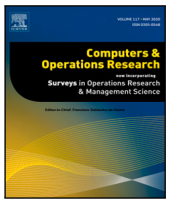
Computers & Operations Research 176 (2025) 106947

Contents lists available at ScienceDirect

Computers and Operations Research

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



A model-based algorithm for the Probabilistic Orienteering Problem

Roberto Montemanni [a,*], Derek H. Smith [b]

[a] Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Via Amendola, 2, 42122 Reggio Emilia, Italy
[b] Computing and Mathematics, University of South Wales, Pontypridd CF37 1DL, Wales, UK

ARTICLE INFO

Keywords:
Probabilistic Orienteering Problem
Heuristic algorithms
Exact algorithms

ABSTRACT

The Orienteering Problem is a routing problem aiming at selecting a subset of a given set of customers to be visited within a given time budget, so that a total revenue is maximized. Multiple variants of the problem have been studied. The Probabilistic Orienteering Problem is one of these variants, where customers will require a visit according to a certain given probability. Stochasticity makes the model more practical, but concurrently more difficult to solve. Effective approaches to solve the problem potentially lead to higher quality planning in real-life logistics, thanks to the exploitation of the probabilistic informations that can normally be derived from historical data.

In this paper we present an iterative model-based algorithm that solves a sequence of deterministic problems and is able to retrieve and certify optimal solutions if run for sufficient time. Experimental results show that the new approach is performing well when compared against both the exact (proven optimality) and heuristic (high quality solutions) algorithms available in the literature.

1. Introduction

The Orienteering Problem (OP) was introduced in Tsiligirides (1984) and Golden et al. (1987) in the 1980s. In an interpretation of the problem from the viewpoint of logistics, there is a single vehicle, leaving from and returning to a depot, that serves a set of customers, each one associated with a spacial location and a profit, with such a profit collected upon visiting the location. Travel times among locations are known, deterministic, and given. However, not all the customers can typically be serviced, since the vehicle mission cannot be longer than a given maximum time. The aim of the problem is to maximize the total profit collected by the vehicle in the available time. The problem has attracted a lot of attention due to its practical implications, and many variations of the original problem have been introduced over the years. We refer the interested reader to Vansteenwegen et al. (2011), Gunawan et al. (2016), Fischetti et al. (2007) and Archetti et al. (2014) for exhaustive reviews of the scientific literature on these problems. The OP has important applications in last-mile logistics, where it is typically used to plan operations in advance. In recent years, the optimization of last-mile logistics processes has been getting more and more central, due to the remarkable actual and forecasted increase in e-commerce (Statista, 2024), which in turns generates a strong need for effective operational solutions (Angelelli et al., 2014; Archetti et al., 2018, 2024).

Stochastic optimization problems have raised considerable attention among researchers in the last decades. In contrast to traditional optimization paradigms, these problems integrate uncertainty parameters, resulting in better representations of real-world situations, as long as the probabilistic estimates are correct. The drawback is that addressing stochastic problems typically implies more complex models that require a substantially heavier computational effort to solve, posing therefore algorithmic challenges. The Probabilistic Orienteering Problem (POP) was initially introduced by Angelelli et al. (2017), motivated by a real application. When ahead-planning the route for a truck to collects parcels from different customers from a certain area, the logistics operator has typically to include some mandatory customers and to select a subset of possible transportation requests from spot customers under negotiation. In this context, the maximum length of each route is a hard constraint, being regulated by laws. The firm wants to be able to satisfy all the selected requests in case the contracts are finalized, although it is not guaranteed that all of them will be eventually arranged. Notice that the revenue associated with a certain shipping is quite certain, since it follows standard pricing tables. However, the possibility that such a shipping will be arranged can be highly uncertain. So the POP was originally designed as a tool for tactical/operational decisions, to select which spot customers should be selected for the negotiation phase, and which not. This tool requires an almost online evaluation of the expected revenues and the expected traveling times/costs associated with any customer.

* Corresponding author.
E-mail addresses: roberto.montemanni@unimore.it (R. Montemanni), derek.smith@southwales.ac.uk (D.H. Smith).

https://doi.org/10.1016/j.cor.2024.106947
Received 13 June 2024; Received in revised form 15 November 2024; Accepted 6 December 2024
Available online 13 December 2024

The POP represents a variant of the Orienteering Problem where each customer requires a visit according to a given probability. Consequently, the POP seeks to devise optimal plans by selecting a subset of customer to be visited within the given time, that provides the highest expected reward associated to the visits, while minimizing the expected travel time. The calculation of the expected travel time is computationally demanding and difficult to insert in Mathematical Programming models, and represents therefore the bottleneck of the problem. Thus attention has focused on heuristic approaches. A study on the use of Monte Carlo sampling for the POP was proposed in Chou et al. (2018), while a Tabu Search algorithm incorporating such technology, was presented in Chou et al. (2021).

