur more penalties when applied in practice, but obliviousness to this risk may lead to the collection of more prizes. We note that the solution times of the OP-STW and TD-OP-STW are about 40 times shorter while the running DD-OP-STW is about half of the full DD-TD-OP-STW model.

Whether the extra effort required to solve the richer model pays off is an interesting question. To this end, we solve the 20 instances of 24 customers using the above four orienteering models and calculate the value of their solutions in the stochastic time-dependent environment represented by our 40 scenarios. For each solved instance, we calculate the relative gap between the solution value (expected net profit) of each of the simpler models and that of the full model. We also repeat this calculation for the two components of the objective function, namely, the total prizes and the expected total penalties. In Table 5, we present the average and ranges of these relative gaps. A positive gap value implies that the focal model lags behind the full one. Negative gaps imply that the simpler model outperforms the full model.

It is apparent from the table that the average penalty incurred by the service provider is significantly higher in the simpler models, especially when both time dependence and stochasticity are ignored. However, as expected, the provider is able to collect more prizes. The full model features a 0.96% gain on average compared with the OP-STW model. The average gaps are smaller when a complicating property (either time dependence or stochasticity) is introduced.

6. Conclusions and future research

This is the first study to consider the data-driven time-dependent orienteering problem with soft time windows. While the orienteering problem and its variants are well studied, the challenge of planning an optimal orienteering tour in a realistic environment where travel times are unknown and time dependent has been largely overlooked.

Our approach to modeling randomness is based on optimizing the route with respect to multiple representative scenarios collected from recent historical data. Recent advances in geographical information technology have made historical data on travel times on any possible route readily available, for example, from services such as Google Maps. Moreover, using historical scenarios enables the planner to capture the spatial and temporal interdependencies in such a way that cannot be accomplished by fitting a known joint distribution to the travel times.

We presented an enhanced B&B algorithm to solve the problem and showed that it can be used to solve instances with up to 30 customers and 40 scenarios in a reasonable time. We demonstrate numerically that enhancing the B&B algorithm by integrating a local search procedure at the nodes can significantly accelerate convergence to a near-optimal solution.

We believe that future research on DD-TD-OP-STW should focus on developing heuristics that can deliver high-quality solutions in a short time. Such heuristic methods can be used as building blocks for methods to solve the equivalent team orienteering problem. Another interesting direction for research is to consider a dynamic version of the problem where the route is determined during its execution as partial information about the stochastic parameters is revealed.

Table 5
The DD-TD-OP-STW model versus simpler models.

Model	Net profit			Sum of prizes			Penalties		
	Average	Min	Max	Average	Min	Max	Average	Min	Max
OP-STW	0.96%	0.00%	3.80%	−1.25%	−4.12%	0.84%	111.84%	−14.90%	545.17%
TD-OP-STW	0.47%	0.00%	2.16%	−1.15%	−3.87%	0.84%	85.45%	−14.90%	373.72%
DD-OP-STW	0.34%	0.00%	1.53%	−0.62%	−4.12%	0.84%	55.53%	−15.08%	509.51%

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Edison Avraham reports financial support was provided by Shlomo-Shmeltzer institute for smart transportation in Tel-Aviv university. Tal Raviv reports financial support was provided by Israel Science Foundation (ISF).

Acknowledgment

This research was supported by the Israel Science Foundation (ISF) grant no 1367/17.

The first author is partially supported by a scholarship from the Shlomo-Shmeltzer institute for smart transportation in Tel-Aviv university.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.ejtl.2023.100112.

Appendix. linear formulation of the problem

Using the same parameters as those used in Section 3 and an upper bound, T, on the latest time of start service at a customer, we formulate the DD-TD-OP-STW problem using the following decision variables:

x_{ij} Binary variable that equals 1 if customer j is visited immediately after customer i.

u_{ijkt} Binary variable that equals 1 if the vehicle departs from customer i to customer j at time t in scenario k..

