The Hazardous Orienteering Problem

```
@article{Santini2022,
   title={The Hazardous Orienteering Problem},
   journal={Preprint},
   author={Santini, Alberto and Archetti, Claudia},
   year={2022}
}
```

The Hazardous Orienteering Problem

Alberto Santini[1] and Claudia Archetti[2]

[1]Department of Economics & Business, Universitat Pompeu Fabra, Spain; ESSEC Business School, France; Institute for Advanced Studies, Paris Cergy Université, France
[2]ESSEC Business School, France

5th April 2022

Abstract

This paper studies the Hazardous Orienteering Problem (HOP), a variant of the more famous Orienteering Problem (OP). In the OP, a vehicle earns a profit for each customer it visits (e.g., to pick up a parcel) subject to an upper bound on the tour time. In the HOP, the parcels picked up at some customers have a probability of exploding which depends on how long they travel on the vehicle and, if any parcel explodes, the entire collected profit is lost. The goal is to determine the tour that maximises the expected profit. The problem has interesting applications in routing of hazardous material, cash-in-transit and law enforcement. We propose a mixed-integer non-linear formulation and techniques both to obtain dual bounds and to produce primal solutions. Computational tests investigate the efficacy of the methods proposed and allow to gain insights into solution features.

Keywords: Orienteering Problem, Primal and Dual Bounds, Hazardous Material Transportation, Cash-in-transit logistics

1 Introduction

Travelling Salesman Problems (TSP) with profits are well-studied combinatorial optimisation problems. The TSP asks to find the cheapest Hamiltonian cycle on a graph with costs on the edges (or arcs). Interesting variants occur when not all vertices need to be visited and profits are associated with each of them. In this case, there are three main problems which collectively fall under the umbrella of TSP with profits:

- The Orienteering Problem (OP), which asks to collect the highest possible profit, while respecting an upper bound on travel costs (or time).
- The Prize-collecting TSP, which asks to minimise the tour cost (or time), while respecting a lower bound on the amount of profit collected.
- The Profitable Tour Problem, which asks to maximise the difference between collected profits and tour cost.

In this paper we consider a novel stochastic generalisation of the OP, which arises in the context of routing of hazardous material, law enforcement with drones, and transportation of high-value objects. Consider the following scenarios.

Routing of hazardous material. A logistic company collecting parcels at client sites must decide which pickups to accept and how to route a vehicle to satisfy the corresponding requests. Each accepted pickup earns a profit, so the company would like to accept as many as possible. However, the number of accepted pickups is limited by the total time the vehicle can spend en route, e.g., to comply with driver regulations or because collected parcels must be available at a warehouse by a given deadline. This setting can be modelled as an Orienteering Problem. In our case, however, some requests involve hazardous material. Such material has a small (but non-zero) probability to catch fire and cause the loss of all transported parcels. The probability of this event is different for each hazardous shipment and is directly proportional to the time the shipment spends in the vehicle, following an exponential distribution. For example, lithium-ion batteries have a small probability to catch fire under mechanical stress suffered on a road vehicle [10, 17].

Law enforcement with drones. A law enforcement agency must decide which targets to visit with an unmanned aerial vehicle, subject to a maximum travel distance dictated by the range of the aircraft. Each target has an associated profit, which reflects how important it is to perform a reconnaissance action at the corresponding location. This scenario can also be modelled as an Orienteering Problem. However, visiting some of the targets allows criminals to detect the aircraft and try to interfere with its operation. The more targets are visited and the longer the aircraft keeps flying afterwards, the higher is the chance of being intercepted or shot down—a catastrophic event which we can model with the loss of the entire collected profit.

Cash-in-transit. A security transport company receives requests for cash pickup from several shops, to deposit their daily income to a bank. The company charges each customer that it can include in its tour. The scenario can be modelled as an Orienteering Problem, given a bound on the travel time due to, e.g., the security guards' work shift duration. The more customers are visited and the higher the amount of cash collected, the more the vehicle becomes an interesting target for assailants: each additional minute spent on the road gives potential robbers a chance to attack the vehicle and cause the total loss of its content.

To model the above scenarios, we introduce the stochastic **Hazardous Orienteering Problem** (HOP). The difference with the classical OP is that visiting some "hazardous" (or "time-bomb") customers causes that, after the visit, the vehicle is at risk of losing its entire collected profit. This probability increases with the number of hazardous customers visited and with the time that the vehicle travels after visiting each hazardous customer. Although, as seen above, the HOP models different real-life applications, in the rest of the paper we will use logistic-based terminology and we will say that the vehicle visits "customers" to collect "parcels", which can "explode" en route (an event which we denote as a "catastrophic event"). We propose a problem formulation which involves the maximisation of a non-concave objective function, together with a set of valid inequalities. Then, we develop different upper bounds based on either a relaxation of the objective function (leading to functions which are easier to handle than the original one) or to a relaxation of a subset of the constraints. We design both exact solution algorithms and a heuristic approach. Extensive computational results show that it is hard to devise tight dual bounds but possible to obtain good primal solutions. In particular, using (i) a black-box non-linear optimisation solver after a log-transformation of the objective function and (ii) a neighbourhood-based heuristic, we were able to generate high-quality solutions in short runtimes.

The main contributions of the paper can be summarised as follows:

1. We introduce the HOP and provide a formal definition of the problem.
2. We present a mathematical formulation of the problem which maximises a non-concave (but log-concave) objective function.
3. We propose multiple techniques to find effective upper and lower bounds, focusing on the diversity of the techniques we propose. In this introductory paper we do not further fine-tune the most powerful approaches, but instead we present promising ideas and compare both exact and heuristic methods.
4. We perform a computational study to investigate the efficacy of the bounds proposed and to gain insights into solution features.

The paper is organised as follows. Section 2 reviews the literature on closely related problems. Section 3 provides the formal description of the HOP and its mathematical formulation, while dual bounds are described in Section 4. Section 5 introduces algorithms to obtain primal solutions. Finally, Section 6 presents computational results and an analysis of solution characteristics, while we report our conclusions in Section 7.

2 Literature Review

In this section we revise the literature related to the HOP. Although the problem is new, it has aspects in common with other problems arising in contiguous areas: stochastic orienteering problems and stochastic problems with catastrophic consequences. It also shares characteristics or application areas with other problems, such as routing of hazardous materials and cash-in-transit.

Stochastic Orienteering Problems. Similar to other routing problems, real-life applications of the OP are often affected by uncertainty. As a consequence, researchers have developed stochastic variants of the OP that incorporate such uncertainty. The main stochastic OP variants are the following:

- The *OP with Stochastic Profits*, in which uncertainty affects the profit collected at each customer. This problem was introduced by İlhan, Iravani and Daskin [16] who associated a normal random distribution with each customer and aimed at maximising the probability that the total collected profit is not smaller than a given threshold. Different from the HOP, the visit order does not affect the stochastic profits of the customers.
- The *Probabilistic OP* [2, 3], in which each customer has an associated Bernoulli random variable that models whether the customer will be available. The decision-maker first builds an a priori OP tour visiting a subset of customers and respecting the time bound. In the second stage, when customer availability is revealed, the decision-maker removes unavailable customers from the tour and visits the remaining customers. The objective is to maximise the expected profit, considering that only available customers yield their corresponding profit. Different from the HOP, if a customer is not available, it is only its profit which is lost, rather than the entire collected profit.
- Uncertainty can also affect time, as in the case of the *OP with Stochastic Travel and Service Times* [5, 9, 22], in which travel and service times are uncertain, and the *OP with Stochastic Time Windows* [30], in which customers are available during stochastic time windows.

Stochastic problems with catastrophic consequences. In the HOP, if a catastrophic event happens then the whole collected profit is lost. Therefore, we can place the HOP in the category of stochastic problems in which some low-probability event has a large negative impact. Sherali et al. [27] were among the first to consider this type of problems in the context of routing of hazardous materials. In their work, different from the HOP, the authors are more concerned with assessing the external impact of the catastrophic event: for example, it is worse to spill toxic liquids in a city centre than in an unpopulated area. Thus, rather than minimising the expected probability that a catastrophic event happens, they minimise the expected impact conditional on the event happening. Other examples of problems with catastrophic consequences are the *Distributionally Robust Stochastic Knapsack Problem* [6] and the *0–1 Time-Bomb Knapsack Problem* [18]. Both are stochastic variants of the classical 0–1 Knapsack Problem. In the first problem, items arrive one by one and their weight is given by a random variable. At any point, the user packing the knapsack can either ask for a new object or stop and collect the profit currently accumulated. If the user asks for a new object and its weight is larger than the residual capacity, the knapsack breaks and all the profit accumulated is lost. In the second problem, a subset of items to pack are time-bombs which can explode with a Bernoulli-distributed probability. If any such item is packed and explodes, then the entire profit packed in the knapsack is lost. Similar to the HOP, the objective is to maximise the expected profit.