The application scope of POP solutions extends to various domains, including tourism-related scenarios such as Tourist Trip Design Problems (TTDP) (Vansteenwegen and Oudheusden, 2007). In these contexts, high-quality solutions within short calculation times are important for real-time decision-making. Problems similar in nature to the POP that have recently appeared in the literature, can be found, for example, in Angelelli et al. (2021), where an online OP is discussed and in Santini and Archetti (2023), where a version of the OP tailored for the transportation of hazardous material is proposed.

The contributions of this work are as follows. Firstly, a new approximation for the expected travel time of a POP solution is introduced. It is fast to calculate and can be easily inserted in mathematical programming models. Two models are then introduced, that are able to provide heuristic solutions for the POP and upper bounds for its optimal cost. One of these model adopts the new approximation. The new models are then inserted into an iterative algorithmic framework – which can be generalized to similar problems – able to provide high-quality heuristic solutions and proven optimal solutions if enough computation time is allowed. Computational experiments on the benchmark instances commonly adopted in the literature are then presented, positioning the new approaches with respect to the exact and heuristic methods previously presented in the literature.

2. Problem description

The Probabilistic Orienteering Problem (POP) can be defined on a digraph $G = (V, A)$ where $V = \{0, 1, \dots, N\}$ represents the depot 0 and customers, while A denotes connections between pairs of elements of V. A deterministic travel time t_{ij} is provided for each $(i, j) \in A$. For each node $i \in V$ a prize p_i, which is collected upon visiting the customer is provided, together with a probability π_i of the customer to require a visit. By definition $p_0 = 0$ and $\pi_0 = 1$. Finally, a maximum time T_{max} is provided. The objective of the POP is to retrieve a feasible tour of length at most T_{max} that maximizes the expected prize collected minus the expected travel time multiplied by a normalization factor C (normally decided by the final user).

Given a solution $\sigma = \{\sigma_0, \sigma_1, \dots, \sigma_q, \sigma_{q+1}\}$, with $\sigma_0 = \sigma_{q+1} = 0$ (the depot), and represented as a sequence of vertices selected to be visited, its objective function value is obtained by an exact evaluation, as described in Angelelli et al. (2017), and with the product term defined to take value 1 when $h + 1 > k - 1$:

$$u(\sigma) = \sum_{k=1}^{q} \pi_{\sigma_k} p_{\sigma_k} - C \sum_{h=0}^{q} \left[\pi_{\sigma_h} \sum_{k=h+1}^{q+1} t_{\sigma_h \sigma_k} \pi_{\sigma_k} \prod_{j=h+1}^{k-1} \left(1 - \pi_{\sigma_j}\right) \right] \quad (1)$$

where the probability of appearance of each arc for the estimation of the expected travel time, is calculated based on the probabilities of being present associated to each vertex of the selected route. Behind this expression is the idea that a solution σ is an a-priori tour, such that, in case of a no-show of a customer, the vehicle moves directly to the next customer, making a shortcut and saving time. The quantity in square brackets is therefore based on all the possible combinations of customers being present or not, and the respective probabilities.

A small example of a POP instance can be found in Fig. 1.

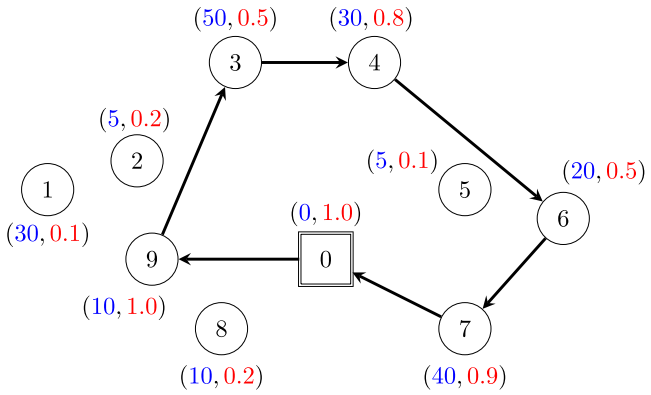


Fig. 1. Example of a POP instance, where prizes are represented in blue and probabilities in red. Travel times and the maximum duration T_{max} are omitted from the figure. However, the length of the tour cannot exceed T_{max}. Nodes to visit are selected based on their expected contribution to the collected prize and to the traveled distance.