The decision variables u_{ijkt} define the arcs of K copies of a time-expanded graph, one for each scenario. Our problem can be viewed as finding a path that visits some customers on each copy in the same sequence across all scenarios.

$$maximize \sum_{i=1}^{n} p_i \sum_{j=0}^{n} x_{ij} - \frac{1}{K} \sum_{i,j=1}^{n} \sum_{k=1}^{K} \sum_{t=b_i+s_{ik}+1}^{T} G_i(t - s_{ik} - b_i) u_{ijkt} \tag{10}$$

subject to

$$\sum_{j=0}^{n} x_{ij} = \sum_{j=0}^{n} x_{ji} \forall i = 0, \ldots, n \tag{11}$$

$$\sum_{j=0}^{n} x_{ij} \leq 1 \forall i = 0, \ldots, n \tag{12}$$

$$\sum_{t=a_i+s_{ik}}^{T} u_{ijkt} = x_{ij} \forall i, j = 0, \ldots, n, k = 1, \ldots, K \tag{13}$$

$$\sum_{i'=1}^{n} \sum_{t=a_j+s_{jk}}^{T} t u_{ji'kt} \geq \sum_{t=a_i+s_{sik}}^{T} t u_{ijkt} + \sum_{t=a_i+s_{ik}}^{T} t_{ijk}(t) u_{ijkt} + s_{jk} - T(1 - x_{ij}) \forall i, j = 1, \ldots, n, k = 1 \ldots .K \tag{14}$$

$$u_{ijkt} \in \{0, 1\} \forall i, j = 0, \ldots, n; k = 1, \ldots K, t = a_i + s_{ik}, \ldots, T \tag{15}$$

$$x_{ij} \in \{0, 1\} \forall i, j = 0, \ldots, n \tag{16}$$

The objective function (10) maximizes the sum of the prizes net of the time window violation penalty. Constraints (11) and (12) are identical to Constraints (2) and (3) of the original model. Constraint (13) stipulates that if customer j is visited immediately after customer i, a corresponding arc is selected on each of the time-expanded graphs. Constraint (14) ensures that the departure times in each of the scenarios are compatible with the travel and service times of the scenario and the sequence (common to all of the scenarios). This constraint also enforces service only after the opening of the time windows and eliminates subtours together with (12) and (13). Constraints (15) and (16) define the domains of the decision variables.

We note that the number of binary decision variables u_{ijkt} is approximately n^2KT. The smallest problem presented in Section 5 was related to working days with 24 customers, 40 scenarios and 540 time units (of 1 min). This leads to a model with more than 10 million binary variables.

References

Abbaspour, R.A., Samadzadegan, F., 2011. Time-dependent personal tour planning and scheduling in metropolises. Expert Syst. Appl. 38 (10), 12439–12452.

Angelelli, E., Archetti, C., Filippi, C., Vindigni, M., 2017. The probabilistic orienteering problem. Comput. Oper. Res. 81, 269–281.

Angelelli, E., Archetti, C., Filippi, C., Vindigni, M., 2021. A dynamic and probabilistic orienteering problem. Comput. Oper. Res. 136, 105454.

Avraham, E., Raviv, T., 2020. The data-driven time-dependent traveling salesperson problem. Transport. Res. Part B 134, 25–40.

Bian, Z., Liu, X., 2018. A real-time adjustment strategy for the operational level stochastic orienteering problem: a simulation-aided optimization approach. Transport. Res. Part E 115, 246–266.

Campbell, A.M., Gendreau, M., Thomas, B.W., 2011. The orienteering problem with stochastic travel and service times. Ann. Oper. Res. 186 (1), 61–81.

Çelik, Ş., 2021. Team Orienteering Problem with Stochastic Time-dependent Travel Time, Master's Thesis. Bilkent University.

Chou, X., Gambardella, L.M., Montemanni, R., 2021. A tabu search algorithm for the probabilistic orienteering problem. Comput. Oper. Res. 126, 105107.