Problems with similar characteristics or applications. Finally, we describe existing problems which share some of the peculiarities of the HOP and which are used to model real-life applications in similar areas. One characteristic of the HOP is that the (expected) profit collected at a hazardous customer depends indirectly on the visit time. More precisely, the expected profit depends on the travel time after visiting the customer and therefore a hazardous customer yields a higher expected profit if the vehicle visits it towards the end of the tour (later time) compared to the beginning of the tour (earlier time). The *Team OP with Decreasing Profits* (TOPDP), introduced by Afsar and Labadie [1], also has time-dependent profits. The differences are that in the TOPDP (i) the time dependence is explicit (the profit is a linear function of the visit time), (ii) the dependence is inverse compared to the HOP (later visits yield lower profit), (iii) there is no probability of losing the entire profit in case a negative event happens, (iv) the authors consider a multi-vehicle version of the problem. Typical applications of the TOPDP are in maintenance routing, in which a customer might be willing to pay more to be visited earlier, and in humanitarian logistics or firefighting, in which visiting a target earlier increases the probability of saving lives or property. Other problems which share applications with the HOP are *cash-in-transit* problems [7]. They involve routing one or more vehicles containing cash, which are typically used to either replenish ATM machines or collect cash at stores at the end of the day. These problems have been modelled as extensions of the OP (see, e.g., [20]), although the main focus has been put on coverage and service level considerations [21] rather than on the probability that the vehicle is attacked. An exception is the case of multiperiod cash-in-transit problems, in which this risk is often explicitly considered and mitigated by generating different routes for each period [13, 14, 19]. A similar problem is the *Risk-Constrained Cash-in-Transit Vehicle Routing Problem* (RCTVRP) introduced in Talarico, Sörensen and Springael [28] where the goal is to design routes for a fleet of vehicles carrying cash considering that the probability that a vehicle is robbed is a function of both the amount of cash it carries and the distance it travels. More precisely, the authors define the risk of being attacked over an edge of length t_{ij} while carrying an amount D_i of cash as $D_i \cdot t_{ij}$. The objective function of the RCTVRP minimises the total travel cost, under a constraint that the maximum risk over any used arc is not larger than a given threshold. Under these assumptions, the authors formulate the RCTVRP as a Mixed-Integer Program, which they show to work in reasonable computing times only on small instances. The authors then develop a metaheuristic algorithm to deal with medium- and large-size instances.

3 Problem definition and mathematical formulation

Let $G = (V', A)$ be a complete directed graph with vertex $0 \in V'$ representing the depot, where the route starts and ends, while set $V = \{1, \ldots, n\} \subset V'$ represents the customer locations; set A is the arc set. Each vertex $i \in V$ has an associated profit $p_i \in \mathbb{R}^+$ and each arc $(i, j) \in A$ has a travel time $t_{ij} \in \mathbb{R}^+$. We assume that travel times satisfy the triangle inequality. We model the probability of catastrophic events with exponential distributions. As such, we consider a subset $H \subseteq V$ of hazardous vertices, and associate with each vertex $i \in H$ a parameter $\lambda_i \in \mathbb{R}^+$. A vertex $i \in H$ has an associated exponential random variable with rate λ_i modelling the possibility that the parcel collected at i explodes and causes the loss of the entire vehicle content. The cumulative distribution function of the exponential random variable associated with i is

$$F_i(t) = 1 - e^{-\lambda_i t} \quad (t \geq 0), \tag{1}$$

and it represents the probability that the catastrophic event will happen within a time $t \geq 0$ counted from the moment the vehicle visits vertex i. Finally, we denote with $T \in \mathbb{R}^+$ the upper bound on the tour travel time. The goal is to determine an elementary tour that maximises the expected collected profit and does not exceed the bound T on the duration.

3.1 A non-linear mixed-integer formulation

Consider the following decision variables:

- $x_{ij} \in \{0, 1\}$ for $(i, j) \in A$, taking value 1 iff the vehicle traverses arc (i, j).
- $y_i \in \{0, 1\}$ for $i \in V'$, taking value 1 iff the vehicle visits vertex i.
- $w_i \in \mathbb{R}^+$ for $i \in V$, representing the vehicle travel time after it leaves customer i, if i is visited, or 0 otherwise.

A non-linear mixed-integer formulation for the HOP is:

$$\max \quad \Big(\sum_{i \in V} p_i y_i\Big) \cdot \Big(\prod_{i \in H} e^{-\lambda_i w_i}\Big) \tag{2}$$

$$\begin{aligned}
\text{s.t.}\quad & y_0 = 1 && && (3)\\
& \sum_{(i,j)\in\delta^+(i)} x_{ij} = y_i && \forall i \in V' && (4)\\
& \sum_{(j,i)\in\delta^-(i)} x_{ji} = y_i && \forall i \in V' && (5)\\
& \sum_{(i,j)\in A} t_{ij}x_{ij} \le T && && (6)\\
& w_i \le My_i && \forall i \in V && (7)\\
& w_i \ge w_j + t_{ij} - M(1-x_{ij}) && \forall (i,j) \in A \ :\ i,j \ne 0 && (8)\\
& w_i \ge t_{i0}x_{i0} && \forall i \in V && (9)\\
& x_{ij} \in \{0,1\} && \forall (i,j) \in A && (10)\\
& y_i \in \{0,1\} && \forall i \in V' && (11)\\
& w_i \in \big[0, T - t_{0i}\big] && \forall i \in V, && (12)
\end{aligned}$$

where $\delta^+(i)$ denotes the set of arcs with origin i, $\delta^-(i)$ denotes the set of arcs with destination i, and M is a sufficiently large number (for example, $M = T - t_{0i}$ in (7) and $M = T - t_{j0} + t_{ij}$ in (8)). Objective function (2) maximises the expected profit under the assumption that the entire profit is lost if a parcel explodes. Constraint (3) ensures that the depot is visited, while (4) and (5) are the classical flow balance constraints. Constraint (6) enforces that the total travel time of the tour does not exceed the bound T. Constraint (7) forces w_i to be zero for vertices not visited in the tour. Constraints (8) set a bound on the value of w_i in case i is visited immediately after j. Note that these constraints exclude the case in which j is the depot and, thus, i is the last visited vertex. This latter case is considered in constraint (9). Also note that, if $x_{ij} = 1$, constraint (8) forces $w_i \ge w_j + t_{ij}$, but because higher values of w_i are penalised in the objective function, the above inequality will be enforced with strict equality in any optimal solution. Analogously, (9) is enforced with equality when $x_{i0} = 1$. Finally, we note that inequalities (8) and (9) prevent subtours. Constraints (10)–(12) define variables domain. Note that the upper bound on variable w_i in constraint (12) is valid because the latest time a customer can be visited corresponds to the time bound T minus the shortest time to reach the depot from the customer location, i.e., t_{0i}.

3.2 Valid inequalities

We propose four families of valid inequalities. The first inequality is the following:

$$w_i \ge \sum_{(i,j)\in\delta^+(i)} t_{ij}x_{ij} \quad \forall i \in V, \tag{13}$$

which ensures that the travel time after visiting i is at least as large as the travel time of the arc used to leave i, if i is visited. This inequality is particularly useful when solving the continuous relaxation of the HOP (see Section 4.4) because "big-M" constraints (8) and (9) allow fractional solutions in which variables w_i associated with visited customers take value 0.

The second inequality reads as follows:

$$w_i \le \sum_{(j,k)\in A} t_{jk}x_{jk} - t_{0i} \quad \forall i \in V. \tag{14}$$

It assumes that the triangle inequality holds and ensures that the travel time after visiting i cannot exceed the total tour travel time, minus the minimum time required to get from the depot to i (this minimum is t_{0i} and is achieved when i is the first visited vertex). This latter inequality can be extended considering the case when i is at least the *second* vertex visited after leaving the depot. Let $t'_{0i} = \min_{j\ne i}\{t_{0j} + t_{ji}\}$ be the shortest path from 0 to i when visiting at least one more vertex in between. Then the following inequality is valid:

$$w_i \le \sum_{(j,k)\in A} t_{jk}x_{jk} - t'_{0i} + Mx_{0i} \quad \forall i \in V, \tag{15}$$

where M is a sufficiently large number (e.g., $M = t'_{0i} - t_{0i}$). Note that constraint (15) becomes moot when i is the first vertex visited after leaving the depot. We also remark that the last two valid inequalities, which bound w_i from above, can be omitted for customers $i \in H$. For these customers, in fact, the objective function already penalises the corresponding w_i.

We also include the generalised subtour elimination constraints (GSECs)

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq y_k \quad \forall S \subseteq V, \ \forall k \in S \tag{16}$$

as valid inequalities. Although subtours are already prevented by inequalities (8) and (9), these "Miller-Tucker-Zemlin-like" constraints give notoriously poor continuous relaxation bounds (see [4]). GSECs, on the other hand, give better bounds but are impractical to enumerate because their number grows exponentially in the instance size. Therefore, we initially do not include cuts (16) in the model and we dynamically separate them, as explained below.

To detect the GSECs violated by a fractional solution $\left(x^*_{ij}, y^*_i\right)$, we solve several max-flow/min-cut problems extending the "GSECs" separation procedure of Fischetti, Gonzalez and Toth [11]. We build an auxiliary graph $G^* = (V^*, A^*)$, with $V^* = \left\{i \in V' \ : \ y^*_i > 0\right\}$ and $A^* = \left\{(i,j) \mid i,j \in V^*, \ i \neq j\right\}$. We associate with each arc $(i,j) \in A^*$ a *capacity* equal to x^*_{ij}. The separation procedure considers vertices $i \in V^* \setminus \{0\}$ in decreasing order of value of y^*_i. For each vertex, it finds a minimum-capacity $(0,i)$-cut. Let $S_i, V^* \setminus S_i$ be such cut, where $i \in S_i$ and $0 \notin S_i$. If the corresponding maximum flow between 0 and i is smaller than y^*_i, then the solution violates a GSEC and we add to the model the constraint associated with S_i. Similar to what Fischetti, Gonzalez and Toth [11] proposed for the OP, a simple way to avoid generating the same set S_i twice is to increase the capacity of arc $(0,i)$ by $1 - \sum_{j \notin S_i} \sum_{k \in S_i} x_{jk}$ before moving to the next vertex.

4 Upper bounds for the HOP

In this section, we propose upper bounds for the HOP based on either the relaxation of the objective function or of a subset of the constraints. We also present an upper bound obtained from dynamic programming.