3. A model providing heuristic solutions and upper bounds

In this section we discuss a model that by construction underestimates the expected total travel times, providing therefore an upper bound for the cost of the optimal solution of the POP. The advantage of such a model is that it is fully deterministic, and therefore simpler to solve than a probabilistic one. For this reason, the new model can potentially be a strategic ingredient for an iterative algorithm, that will be explained in Section 4.

Every solution which is feasible for the new model, is feasible also for the POP, providing therefore a heuristic solution. Moreover, the POP-cost of such a solution can be evaluated via Eq. (1). Since the new model underestimates (by construction) the expected total travel time incurred by a solution, the optimal cost provided by the new model is a valid upper bound for the optimal solution of the POP.

The models we propose are tailored for the Constraint Programming solver CP-SAT, which is part of the Google OR-Tools suite (Perron and Didier, 2024). The choice is motivated by the successful recent applications of such a solver on similar problems characterized by a compact model (Montemanni and Dell'Amico, 2023; Montemanni et al., 2024), and by its intrinsic simplicity of use (in our case models are embedded into a higher level algorithm, see Section 4). It should however be observed that it is straightforward to express the models as traditional Mixed Integer Linear Programming (MILP) problems and solve them with dedicated solvers, eventually further enhancing the overall performance of the methods we propose.

In detail, the new model solves a generalized deterministic Orienteering Problem, and is based on the some y binary variables such that y_{ij} takes value 1 if the planned route visits vertex j right after vertex i, and 0 otherwise. By construction y_{ii} takes value 1 if node i is not visited and 0 otherwise.

$$(B) \quad \max \sum_{i \in V} \pi_i p_i \neg y_{ii} - C \sum_{i \in V} \sum_{j \in V, j \neq i} \pi_i \pi_j t_{ij} y_{ij} \quad (2)$$

$$s.t. \quad \text{AddCircuit}(y_{ij}; i, j \in V) \quad (3)$$

$$\sum_{i \in V} \sum_{j \in V, j \neq i} t_{ij} y_{ij} \leq T_{max} \quad (4)$$

$$y_{ij} \in \{0; 1\} \qquad i, j \in V \quad (5)$$

The objective function (2) of model B (as *Basic*) maximizes the expected profit collected in the tour minus a lower bound for the expected travel time, calculated by considering only the contributions (in probability) of the arcs of the planned tour, and neglecting the travel times incurred when one or more customers of the route are not present, according to the associated probability. Notice the use of

the negation operator "¬" (such that $\neg y = 1 - y$), which is common in Constraint Programming and is used in the rest of the paper to assert that a node is visited. Constraint (3) imposes that the tour associated with the active y variables forms a feasible circuit. This is imposed by the CP-SAT statement *AddCircuit* that also ensures that $y_{ii} = 1$ for each variable $i \in V$ not touched by the circuit itself. Constraint (4) is a budget constraint requiring that the length of the tour described by the active y variables has a length of at most T_{max}. Constraints (5) finally define the domain of the variables of the model.

Observe that, given a POP solution, it is computationally and logically easy to pass from an arc-based representation (y as in the model) to a sequence-based representation (σ as in (1)), and vice versa.

3.1. An improved model

In the objective function (2) of model B, the only tour strictly identified by the y variables is accounted for in the calculation of the expected travel time, while the cost incurred once one or more customers are not present, is simply neglected. A way of tightening the estimation for the expected total travel time can be derived as follows. Once some of the customers directly following a customer i along the tour (identified by the y variables) are not present, the first location j present on the tour after i is assumed to be visited instead, and the travel time from i to j is incurred. Notice that such a travel time is a lower bound for the real travel time incurred by the customer, but it is easier to calculate.