Dolinskaya, I., Shi, Z., Smilowitz, K., 2018. Adaptive orienteering problem with stochastic travel times. Transport. Res. Part E 109, 1–19.

Evers, L., Glorie, K., van der Ster, S., Barros, A.I., Monsuur, H., 2014. A two-stage approach to the orienteering problem with stochastic weights. Comput. Oper. Res. 43, 248–260.

Fomin, F., Lingas, A., 2002. Approximation algorithms for time-dependent orienteering. Inf. Process. Lett. 83, 57–62.

Garcia, A., Arbelaitz, O., Vansteenwegen, P., Souffriau, W., Linaza, M.T., 2010. Hybrid approach for the public transportation time dependent orienteering problem with time windows. In: Corchado, E., Romay, M.G., Savio, A.M. (Eds.), Hybrid Artificial Intelligence Systems, Lecture Notes in Artificial Intelligence, vol. 6077. Springer, Berlin, Germany, pp. 151–158.

Garcia, A., Vansteenwegen, P., Arbelaitz, O., Souffriau, W., Linaza, M.T., 2013. Integrating public transportation in personalised electronic tourist guides. Comput. Oper. Res. 40 (3), 758–774.
Gavalas, D., Konstantopoulos, C., Mastakas, K., Pantziou, G., Vathis, N., 2014. Efficient heuristics for the time dependent team orienteering problem with time windows. In: Gupta, P., Zaroliagis, C. (Eds.), Applied Algorithms, Lecture Notes in Computer Science, vol. 8321. Springer, pp. 152–163.
Gavalas, D., Konstantopoulos, C., Mastakas, K., Pantziou, G., Vathis, N., 2015. Heuristics for the time dependent team orienteering problem: application to tourist route planning. Comput. Oper. Res. 62, 36–50.
Gunawan, A., Chuin Lau, H., Vansteenwegen, P., 2016. Orienteering Problem: a survey of recent variants, solution approaches and applications. Eur. J. Oper. Res. 255, 315–332.
Gunawan, A., Yuan, Z., Lau, H.C., 2014. A mathematical model and metaheuristics for time dependent orienteering problem. In: Proceedings of the 10th International Conference of the Practice and Theory of Automated Timetabling (Patat 2014), pp. 202–217. York, United Kingdom.
Gupta, A., Krishnaswamy, R., Nagarajan, V., Ravi, R., 2015. Running errands in time: approximation algorithms for stochastic orienteering. Math. Oper. Res. 40 (1), 56–79.
Ilhan, T., Iravani, S.M.R., Daskin, M.S., 2008. The orienteering problem with stochastic profits. IIE Trans. 40 (4), 406–421.
Jiang, Y., Weise, T., Lässig, J., Chiong, R., Athauda, R., 2014. Comparing a hybrid branch and bound algorithm with evolutionary computation methods, local search and their hybrids on the tsp. In: 2014 IEEE Symposium on Computational Intelligence in Production and Logistics Systems (CIPLS), pp. 148–155.
Karunakaran, D., Mei, Y., Zhang, M., 2019. Multitasking genetic programming for stochastic team orienteering problem with time windows. In: Proceedings of the IEEE Symposium Series on Computational Intelligence, pp. 1598–1605. SSCI 2019.
Khodadadian, M., Divsalar, A., Verbeeck, C., Gunawan, A., Vansteenwegen, P., 2022. Time dependent orienteering problem with time windows and service time dependent profits. Comput. Oper. Res. 143, 105794.
Lau, H.C., Yeoh, W., Varakantham, P., Nguyen, D.T., Chen, H., 2012. Dynamic stochastic orienteering problems for risk-aware applications. In: Proceedings of the 28th Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-12), pp. 448–458 (Corvallis, Oregon).
Laporte, G., Martello, S., 1990. The selective travelling salesman problem. Discrete Appl. Math. 26, 193–207.
Li, J., 2012. Research on team orienteering problem with dynamic travel times. J. Software 7 (2), 249–255.