4.1 Local hazard bound

The first bound stems from the following observation. If, instead of destroying the entire vehicle content, the explosion of a parcel only caused the loss of the parcel itself, the corresponding expected profit would overestimate the expected profit of the HOP. Therefore, the first upper bound is based on assuming that, when a catastrophic event caused by a visit to $i \in V$ occurs, the planner only loses the profit associated with i. This corresponds to changing objective function (2) as follows:

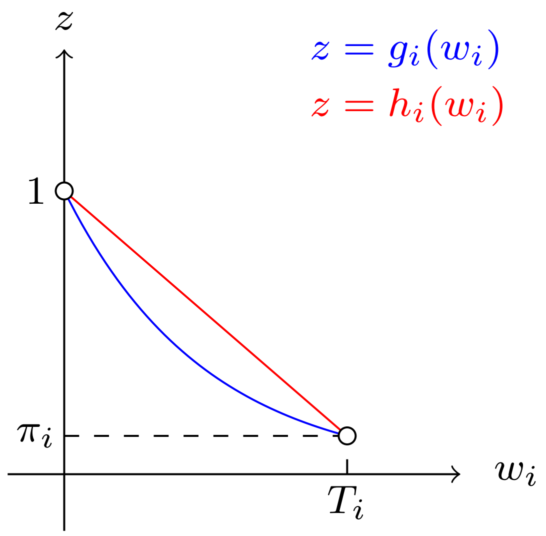
$$\sum_{i \in V} p_i y_i e^{-\lambda_i w_i}, \tag{17}$$

with the convention that $\lambda_i = 0$ if $i \notin H$. This bound, *per se*, does not dramatically simplify the objective function; for example, (17) is still non-linear. However, we will use (17) in the rest of this section to devise linear upper bounds.

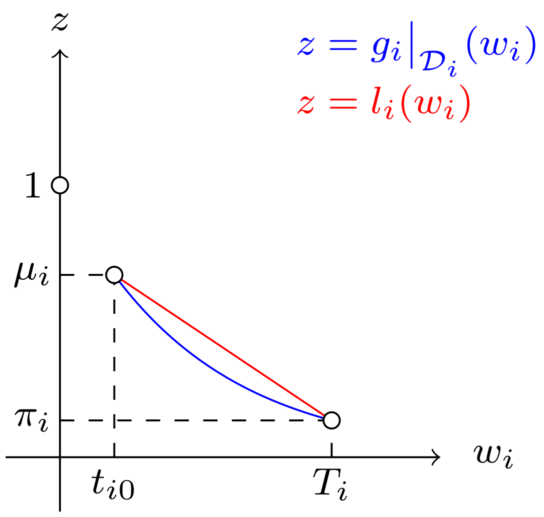
4.2 Linear approximation

The product of exponential functions in objective (2) can be bounded from above by a product of linear functions. Let $g_i(w_i) = e^{-\lambda_i w_i}$. When considering g_i as a function $[0, T - t_{0i}] \to \mathbb{R}$, it is easy to show that g_i is bounded from above by the linear function

$$h_i(w_i) = \frac{\pi_i - 1}{T_i} w_i + 1, \tag{18}$$



(a) Relation between g_i and h_i.



(b) Relation between $g_i\big|_{\mathcal{D}_i}$ and l_i.

Figure 1: Linear and piecewise-linear approximations overestimating of the original objective function.

where $T_i = T - t_{0i}$ and $\pi_i = e^{-\lambda_i T_i}$. Figure 1a depicts the relation between g_i and h_i. Each function h_i overestimates the corresponding g_i and, thus, objective function

$$\Big(\sum_{i\in V} p_i y_i\Big) \cdot \Big(\prod_{i\in H} \big(\frac{\pi_i - 1}{T_i} w_i + 1\big)\Big) \tag{19}$$

is an upper bound for (2).

Combining (18) with (17) we obtain a quadratic relaxation of the original problem with objective function

$$\sum_{i\in V} p_i y_i \Big(\frac{\pi_i - 1}{T_i} w_i + 1\Big). \tag{20}$$

Because $y_i w_i = w_i$ and $\pi_i = 1$ for $i \notin H$, we can write (20) as a linear objective function:

$$\sum_{i\in H} p_i \frac{\pi_i - 1}{T_i} w_i + \sum_{i\in V} p_i y_i. \tag{21}$$

Using (21) we can solve a mixed-integer linear programme to obtain an upper bound on the original non-linear problem.

4.3 Piecewise linear approximation

We can tighten the linear approximation presented in Section 4.2 if we note that the domain of variables w_i is not the entire interval $[0, T_i]$, but rather the disconnected set $\mathcal{D}_i = \{0\} \cup [t_{i0}, T_i]$. Thus, functions g_i restricted to domain $\mathcal{D}_i$ are bounded from above by functions l_i defined as

$$l_i(w_i) = \begin{cases} 1 & \text{if } w_i = 0 \\ \frac{\mu_i - \pi_i}{t_{i0} - T_i} w_i + \pi_i - \frac{T_i(\mu_i - \pi_i)}{t_{i0} - T_i} := \alpha_i w_i + \beta_i & \text{if } w_i \in [t_{i0}, T_i] \end{cases}, \tag{22}$$

where $\mu_i = e^{-\lambda_i t_{i0}}$. Figure 1b depicts the relation between $g_i\big|_{\mathcal{D}_i}$ and l_i.

Each function l_i overestimates the corresponding function $g_i\big|_{\mathcal{D}_i}$ and thus, by the fact that $w_i = 0 \iff y_i = 0$, function

$$\big(\sum_{i\in V} p_i y_i\big) \cdot \Big(\prod_{i\in H} \big(y_i(\alpha_i w_i + \beta_i - 1) + 1\big)\Big) \tag{23}$$

is an upper bound for (2). Combining (23) with (17) we obtain a polynomial relaxation of the original problem with objective function

$$\sum_{i\in V} p_i\beta_i y_i\big(y_i(\alpha_i w_i + \beta_i - 1) + 1\big) \tag{24}$$

and, using the fact that $y_i^2 = y_i$, $y_i w_i = w_i$, and $\alpha_i = 0$ for $i \notin H$, we obtain the linear objective function

$$\sum_{i\in V} p_i\beta_i y_i + \sum_{i\in H} p_i\alpha_i w_i. \tag{25}$$

Using (25), we can solve a mixed-integer linear program to obtain an upper bound on the original problem and, because $l_i(w_i) \leq h_i(w_i)$ (for all $i \in H$), such bound is tighter than the one derived using objective function (21).

4.4 Continuous relaxation

Denote with $X = (\vec{x}, \vec{y}, \vec{w})$ the vector of all variables, and with $b(X)$ objective function (2). The continuous relaxation of model (2)–(12) asks to maximise b subject to constraints (3)–(9), but with the variable domain definitions replaced by:

$$\begin{aligned} 0 \leq x_{ij} &\leq 1 \quad \forall (i,j) \in A &(26)\\ 0 \leq y_i &\leq 1 \quad \forall i \in V' &(27)\\ 0 \leq w_i &\leq T_i \quad \forall i \in V, &(28)\end{aligned}$$

where T_i is as defined in Section 4.2.

Unfortunately, b is neither convex nor concave (as it follows immediately from $B(u, v) = u \cdot e^v$ being neither convex nor concave). However, b is log-concave and by maximising the logarithm of (2) one gets the same maximiser as for the original problem. We then propose to solve the continuous optimisation problem of minimising $f(X) := -\ln b(X)$ subject to (3)–(9),(13)–(16),(26)–(28); we include valid inequalities (13)–(16) to benefit from a stronger continuous relaxation. An explicit formula for f is

$$f(X) = -\ln\left(\sum_{i\in V} p_i y_i\right) + \sum_{i\in H} \lambda_i w_i.$$

Because both the feasible region of this problem (which we denote as P) and objective function f are convex, we can use the Frank-Wolfe algorithm [12] to find a minimum of f. In the following, we detail the procedure used to minimise f.

The Frank-Wolfe algorithm starts with a feasible solution $\bar{X}$ and gets an improving solution $\bar{X}'$ solving the auxiliary problem

$$\bar{X}' = \arg\min\big\{\nabla f(\bar{X})^\top X \mid X \in P\big\}. \tag{29}$$

Solution $\bar{X}$ is a local (and, by convexity of f, global) optimum if $\nabla f(\bar{X})^\top(\bar{X}' - \bar{X}) \leq 0$. Otherwise, the next solution is obtained by minimising f over the segment joining $\bar{X}$ and $\bar{X}'$:

$$\bar{X} \leftarrow \arg\min\big\{f(X) \mid X \in [\bar{X}, \bar{X}']\big\}. \tag{30}$$

We can solve (30) with a line search, minimising the one-dimensional function which maps $\gamma \mapsto f\big(\gamma\bar{X} + (1-\gamma)\bar{X}'\big)$. Such problem has closed-form solution

$$\gamma^* = \frac{\frac{\sum_{i\in V} p_i(\bar{y}_i - \bar{y}'_i)}{\sum_{i\in H} \lambda_i(\bar{w}_i - \bar{w}'_i)} - \sum_{i\in V} p_i\bar{y}'_i}{\sum_{i\in V} p_i(\bar{y}_i - \bar{y}'_i)}, \tag{31}$$

where $\bar{y}_j$ and $\bar{w}_j$ are, respectively, the values of variables y_j and w_j in solution $\bar{X}$ (and analogously for $\bar{y}'_j$ and $\bar{w}'_j$ in solution $\bar{X}'$).