The travel time δ_i between customer i and its closest customer is precalculated for each customer during preprocessing and this can be done in a negligible polynomial time. Notice that $\delta_0 = 0$ by definition, since the depot 0 is the only node appearing twice in any feasible solution. The following new function can be derived to approximate the cost of a solution defined by (1):

$$e(y) = \sum_{i \in V} (\pi_i p_i - C\delta_i \pi_i) \neg y_{ii} - C \sum_{i \in V} \sum_{j \in V, j \neq i} \left(t_{ij} - \delta_i\right) \pi_i \pi_j y_{ij} \tag{6}$$

In the first part of the formula we calculate the estimated price collected by a given tour, and we pre-charge (weighted by constant C) a contribution to the estimated travel times given by $\pi_i \delta_i$ for each customer i on the tour. In the second part of the formula we rectify the pre-charged values by increasing a part of the probabilistic contribution of each node i on the tour from $\delta_i \pi_i \pi_j$ to $t_{ij} \pi_i \pi_j$, where j is the customer visited after i in the tour. Notice that $t_{ij} \geq \delta_i$ by definition. As a result, for each customer i we charge a contribution $t_{ij} \pi_i \pi_j + (\pi_i - \pi_i \pi_j)\delta_i$, with the second component enhancing the estimation provided by model B.

More formally, the result of Proposition 1 can be derived. However, we first need a lemma that will be used in the proof of Proposition 1.

Lemma 1. *Given a feasible POP solution, expressed as a sequence of customers $\sigma = \{\sigma_0, \sigma_1, \ldots, \sigma_q, \sigma_{q+1}\}$, and an index $1 \leq h < q$, the following result holds:*

$$\sum_{k=h+2}^{q+1} \pi_{\sigma_k} \prod_{j=h+1}^{k-1} \left(1 - \pi_{\sigma_j}\right) = 1 - \pi_{\sigma_{h+1}} \tag{7}$$

Proof. Given a POP solution expressed as a sequence of customers σ, the left-hand-side of Eq. (7) accounts for the probabilities of all the possible outgoing arcs from a given node σ_h which is part of the solution, with the exclusion of the arc (σ_h, σ_{h+1}). Since σ_h is not the depot, once it is visited, there has to be an outgoing arc from σ_h with probability 1. However, we need to subtract $\pi_{\sigma_{h+1}}$, the probability of σ_{h+1} to be present (ruling out the arc (σ_h, σ_{h+1})). The resulting quantity corresponds to the right-hand-side of Eq. (7). □

We are now ready to enunciate the main result:

Proposition 1. *Given a feasible POP solution, that can be expressed alternatively as a sequence of customers as $\sigma = \{\sigma_0, \sigma_1, \ldots, \sigma_q, \sigma_{q+1}\}$, or as a set of active arcs as y, we have that $u(\sigma) \leq e(y)$.*

Proof.

$$u(\sigma) = \sum_{k=1}^{q} \pi_{\sigma_k} p_{\sigma_k} - C \sum_{h=0}^{q} \left[\pi_{\sigma_h} \sum_{k=h+1}^{q+1} t_{\sigma_h \sigma_k} \pi_{\sigma_k} \prod_{j=h+1}^{k-1} \left(1 - \pi_{\sigma_j}\right) \right]$$

extracting arc (σ_h, σ_{h+1}) from the second sum

$$u(\sigma) = \sum_{k=1}^{q} \pi_{\sigma_k} p_{\sigma_k} - C \sum_{h=0}^{q} \left[t_{\sigma_h(\sigma_{h+1})} \pi_{\sigma_h} \pi_{\sigma_{h+1}} + \pi_{\sigma_h} \sum_{k=h+2}^{q+1} t_{\sigma_h \sigma_k} \pi_{\sigma_k} \prod_{j=h+1}^{k-1} \left(1 - \pi_{\sigma_j}\right) \right]$$

with the third summation defined to take value 0 when $h > q - 1$.

As $t_{\sigma_h \sigma_k} \geq \delta_{\sigma_h}$, we have

$$u(\sigma) \leq \sum_{k=1}^{q} \pi_{\sigma_k} p_{\sigma_k} - C \sum_{h=0}^{q} \left[t_{\sigma_h(\sigma_{h+1})} \pi_{\sigma_h} \pi_{\sigma_{h+1}} + \delta_{\sigma_h} \pi_{\sigma_h} \sum_{k=h+2}^{q+1} \pi_{\sigma_k} \prod_{j=h+1}^{k-1} \left(1 - \pi_{\sigma_j}\right) \right]$$

according to Lemma 1, we have that

$\pi_{\sigma_h} \sum_{k=h+2}^{q+1} \pi_{\sigma_k} \prod_{j=h+1}^{k-1} \left(1 - \pi_{\sigma_j}\right) = \pi_{\sigma_h} - \pi_{\sigma_h} \pi_{\sigma_{h+1}}$, so