Liao, Z., Zheng, W., 2018. Using a heuristic algorithm to design a personalized day tour route in a time-dependent stochastic environment. Tourism Manag. 68, 284–300.
Mei, Y., Salim, F.D., Li, X., 2016. Efficient meta-heuristics for the multi-objective time-dependent orienteering problem. Eur. J. Oper. Res. 254, 443–457.
Papapanagiotou, V., Montemanni, R., Gambardella, L.M., 2014. Objective function evaluation methods for the orienteering problem with stochastic travel and service times. Journal of Applied Operations Research 6 (1), 16–29.
Papapanagiotou, V., Montemanni, R., Gambardella, L.M., 2015. Hybrid sampling-based evaluators for the orienteering problem with stochastic travel and service times. Journal of Traffic and Logistics Engineering 3 (2), 108–114.
Panadero, J., Juan, A.A., 2020. Maximizing reward from a team of surveillance drones: a simheuristic approach to the stochastic team orienteering problem. Eur. J. Ind. Eng. 14 (4), 485–516.
Peng, G., Dewil, R., Verbeeck, C., Gunawan, A., Xing, L., Vansteenwegen, P., 2019. Agile earth observation satellite scheduling: an orienteering problem with time-dependent profits and travel times. Comput. Oper. Res. 111, 84–98.
Peng, G., Song, G., Xing, L., Gunawan, A., Vansteenwegen, P., 2020. An exact algorithm for agile earth observation satellite scheduling with time-dependent profits. Comput. Oper. Res. 120, 104946.
Ramesh, R., Yoon, Y., Karwan, M., 1992. An optimal algorithm for the orienteering tour problem. ORSA J. Comput. 4, 155–165.
Song, Y., Ulmer, M.W., Thomas, B.W., Wallace, S.W., 2020. Building trust in home services - stochastic team-orienteering with consistency constraints. Transport. Sci. 54 (3), 823–838.
Tang, H., Miller-Hooks, E., 2005. Algorithms for a stochastic selective travelling salesperson problem. J. Oper. Res. Soc. 56 (4), 439–452.
Teng, S.Y., Ong, H.L., Huang, H.C., 2004. An integer L-shaped algorithm for time-constrained traveling salesman problem. Asia Pac. J. Oper. Res. 21 (2), 241–257.
Vansteenwegen, P., Souffriau, W., Van Oudheusden, D., 2011. The orienteering problem: a survey. Eur. J. Oper. Res. 209, 1–10.
Varakantham, P., Kumar, A., 2013. Optimization approaches for solving chance constrained stochastic orienteering problems. In: Perny, P., Pirlot, M., Tsoukiàs, A. (Eds.), Algorithmic Decision Theory, Lecture Notes in Computer Science, vol. 8176. Springer, pp. 387–398.
Varakantham, P., Kumar, A., Lau, H.C., 2018. Risk-sensitive stochastic orienteering problems for trip optimization in urban environments. ACM Transactions on Intelligent Systems and Technology 9 (3), 24:1-24:25.
Verbeeck, C., Sörensen, K., Aghezzaf, E.H., Vansteenwegen, P., 2014. A fast solution method for the time-dependent orienteering problem. Eur. J. Oper. Res. 236 (2), 419–432.
Verbeeck, C., Vansteenwegen, P., Aghezzaf, E.H., 2016. Solving the stochastic time-dependent orienteering problem with time windows. Eur. J. Oper. Res. 255 (3), 699–718.
Verbeeck, C., Vansteenwegen, P., Aghezzaf, E.H., 2017. The time-dependent orienteering problem with time windows: a fast ant colony system. Ann. Oper. Res. 254, 481–505.
Zhang, S., Ohlmann, J.W., Thomas, B.W., 2014. A priori orienteering with time windows and stochastic wait times at customers. Eur. J. Oper. Res. 239 (1), 70–79.
Zhang, S., Ohlmann, J.W., Thomas, B.W., 2018. Dynamic orienteering on a network of queues. Transport. Sci. 52 (3), 691–706.