As for the auxiliary problem (29), it is a linear problem (LP) with feasible region P and objective function

$$\min \sum_{i \in V} \frac{\partial f}{\partial y_i}(\bar{X}) \cdot y_i + \sum_{i \in H} \frac{\partial f}{\partial w_i}(\bar{X}) \cdot w_i =$$
$$\min \sum_{i \in V} \frac{-p_i}{\sum_{j \in V} p_j \bar{y}_j} y_i + \sum_{i \in H} \lambda_i w_i. \tag{32}$$

4.5 Non-elementary tour

We obtain an additional upper bound by dropping the elementarity requirement and allowing the same customer to be visited multiple times. To obtain the optimal solution to this relaxed problem, we use the state-space-relaxed Dynamic Programming algorithm described in Section 5.1.1.

5 Heuristic solutions

We present two algorithms to build feasible solutions for the HOP. The first is an exact method based on dynamic programming. In practice, the runtime of this algorithm is too high to use it as an exact solution method. However, one can use it as a heuristic stopping its execution and using the best solution found. The second algorithm is a heuristic based on a concept similar to that of the *Adaptive Large Neighbourhood Search* of Ropke and Pisinger [23].

We also note that, after solving the MILPs required by the linear and piecewise-linear bounds presented in section 4.2 and 4.3, their solutions are feasible for the HOP. Therefore, by computing the original objective function (2) using variables y and w from the optimal (or any feasible) solution of these MILPs, we obtain valid lower bounds.

5.1 Dynamic Programming

We propose a labelling algorithm for the HOP which associates a label with each partial path from a vertex to the depot. Extending such partial paths "backwards" until the first visited vertex is the depot itself (while ensuring that no constraint is violated) produces a valid HOP tour. The number of such tours to consider is limited by pruning unpromising labels via dominance rules as explained below.

Label definition. We denote a label as a tuple $L = (v, W, p, \eta, \pi, t)$, where: $v \in V$ is the current starting vertex of the partial path; $W \in \{0,1\}^{|V'|}$ is a binary vector whose i-th component is 1 iff the partial path visits vertex $i \in V'$; $p \in \mathbb{R}_0^+$ denotes the total profit accumulated at the customers visited by the partial path; $\eta \in (0,1]$ is the probability that no loaded parcel explodes while travelling along the partial path; $\pi = p \cdot \eta$ is the expected profit accumulated along the partial path; $t \in \mathbb{R}_0^+$ denotes the travel time of the partial path. The set of labels is initialised with the single label $L_0 = (0, \vec{0}, 0, 1, 0, 0)$.

Label extension. A label $L = (v, W, p, \eta, \pi, t)$ is extended to a vertex $i \in V'$ if $W_i = 0$, i.e., the path has not visited customer i, and $t_{0i} + t_{iv} + t \leq T$, i.e., visiting i would not violate the travel time bound. If the extension is feasible, label L is extended to label $L' = (v', W', p', \eta', \pi', t')$, where:

$$v' = i, \quad W'_j = \begin{cases} W_j & \text{if } j \neq i \\ 1 & \text{if } j = i \end{cases}, \quad p' = p + p_i, \quad \eta' = \eta e^{-\lambda_i (t_{iv} + t)}, \quad \pi' = p'\eta', \quad t' = t_{iv} + t.$$

Label dominance. We say that label L_1 dominates label L_2 if any feasible completion of the partial path associated with L_1 is feasible for L_2 and this latter has a worse objective value. In this case, we shall not extend L_2, as the optimal solution cannot contain the path associated with L_2 as a sub-path. In our problem, L_1 dominates L_2 if: (i) $v_1 = v_2$, i.e., both labels refer to a partial path starting at a same vertex; (ii) $W_1 \leq W_2$ component-wise, i.e., the partial path associated with L_1 visits a subset of the vertices visited by the partial path associated with L_2; (iii) $p_1 \geq p_2$, i.e., the profit collected by

L_1 is not lower than the profit collected by L_2; (iv) $\eta_1 \geq \eta_2$, i.e., L_1 has a probability of *not* exploding which is not lower than that of L_2; (v) $t_1 \leq t_2$, i.e., the travel time of L_1 does not exceed that of L_2; (vi) at least one of the inequalities at points (ii)–(v) is strict. Note that $W_1 \leq W_2$ and $p_1 \geq p_2$ imply that $W_1 = W_2$ and $p_1 = p_2$. Therefore, $\eta_1 \geq \eta_2$ also implies $\pi_1 \geq \pi_2$.

Optimal path. We recover the optimal path choosing the label with the highest objective value π, among all undominated labels which are extended up to the depot 0. Keeping, for each label L, a pointer to the label from which L was extended, allows us to recover the entire path.

Heuristic. We note that any partial path can be converted into a complete solution by connecting the depot to the head of the path. As such, if we take care to always perform such a connection as the first one tried when extending a label, we can build a pool of primal feasible solutions very early during the run of the algorithm. Therefore, we are able to return a feasible solution at any moment (for example, when we reach a maximum runtime), by choosing the best label which was extended up to the depot.

5.1.1 State-space relaxation

We have explained how to use the proposed labelling algorithm to produce primal solutions either as an exact algorithm or as a heuristic. We conclude this section showing that the same algorithm can also be used to compute upper (dual) bounds, using the relaxation technique described below.

State-space relaxation (SSR) is a technique introduced by Christofides, Mingozzi and Toth [8] to reduce the number of labels generated by a labelling algorithm. SSR projects the space of all possible labels (say, $\mathcal{S}$) into a lower-dimensional space (say, $\mathcal{S}'$), and runs the algorithm in this reduced space. However, during the reduction, the original labels corresponding to both feasible and infeasible partial paths are projected to $\mathcal{S}'$. Therefore, the optimal label in the reduced space might correspond to an infeasible tour. In this case, the algorithm would provide an upper bound on the value of the optimal solution.

In our case, the projection maps vector W to the sum of its elements $\Sigma = \sum_{j \in V'} W_j$. In this way, labels cannot keep track of which customers have been visited and their partial paths can include multiple visits to the same customer.

In the label extension checks, condition $W_i = 0$ is replaced by two checks: $j \neq v$, i.e., we are not "staying" at the same customer, and $\Sigma < n + 1$, i.e., we are not visiting more vertices than there are in the graph. The new component is updated as $\Sigma' = \Sigma + 1$ and dominance condition (ii) is replaced by $\Sigma_1 \leq \Sigma_2$.

Remark. *It might be tempting, when using SSR, to replace dominance conditions (iii)* $p_1 \geq p_2$ *and (iv)* $\eta_1 \geq \eta_2$ *with the single condition* $p_1\eta_1 = \pi_1 \geq \pi_2 = p_2\eta_2$*. This choice would allow to dominate more labels, because* π_1 *can be larger than* π_2 *without both* p_1 *and* η_1 *being, respectively, larger than* p_2 *and* η_2*. However, this check would lead to an incorrect dominance rule. In fact, it is possible that* $\pi_1 > \pi_2$ *but, when extending both* L_1 *and* L_2 *to the same vertex* j*,* $\pi'_1 = \pi'_2$*. In case* $\pi_1 > \pi_2$ *was the only inequality satisfied strictly, we would have that* L_1 *dominates* L_2 *but extension* L'_1 *does not dominate* L'_2*.*

This situation, although rare, can happen; Figure 2 shows an example with a graph and the partial path associated with two labels (in blue and red, respectively). Consider label $L_1 = (v_1 = 3, \Sigma_1 = 2, p_1 = 20, \eta_1 = 0.05, \pi_1 = 1, t_1 = 31)$ *and* $L_2 = (v_2 = 3, \Sigma_2 = 2, p_2 = 1, \eta_2 = 0.1, \pi_2 = 0.1, t_2 = 31)$*. If we do not consider* p *and* η *in the dominance, but only consider* π*,* L_1 *would dominate* L_2 *and the only strict inequality would be* $\pi_1 > \pi_2$ *(*Σ *and* t *being equal). If we now extend both labels to a new non-hazardous vertex* 4 *with* $p_4 = 18$*,* $\lambda_4 = 0$ *and* $t_{43} = 1$*, we would obtain:* $L'_1 = (v'_1 = 4, \Sigma'_1 = 3, p'_1 = 38, \eta'_1 = 0.05, \pi'_1 = 1.9, t'_1 = 32)$ *and* $L'_2 = (v'_2 = 4, \Sigma'_2 = 3, p'_2 = 19, \eta'_2 = 0.1, \pi'_2 = 1.9, t'_2 = 32)$*. Because all tested components are equal,* L'_1 *does not dominate* L'_2*, invalidating the validity of the "simplified" dominance criterion.*

We strengthen the SSR version of the algorithm by explicitly forbidding 2-cycles (see, e.g., [15]). To this end, we add to our label a component $u \in V'$ indicating the vertex which, in the partial path, is

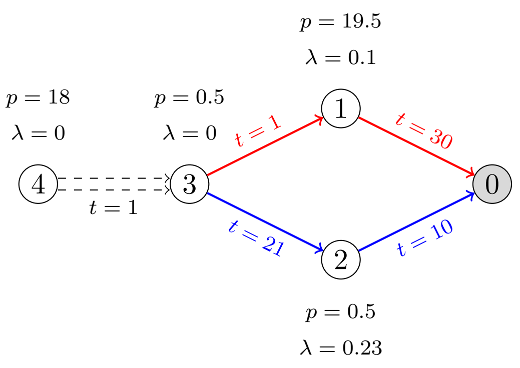


Figure 2: Sample instance showing that it is necessary to use both p and η in the dominance criterion of state-space relaxed labels, because comparing π can lead to wrongly excluding valid labels.

visited immediately after the current vertex v. We forbid extending a label L to a vertex j if $u = j$. When L is extended to a new label L', we set $u' = v$. Finally, when checking dominance between labels L_1 and L_2, we add the additional condition that $u_1 = u_2$. This version of the algorithm allows fewer labels to dominate each other, but produces a considerably tighter upper bound on the objective value of the optimal solution.

5.2 Adaptive Neighbourhood Search

We propose a primal heuristic for the HOP based on the application of several neighbourhoods, similar to the popular Adaptive Large Neighbourhood Search (ALNS) metaheuristic framework [23]. Unlike standard ALNS, our neighbourhoods are not necessarily large and are defined explicitly, while in ALNS they are implicitly defined by the successive application of *destroy* and *repair* moves. We call the approach Adaptive Neighborhood Search (ANS). It is summarised in Algorithm 1.

We propose three classes of neighbourhoods and, within each class, we use neighbourhoods which favour diversification and others which favour intensification. The three classes are insertion, removal and swap. The insertion class attempts to insert new customers within the current tour (note that, different from the OP, inserting a customer does not necessarily improves the objective value). The removal class attempts to remove customers from the current tour (again, removing a customer does not necessarily worsen the objective value). The swap class attempts to swap the visit order of two customers already visited by the current tour. We allow neighbourhoods to create solutions violating the time bound, but we give their objective value a large negative penalty. The neighbourhoods that we use are the following:

Insertion. Within the insertion class, we use the following:

1. **Insert best customer at best position.** It cycles through all customers not included in the tour and, for each of them, it attempts to insert it in all possible positions of the tour. At the end, it chooses the customer-position pair that increases the objective value the most (or that decreases it the least, if no insertion is improving).
2. **Insert random customer at best position.** It first select a random customer among those which are not part of the tour. It then inserts it in the position that increases the objective value the most (or decreases it the least).
3. **Insert random customer at random position.** It selects a random customer among those which are not part of the tour and inserts it in a random position.

Removal. The removal class contains the following neighbourhoods:

1. **Remove best.** Removes from the tour the customer which results in the highest objective value increase (or in the smallest decrease).
2. **Remove best or random.** Removes the customer which results in the highest objective value increase. If no such customer exists (i.e., removing any customer decreases the objective

Algorithm 1 ANS heuristic for the HOP.

```
1: $x \leftarrow$ RandomSolution()                                   ▷ current solution
2: $x^* \leftarrow x$                                                ▷ best solution
3: $i \leftarrow 1$                                                  ▷ iteration counter
4: $\mathcal{N} \leftarrow$ (neighbourhoods)                         ▷ list of neighbourhoods
5: $S \leftarrow \vec{1}$                                            ▷ list of neighbourhood scores

6: while $i \leq \pi^{\text{iter}}$ do
7:     $i \leftarrow i + 1$
8:     $N \leftarrow$ RouletteWheel($\mathcal{N}, S$)    ▷ select with probability proportional to the scores
9:     $x' \leftarrow N(x)$                                          ▷ new solution
10:    $s' \leftarrow 1$                 ▷ score multiplier for $N$ depending on the quality of $x'$

11:    if Accept($x^*, x'$) then
12:        $s' \leftarrow \pi^{\text{sa}}$                           ▷ score for accepted solution

13:        if $\phi(x') > \phi(x)$ then
14:            $s' \leftarrow \pi^{\text{sic}}$                      ▷ score for improving on the current solution

15:            if $\phi(x') > \phi(x^*)$ then
16:                $s' \leftarrow \pi^{\text{sib}}$                  ▷ score for improving on the best solution
17:                $x^* \leftarrow x'$                               ▷ update best solution
18:            end if
19:        end if

20:        $x \leftarrow x'$                                         ▷ update current solution
21:    end if

22:    $S[N] \leftarrow s' \cdot S[N]$                               ▷ update the score associated with $N$

23:    if $i \equiv 0 \mod \pi^{\text{ls}}$ then
24:        $x \leftarrow$LocalSearch($x$)                            ▷ perform local search on the current solution

25:        if $\phi(x) > \phi(x^*)$ then
26:            $x^* \leftarrow x$                                    ▷ update best solution
27:        end if
28:    end if

29:    if $x^*$ not updated for more than $\pi^{\text{id}}$ iterations then
30:        if Bernoulli(0.5) $< 0.5$ then
31:            $x \leftarrow$RandomSolution()                        ▷ diversification: restart
32:        else
33:            $x \leftarrow x^*$                                    ▷ intensification
34:        end if
35:    end if
36: end while
    return $x^*$
```

value), it removes a random customer.

3. **Remove random (1).** Removes a customer at random.
4. **Remove random (3).** Removes three customers at random. If the tour contains fewer than three customers, it removes all customers. Note that we jump from 1 to 3 in the size of the remove operator as this gives a higher diversification rate than going from 1 to 2, but it also preserves a good part of the solution.

Swap. In the following, we call *time-bomb* (TB) the customers of set H and *deterministic* (DET) those of set $V \setminus H$. We use the following swap neighbourhoods:

1. **Swap TB-DET.** It first checks if the tour visits any time-bomb customer before any deterministic customer. If this is not the case, then this operator does nothing. Otherwise, it tries to swap the position of all TB-DET customer pairs which satisfy the above condition. It performs the swap which increases the objective value the most (or decreases it the least). We note that this neighbourhood is particularly effective when the current solution is a random one (see lines 1 and 31 in Algorithm 1).
2. **Swap TB-TB.** It similar to the previous neighbourhood, but it looks for pairs of time-bomb customers in which the one visited earlier has a higher value of λ than the one visited later.
3. **Swap DET-DET.** It goes through all the pairs of deterministic customers and performs the swap that improves a hierarchical objective the most (or that worsens it the least). The hierarchical objective favours first the tour with the highest original objective value and, in case of draws, the tour with the lowest travel time. We use such a criterion because if a tour only visits deterministic customers then no swap can change the original objective value.
4. **Swap random-random.** It swaps two customers at random.

We now describe the other components of Algorithm 1.

Acceptance. Following the advice of Santini, Ropke and Hvattum [26], we use a linear record-to-record travel acceptance criterion. Using this method, we accept x' if

$$\frac{\phi(x^*) - \phi(x')}{\phi(x^*)} < T,$$

where $\phi(x)$ denotes the fitness of a solution x, calculated as its objective value, minus a large constant penalty in case the tour time exceeds the time bound. Value T is a threshold, which varies linearly from π^{start} to π^{end} while the algorithm iteration increases.

Neighbourhood scores. Each neighbourhood $N \in \mathcal{N}$ has an associated score $S[N]$ and initially all neighbourhoods have score 1. At each iteration, a neighbourhood is chosen at random with a probability proportional to its score (so-called roulette-wheel selection). At the end of the iteration, the score of a neighbourhood is multiplied by a value s'. This value changes depending on the performance of the neighbourhood during the current iteration: (i) if the new solution x' improved on the best x^*, then s' is set to a large value π^{sib}; (ii) otherwise, if x' improved on the current solution x, s' is set to value π^{sic}; (iii) otherwise, if the new solution was still accepted by the acceptance criterion, s' is set to value π^{sa}; (iv) finally, if the new solution was discarded, $s' = 1$ and the score is not updated.

Local Search. Every π^{ls} iterations, the current solution undergoes local search. The local search operator returns the best solution that can be obtained from x by either removing or inserting other customers. To do so it solves a restricted versions of model (2)–(12). Specifically, in the case of customers' removal, the operator works as follows: (i) it fixes to 0 variables y_i and w_i unless i is visited in x, and variables x_{ij} unless both i and j are visited in x; (ii) for all customers visited in x, it bounds from above the value of variables w_i, using the travel time after visiting customer i in solution x (because removing a customer cannot increase the travel time after any vertex); (iii) it adds the following constraint to preserve the visit order among customers:

$$w_{i_k} \geq w_{i_h} - M\big(2 - y_{i_k} - y_{i_h}\big) \quad \forall k \in \{1, \ldots, \ell - 1\},\ \forall h \in \{k+1, \ldots, \ell\},$$

where $(i_1, \ldots, i_\ell)$ is the ordered list of customers visited by solution x and M is a sufficiently large number (e.g., $M = T - t_{0 i_h}$). This constraint ensures that customers which are kept in the tour are visited in the same order as in x.

In the case of customers' insertion, the operator works as follows:: (i) it fixes to 1 variables y_i associated with customers visited in x and to 0 variables x_{ij} such that both i and j are visited in x, but j is not visited immediately after i; (ii) it bounds from below the value of variables w_i associated with customers visited in x, using the travel time after visiting customer i in solution x (because adding customers to the tour cannot decrease the travel time after any previously visited vertex); (iii) it adds the following constraint to preserve the visit order among customers: $w_{i_k} \geq w_{i_{k+1}}$ for all $k \in \{1, \ldots, \ell - 1\}$. This constraint ensures that the visit sequence of customers already in x does not change even when new customers are inserted among them.

Diversification/Intensification. If the best solution is not improved for π^{id} iterations, the algorithm then starts either a diversification or an intensification phase. Each of these two options has a 50% chance of being selected. To intensify the search, we set the current solution equal to the best solution. To diversify, we set the current solution equal to a random tour (i.e., we perform a restart).

6 Computational results

In this section we present the results of our computational tests using a machine equipped with 4GB RAM and an Intel Xeon CPU running at 2.4GHz. All algorithms were coded using the Python programming language, version 3.8. We used Gurobi 9.0.0 as the black-box MIP solver for the linear and piecewise-linear bounds and as the LP solver for auxiliary problem (29). We used Baron 22.1 as the black-box non-linear solver for model (2)–(12).

The instance generator, instance files, solvers, results files and all the scripts used to generate tables and figures in this paper are available on-line [25].

6.1 Instances

We generate HOP instances starting from the popular Tsiligirides OP instances [29], a set of 49 instances divided into three groups. Instances of each of the three groups contain, respectively, 20, 31 and 32 vertices. To transform OP instances into HOP instances, we use the following generation method. For each base instance, we first choose the number of customers that will be hazardous as $n \cdot \alpha^{\text{Tsi}}$, where n is the number of customers in the instance and $\alpha^{\text{Tsi}} \in \{0.1, 0.2, 0.3, 0.4\}$ is a parameter. The hazardous customers are randomly chosen among all customers. Next, we assign a value for the exponential distribution parameter λ_i to each hazardous customer. In our generation process, λ_i is chosen uniformly at random between 0.05 and 0.1. Finally, to make hazardous customers more attractive, we multiply their original profit by a parameter $\beta^{\text{Tsi}} \in \{2, 3, 4, 5\}$.

Following the above procedure, we generate 784 instances (49 Tsiligirides OP instances $\times$ 4 values of α^{Tsi} $\times$ 4 values of β^{Tsi}).

6.2 Upper bounds

We first analyse the quality of the upper bounds proposed in Section 4. We present the results of our analysis in Table 1. For a given instance and an upper bound UB, we compute the gap (columns "Gap%") as

$$\text{Gap\%} = 100 \cdot \frac{\text{UB} - \text{BKS}}{\text{UB}},$$

where BKS is the best-known solution for the instance, obtained by taking the best integer feasible solution found in all experiments for that instance. Columns "T (s)" report the runtime needed to compute the bound, in seconds. The values in each row are averaged over all instances generated using the corresponding values of α^{Tsi} and β^{Tsi} (first two columns).

		UB-Lin		UB-PwLin		UB-Cont		UB-SSR			UB-SSR-2CE		
α^{Tsi}	β^{Tsi}	Gap%	T (s)	Gap%	T (s)	Gap%	T (s)	Val%	Gap%	T (s)	Val%	Gap%	T (s)
0.10	2	11.30	42.6	8.17	41.9	13.32	1.3	28.57	34.58	2742.3	20.41	13.81	3020.8
0.10	3	18.20	48.4	13.41	46.6	19.70	1.1	28.57	29.83	2722.1	20.41	14.61	3035.8
0.10	4	23.44	48.5	17.24	47.1	22.80	1.2	28.57	28.79	2772.4	18.37	10.63	3015.6
0.10	5	25.28	48.8	19.05	47.2	21.76	1.2	28.57	31.86	2769.0	16.33	11.45	3046.9
0.20	2	24.82	52.9	18.23	52.2	19.04	1.2	22.45	35.30	2893.3	16.33	16.05	3067.6
0.20	3	34.93	55.7	27.09	53.9	16.46	1.2	20.41	30.94	2964.5	14.29	14.66	3126.5
0.20	4	40.34	57.9	32.23	57.0	26.06	1.3	24.49	25.87	2866.6	16.33	8.56	3094.0
0.20	5	44.22	56.6	35.64	56.2	26.16	1.3	24.49	26.48	2876.1	14.29	8.28	3112.4
0.30	2	33.51	53.9	25.38	53.6	17.83	1.4	22.45	29.69	2975.2	16.33	10.15	3075.1
0.30	3	44.56	57.0	36.15	55.9	19.71	1.6	20.41	30.74	2978.2	14.29	9.80	3125.2
0.30	4	50.80	57.1	42.13	55.9	21.60	1.4	24.49	25.40	2915.6	14.29	10.73	3117.5
0.30	5	56.78	58.9	47.99	58.4	25.66	1.6	20.41	23.72	2990.2	14.29	7.41	3105.1
0.40	2	43.77	55.7	35.56	55.2	22.97	1.2	22.45	33.27	2941.4	16.33	10.45	3080.6
0.40	3	54.48	58.2	45.86	58.0	19.86	1.9	22.45	28.00	2950.5	18.37	6.81	3090.8
0.40	4	58.68	60.6	50.19	59.3	22.82	1.7	26.53	21.47	2826.6	16.33	8.14	3102.0
0.40	5	62.30	56.4	54.50	56.4	25.98	1.9	22.45	23.10	2933.4	14.29	8.86	3126.0
Overall		39.21	54.3	31.80	53.4	21.36	1.4	24.23	28.73	2882.3	16.33	10.77	3083.9

Table 1: Comparison of the five upper bounds. Columns "Gap%" report the optimality gap with the best-known primal solution. Columns "T (s)" report the time in seconds. Columns "Val%" give the percentage of instances for which a valid upper bound was produced in dynamic programming based bounds.

We consider the following upper bounds:

1. UB-Lin: bound obtained through the linear approximation of the objective function (see Section 4.2). Formulation (3)–(12), enriched with inequalities (13)–(15), is solved with Gurobi. Violated constraints (16) are separated on fractional and integer solutions alike. Because the model always solved to optimality within the time limit (3600s), we used the value of its optimal solution as an upper bound. However, we note that if the model would have timed out, we could have used the best upper bound returned by the solver as an upper bound for the HOP.

2. UB-PwLin. Similar to bound UB-Lin, but we use the piecewise-linear approximation of the objective function (see Section 4.3).

3. UB-Cont. The upper bound from the optimal solution of the continuous relaxation. It is calculated using the Franke-Wolfe algorithm on the convexified problem in which we replace the objective function with its negative logarithm (see Section 4.4). Once the optimal solution is returned, in terms of variables y and w, we compute the original objective function and use the corresponding value as the bound.

4. UB-SSR. The bound obtained removing the elementarity requirement, and using the corresponding state-space-relaxed (SSR) dynamic programming algorithm (see Section 5.1.1). In these experiments, we did not use the strengthening which forbids 2-cycles. The algorithm provides valid upper bound only if it finishes within the time limit (an early stop provides an invalid upper bound). However, the SSR turns out to be slow: column "Val%" in Table 1 reports the percentage of instances for which the algorithm completed within the limit and, thus, produced a valid upper bound. Then, in column "Gap%" the average is computed over these instances. Column "T (s)" reports the average time over all instances.

5. UB-SSR-2CE. Similar to bound UB-SSR, but using the strengthening which forbids 2-cycles.

Table 1 shows that finding dual bounds for the HOP is hard. The best method, among those that always provide a valid bound, is to solve the continuous relaxation. The average gap provided by UB-Cont is 19.10% and the average solution time is 1.5s. Dynamic programming bounds are too computationally expensive: even if UB-SSR-2CE's gaps are smaller (10.77%), this method only yields a valid bound for 16.33% of the instances and takes a much longer time compared to UB-Cont. Regarding the bounds coming from the relaxation of the objective function, as expected UB-PwLin gives lower gaps than UB-Lin (objective (25) is tighter than objective (21)), while the runtimes are comparable.

6.3 Lower bounds

We now investigate the quality of primal feasible solutions found by the algorithms described in Section 5. For a given instance and a given lower bound LB, we report two values:

$$G_1\% = 100 \cdot \frac{\text{BKS} - \text{LB}}{\text{BKS}}, \quad G_2\% = 100 \cdot \frac{\text{UB}_\text{c} - \text{LB}}{\text{UB}_\text{c}},$$

where UB_c is the value of the upper bound UB-Cont. Thus, G_1 provides a gap with the best known solution and is influenced by the quality of the primal solutions found. G_2, on the other hand, provides a gap with a chosen upper bound.

		LB-NLModel			LB-NLCModel			LB-DP			LB-ANS		
α^{Tsi}	β^{Tsi}	$\text{G}_1\%$	$\text{G}_2\%$	T (s)	$\text{G}_1\%$	$\text{G}_2\%$	T (s)	$\text{G}_1\%$	$\text{G}_2\%$	T (s)	$\text{G}_1\%$	$\text{G}_2\%$	T (s)
0.10	2	0.00	13.32	118.5	0.00	13.32	102.1	47.14	53.42	3177.7	1.70	14.79	359.1
0.10	3	0.09	19.78	188.4	0.00	19.70	129.5	45.47	55.58	3188.4	0.68	20.24	347.0
0.10	4	0.00	22.80	229.5	0.00	22.80	155.8	45.08	57.17	3182.6	1.60	23.99	323.9
0.10	5	0.00	21.76	413.6	0.00	21.76	192.8	44.23	55.89	3183.4	1.04	22.65	336.5
0.20	2	0.00	19.04	484.9	0.00	19.04	193.3	46.52	56.47	3185.7	0.84	19.67	324.0
0.20	3	7.45	23.02	1174.4	0.00	16.46	245.7	46.66	55.49	3190.9	1.82	17.84	309.9
0.20	4	10.17	32.48	1387.7	0.00	26.06	507.9	47.63	60.54	3184.7	1.30	27.04	339.3
0.20	5	20.56	40.81	1881.4	0.00	26.16	651.5	46.40	59.88	3182.8	1.16	26.92	351.3
0.30	2	15.82	32.53	1461.2	0.00	17.83	105.8	50.18	59.57	3188.8	1.25	18.84	344.8
0.30	3	28.70	43.06	2006.8	0.00	19.71	341.2	48.23	59.43	3188.0	1.92	21.26	371.6
0.30	4	38.62	54.43	2278.3	0.00	21.60	1056.2	47.45	60.49	3191.3	1.02	22.46	332.8
0.30	5	39.52	55.74	2325.8	0.10	25.75	1482.3	45.18	59.58	3190.0	0.89	26.29	376.9
0.40	2	31.33	49.25	2079.5	0.00	22.97	389.7	50.93	62.44	3193.3	1.84	24.51	342.7
0.40	3	42.76	56.00	2428.1	0.00	19.86	1142.4	49.05	60.15	3191.4	2.66	21.96	356.0
0.40	4	44.94	59.16	2585.4	0.22	22.97	1600.8	48.48	61.26	3188.7	1.27	23.90	361.8
0.40	5	44.07	60.33	2529.0	1.75	27.30	2068.7	46.38	61.43	3191.6	0.89	26.63	361.9
Overall		20.25	37.72	1473.3	0.13	21.46	647.9	47.19	58.67	3187.5	1.37	22.44	346.2

Table 2: Comparison of the four lower bounds.

We consider the following lower bounds:

1. LB-NLModel. It is the best feasible solution found by solver Baron, solving model (2)–(12) with valid inequalities (13)–(15) with a time limit of 3600s. As Baron does not support user callbacks and dynamic cut generation, we could not use valid inequalities (16).
2. LB-NLCModel. Similar to bound LB-NLModel, but replacing the objective function with its logarithm, to ensure concavity. The value of the bound is then calculated recomputing the original objective value. We observe that, while for LB-NLModel we could also report the solver gap (i.e., the gap between the best lower and upper bounds reported by Baron at the end of the computation), we cannot do the same for LB-NLCModel. The reason is that Baron does not provide a way to access the variables' values of the (fractional or otherwise infeasible) solution

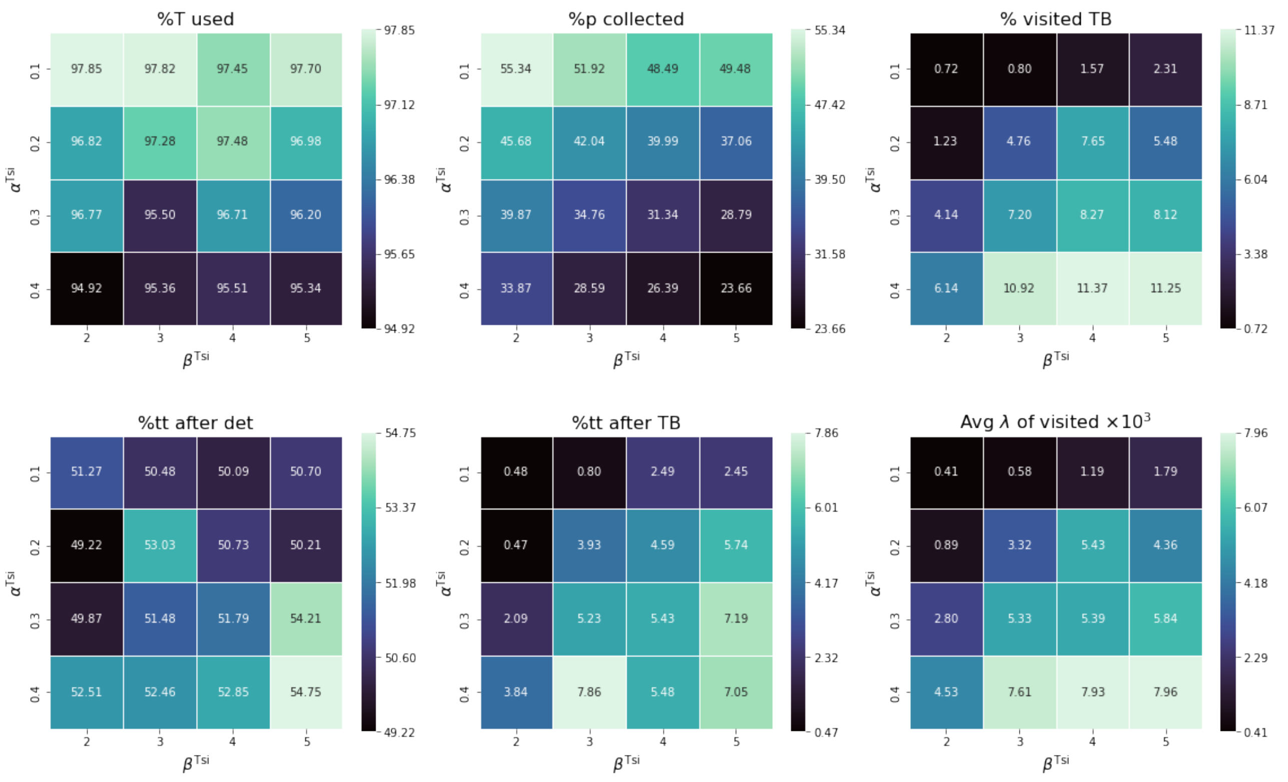


Figure 3: Change in six solution metrics, when instance generation parameters α^{Tsi} and β^{Tsi} change. Each value in the tables is an average over all instances which share the same generation parameters.

which produces the dual bound and, therefore, we cannot recompute the original objective. We did not list the solver gap for LB-NLModel in Table 2 because it would be unique to this method and, thus, unuseful for comparison; we report, however, that its average value is of 23.50% over all instances.

3. LB-DP. Solution computed using the dynamic programming algorithm presented in Section 5.1. The algorithm ran with a time limit of 3600s, at the end of which it returned the best non-dominated feasible solution.

4. LB-ANS. Solution produced using the ANS algorithm presented in Section 5.2 with $\pi^{\mathrm{iter}} = 10\,000$ iterations as the termination criterion. We used the following values for the other parameters: $\pi^{\mathrm{sa}} = 1, \pi^{\mathrm{sic}} = 1.005, \pi^{\mathrm{sib}} = 1.01, \pi^{\mathrm{ls}} = 500, \pi^{\mathrm{id}} = 2000, \pi^{\mathrm{start}} = 0.8, \pi^{\mathrm{end}} = 0.4$. We use the ANS algorithm as a proof-of-concept to show the effectiveness of the proposed neighbourhoods. However, one could envisage multiple ways to improve the efficiency and the efficacy of the ANS, for example by (i) performing parameter tuning to increase the quality of the solutions, and (ii) reducing the number of iterations (or stopping after the best solution is not improved for some iterations) to reduce the runtime. Nevertheless, this is out of scope for the current study and left for future research, as mentioned in the conclusions.

Table 2 summarises the results of our experiments. We first note that transforming the objective function of the HOP has a massive impact on the capacity of black-box solver Baron to solve instances. LB-NLCModel has lower gaps and faster runtimes, compared with LN-NLModel. In fact, LB-NLCModel finds the best-known solution in a large number of cases and thus the associated values for gap G_1 are close to 0. The dynamic programming algorithm (lower bound LB-DP) is not of practical utility due to its large runtimes and because the best solutions found within the time limit are often of inferior quality. Finally, the ANS algorithm produces high quality solutions but only uses around 40% of the time needed by LB-NLCModel, thus being an attractive alternative to solve real-size instances.

6.4 Analysis of the solutions

In this section, we explore the properties of high-quality solutions and how they vary depending on the instance generation parameters. For each instance, we use the best solution returned by the ANS algorithm after 10 000 iterations. Figure 3 shows how the value of six metrics changes when varying instance generation parameters α^{Tsi} and β^{Tsi}. The considered metrics are the following:

- **%T used.** This is the percentage of the time limit T used by the solution. High-quality solutions of the (purely deterministic) OP usually use almost 100% of the time limit [11, 24]. Intuitively, we could expect that the more time-bomb customers are in the instance and the higher is their profit, the lower this value will be. After visiting a time-bomb customer with a high enough profit, it is convenient to go back to the depot without visiting other customers to limit the probability of exploding. Thus, higher values of β^{Tsi} could be correlated with lower %T used. Furthermore, high-quality solutions will visit only a few time-bomb customers. If there are not enough deterministic customers to fill the remaining available time, it might be more convenient to return to the depot earlier. Therefore, we also expect that higher values of α^{Tsi} are correlated with lower %T used. While increasing α^{Tsi}, indeed, causes a decrease in this metric, the pattern is not clear with respect to changes in β^{Tsi}. Further inspection of the solutions revealed that high-quality tours visit time-bomb customers last and tend to use all the available time before. Therefore, an increase in the profits of time-bomb customers (higher values of β^{Tsi}) does not affect the percentage of the time limit used. Only the availability of enough deterministic customers to visit (i.e., lower α^{Tsi}) plays a role.
- **%p collected.** This is the percentage of profit collected by the vehicle over the total profit $\sum_{i\in V} p_i$ in the instance. When more customers are time-bomb (increasing α^{Tsi}) this indicator is lower because the majority of time-bomb customers are not visited and thus a lot of profit is not collected. Increasing β^{Tsi} also lowers this indicator. When the profit of time-bomb customers is higher, in fact, it constitutes a higher share of the total profits. Visiting few time-bomb customers then results in renouncing a higher percentage of the total profit.
- **% visited TB.** It is the percentage of time-bomb customers over all customers visited by a tour. For example, visiting one time-bomb and nine deterministic customers would result in a value of 10%. As we expect, a higher number (higher values of α^{Tsi}) and higher profits (higher values of β^{Tsi}) for time-bomb customers are both associated with more visits.
- **%tt after det** and **%tt after TB**. These metrics report the average value of $\tau_i := 100 \cdot \frac{w_i}{\sum_{(i,j)\in A} t_{ij}x_{ij}}$ computed over visited deterministic and time-bomb customers, respectively. Recall that w_i represents the travel time after customer $i \in V$ and that $\sum_{(i,j)\in A} t_{ij}x_{ij}$ is the tour travel time. Therefore, τ_i is the percentage of travel time after visiting i. Note that if we averaged over all visited customers (without distinguishing between deterministic and time-bomb) then we would expect this value to be close to 50%. We can use these two metrics to confirm our hypothesis that high quality solutions visit time-bomb customers towards the end of the tour: "%tt after TB" is much smaller than "%tt after det". We also note that higher time-bomb profits allow more travel time after visiting time-bomb customers: we can take a slightly higher risk if the reward is also higher.
- **Avg $\boldsymbol{\lambda}$ of visited $\boldsymbol{\times 10^3}$.** This indicator is the average value of λ_i (the parameter of the exponential random variable) over customers visited by the tour. Deterministic customers are assumed to have $\lambda_i = 0$. Because this indicator results in small values, to increase the readability of Figure 3, we multiply it by a factor of 10^3. We note that this indicator increases with both α^{Tsi} and β^{Tsi} and follows quite closely indicator "% visited TB".

7 Conclusions

In this paper we introduced the Hazardous Orienteering Problem, which models a variant of the classical OP in which the parcels picked up at some "time-bomb" customers have a probability of exploding which depends on how long they travel on the vehicle and, if any parcel explodes, the entire collected profit is lost. The problem has interesting practical applications in cash transportation problems, routing of hazardous material and law enforcement with drones. The peculiar nature of the problem, due to the probability of explosion, makes its natural formulation more intriguing than the one of the classical OP, resulting in the maximisation of a non-concave objective function. This of course has implications on solution procedures: the complexity of the objective function makes the problem harder to solve than the standard OP. Thus, in this paper, we present multiple ways for deriving both upper bounds (based on integer, piece-wise integer and continuous relaxations, as well as on dynamic programming) and lower bounds (based on dynamic programming and adaptive neighbourhood search). Our goal is to present and investigate multiple techniques, rather than focusing on one of them and fine-tune its performance. We thus perform an extensive computational analysis to compare the different approaches and determine the most promising ones. Our results show that, in terms of upper bounds, the continuous relaxation is the one leading to the best results. When solving the non-linear model with a black-box solver, transforming the objective function to make it concave has a large impact on the solver's performance. In terms of heuristic solutions, ANS largely outperforms early-terminated dynamic programming. Finally, the analysis of solution features reveals that the solution structure depends on instance parameters, specifically, the number and profitability of time-bomb customers.

We envisage multiple directions for future research. On one side, one might focus on improving the approaches for determining both upper and lower bounds, for example, by enhancing the ANS through parameter tuning. On the other side, problem generalisations are also of interests, for example the case with multiple vehicles or a multi-objective extensions in which profit collected, travel time and probability of explosion define a three-dimensional Pareto frontier.

Acknowledgements

The work of Alberto Santini has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska Curie grant agreement number 945380.

References

[1] Murat Afsar and Nacima Labadie. 'Team Orienteering Problem with Decreasing Profits'. In: *Electronic Notes in Discrete Mathematics* 41 (2013), pp. 285–293. DOI: 10.1016/j.endm.2013.05.104.

[2] Enrico Angelelli, Claudia Archetti, Carlo Filippi and Michele Vindigni. 'A dynamic and probabilistic orienteering problem'. In: *Computers & Operations Research* 136 (2021). DOI: 10.1016/j.cor.2021.105454.

[3] Enrico Angelelli, Claudia Archetti, Carlo Filippi and Michele Vindigni. 'The probabilistic orienteering problem'. In: *Computers & Operations Research* 81 (2017), pp. 269–281. DOI: 10.1016/j.cor.2016.12.025.

[4] Tolga Bektaş and Luis Gouveia. 'Requiem for the Miller–Tucker–Zemlin subtour elimination constraints?' In: *European Journal of Operational Research* 236.3 (2014), pp. 820–832.

[5] Ann Campbell, Michel Gendreau and Barrett Thomas. 'The orienteering problem with stochastic travel and service times'. In: *Annals of Operations Research* 186 (2011), pp. 61–81. DOI: doi.org/10.1007/s10479-011-0895-2.

[6] Jianqiang Cheng, Erick Delage and Abdel Lisser. 'Distributionally robust stochastic knapsack problem'. In: *SIAM Journal on Optimization* 24 (3 2014), pp. 1485–1506. DOI: 10.1137/130915315.

[7] Andrea Chiussi, Christos Orlis, Roberto Roberti and Wout Dullaert. 'ATM cash replenishment under varying population coverage requirements'. In: *Journal of the Operational Research Society* (2021). DOI: 10.1080/01605682.2020.1866443.

[8] Nicos Christofides, Aristide Mingozzi and Paolo Toth. 'State-space relaxation procedures for the computation of bounds to routing problems'. In: *Networks* 11.2 (1981), pp. 145–164. DOI: 10.1002/net.3230110207.

[9] Lanah Evers, Kristiaan Glorie, Suzanne van der Ster, Ana Isabel Barros and Herman Monsuur. ‘A two-stage approach to the orienteering problem with stochastic weights’. In: *Computers & Operations Research* 43 (2014), pp. 248–260. DOI: 10.1016/j.cor.2013.09.011.

[10] Michael Farrington. ‘Safety of lithium batteries in transportation’. In: *Journal of power sources* 96.1 (2001), pp. 260–265. DOI: 10.1016/s0378-7753(01)00565-1.

[11] Matteo Fischetti, Juan Jose Salazar Gonzalez and Paolo Toth. ‘Solving the orienteering problem through branch-and-cut’. In: *INFORMS Journal on Computing* 10.2 (1998), pp. 133–148. DOI: 10.1287/ijoc.10.2.133.

[12] Marguerite Frank and Philip Wolfe. ‘An algorithm for quadratic programming’. In: *Naval Research Logistics* 3 (1-2 1956), pp. 95–110. DOI: 10.1002/nav.3800030109.

[13] Georg Fröhlich, Karl Doerner and Margaretha Gansterer. ‘Secure and efficient routing on nodes, edges, and arcs of simple-graphs and of multi-graphs’. In: *Networks* 76.4 (2020), pp. 431–450. DOI: 10.1002/net.21993.

[14] Maaike Hoogeboom and Wout Dullaert. ‘Vehicle routing with arrival time diversification’. In: *European Journal of Operational Research* 275.1 (2019), pp. 93–107. DOI: 10.1016/j.ejor.2018.11.020.

[15] David Houck, Jean-Claude Picard, Maurice Queyranne and Ramakrishna Vemuganti. ‘The travelling salesman problem as a constrained shortest path problem: theory and computational experiments’. In: *Opsearch* 17 (1980), pp. 93–109.

[16] Taylan İlhan, Seyed Iravani and Mark Daskin. ‘The orienteering problem with stochastic profits’. In: *IIE Transactions* 40.4 (2008), pp. 406–421. DOI: 10.1080/07408170701592481.

[17] Diego Lisbona and Timothy Snee. ‘A review of hazards associated with primary lithium and lithium-ion batteries’. In: *Process Safety and Environmental Protection* 89.6 (2011), pp. 434–442. DOI: 10.1016/j.psep.2011.06.022.

[18] Michele Monaci, Cyara Pike-Burke and Alberto Santini. ‘Exact algorithms for the 0-1 time-bomb knapsack problem’. In: *Computers & Operations Research* To appear (2022).

[19] Sandra Ulrich Ngueveu, Christian Prins and Roberto Wolfler Calvo. ‘Lower and upper bounds for the m-peripatetic vehicle routing problem’. In: *4OR* 8.4 (2010), pp. 387–406. DOI: 10.1007/s10288-010-0148-2.

[20] Christos Orlis, Nicola Bianchessi, Roberto Roberti and Wout Dullaert. ‘The Team Orienteering Problem with Overlaps: An Application in Cash Logistics’. In: *Transportation Science* 54.2 (2020), pp. 470–487. DOI: 10.1287/trsc.2019.0923.

[21] Christos Orlis, Demetrio Laganà, Wout Dullaert and Daniele Vigo. ‘Distribution with Quality of Service Considerations: The Capacitated Routing Problem with Profits and Service Level Requirements’. In: *Omega* 93 (2020). DOI: 10.1016/j.omega.2019.02.003.

[22] Vassilis Papapanagiotou, Roberto Montemanni and Luca Maria Gambardella. ‘Objective function evaluation methods for the orienteering problem with stochastic travel and service times’. In: *Journal of Applied Operational Research* 6.1 (2014), pp. 16–29. ISSN: 1927-0089.

[23] Stefan Ropke and David Pisinger. ‘An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows’. In: *Transportation Science* 40.4 (2006), pp. 455–472. DOI: 10.1287/trsc.1050.0135.

[24] Alberto Santini. ‘An Adaptive Large Neighbourhood Search algorithm for the Orienteering Problem’. In: *Expert Systems with Applications* 123 (2019), pp. 154–167. DOI: 10.1016/j.eswa.2018.12.050.

[25] Alberto Santini. ‘Repository hazardous-orienteering-problem’. In: (2022). DOI: 10.5281/zenodo.6381846. URL: https://github.com/alberto-santini/hazardous-orienteering-problem.

[26] Alberto Santini, Stefan Ropke and Lars Magnus Hvattum. ‘A comparison of acceptance criteria for the Adaptive Large Neighbourhood Search metaheuristic’. In: *Journal of Heuristics* 24 (5 2018), pp. 783–815. DOI: 10.1007/s10732-018-9377-x.

[27] Hanif Sherali, Laora Brizendine, Theodore Glickman and Shivaram Subramanian. ‘Low Probability High Consequence Considerations in Routing Hazardous Material Shipments’. In: *Transportation Science* 31.3 (1997). DOI: 10.1287/trsc.31.3.237.

[28] Luca Talarico, Kenneth Sörensen and Johan Springael. ‘Metaheuristics for the risk-constrained cash-in-transit vehicle routing problem’. In: *European Journal of Operational Research* 244.2 (2015), pp. 457–470.

[29] Theodoros Tsiligirides. ‘Heuristic Methods Applied to Orienteering’. In: *Journal of the Operational Research Society* 35 (1984), pp. 797–809. DOI: 10.1057/jors.1984.162.

[30] Shu Zhang, Jeffrey W Ohlmann and Barrett W Thomas. ‘Dynamic orienteering on a network of queues’. In: *Transportation Science* 52.3 (2018), pp. 691–706. DOI: 10.1287/trsc.2017.0761.