$$u(\sigma) \leq \sum_{k=1}^{q} \pi_{\sigma_k} p_{\sigma_k} - C \sum_{h=0}^{q} \left[t_{\sigma_h(\sigma_{h+1})} \pi_{\sigma_h} \pi_{\sigma_{h+1}} + \delta_{\sigma_h} (\pi_{\sigma_h} - \pi_{\sigma_h} \pi_{\sigma_{h+1}}) \right]$$

changing the viewpoint $\sigma \Rightarrow y$

$$u(\sigma) \leq \sum_{i \in V} \pi_i p_i \neg y_{ii} - C \left[\sum_{i \in V} \sum_{j \in V, j \neq i} (t_{ij} - \delta_i) \pi_i \pi_j y_{ij} + \sum_{i \in V} \delta_i \pi_i \neg y_{ii} \right]$$

rearranging

$$u(\sigma) \leq \sum_{i \in V} \left(\pi_i p_i - C\delta_i \pi_i\right) \neg y_{ii} - C \sum_{i \in V} \sum_{j \in V, j \neq i} (t_{ij} - \delta_i) \pi_i \pi_j y_{ij} = e(y) \quad \square$$

We can therefore define a new heuristic model E (as *Enhanced*) which is defined by constraints (3), (4), (5) and the following objective function:

$$(E) \ \max \sum_{i \in V} (\pi_i p_i - C\delta_i \pi_i) \neg y_{ii} - C \sum_{i \in V} \sum_{j \in V, j \neq i} \left(t_{ij} - \delta_i\right) \pi_i \pi_j y_{ij} \tag{8}$$

Notice that the structure of models B and E is the same since only the coefficients associated with the y variables in the objective function are changed.

4. An exact algorithm

As pointed out in Section 3, the solution of model B (or E) is in fact a heuristic solution for the original POP, producing therefore a lower bound upon evaluation of its cost through formula (1). On the other hand, the cost of the optimal solution of B (or E) represents a valid upper bound for the cost of the optimal solution of the POP, since it underestimates (by construction) the travel time estimation in the last component of the objective function. It is then possible to incorporate model B (or E) into an iterative algorithm, able to retrieve a proven optimal solution. The rest of the section is devoted to the description of such a method.

Let M be the model under consideration (either B or E) and $\overline{y}$ be the set of arcs (y variables) corresponding to an optimal solution of model M and let $\overline{\sigma}$ be the sequence of customers visited in the same solution. As pointed out before, $\overline{\sigma}$ can be trivially obtained by the values of $\overline{y}$, and vice versa.

As $\overline{y}$ is a heuristic solution for the POP, evaluating σ via Eq. (1) provides a valid lower bound for the cost of the optimal solution of the POP, while the cost of the optimal solution of M, calculated according to the objective function of the model itself (either (2) or (8), in our case) remains a valid upper bound for the optimal solution of the POP. With these elements, it is easy to see that an exact algorithm can

be devised by solving model M repeatedly, every time adding a new constraint to the model itself in order to avoid visiting the same solution twice:

$$\sum_{(i,j)\in A:\overline{y}_{ij}=1} \neg y_{ij} \geq 1 \tag{9}$$

Constraint (9) imposes that at least one of the arcs visited in the optimal solution has to be omitted in the future. Given the circuit-conformation of each feasible solution, the constraint avoids repetitions without cutting out any feasible solution, given the optimality of $\overline{y}$ at a certain iteration.

The value of the upper bound provided by model M is monotonically non-increasing as the number of iterations increases, while the value of the lower bound computed by evaluating the solutions of model M through (1) oscillates. It is however possible to store the highest among the bounds encountered, which corresponds to the best heuristic solution visited so far. The algorithm ideally stops once the lower bound matches the upper bound, and the optimality of the best heuristic solution encountered is proven. In a practical implementation the computation might be stopped before its natural end, providing a heuristic solution and a valid upper bound in this case.

The exact algorithm can be summarized by the pseudocode presented in Algorithm 1.

Algorithm 1: IterAlg(M)

```
   Data: POP instance
   Result: A solution, a lower and an upper bound for the optimal
           solution cost
1  Build model M;
2  LB=0;
3  UB=+∞;
4  while LB < UB and ComputationTime < MaxTime do
5  |  Solve model M;
6  |  ȳ, σ̄ = optimal solution of model M;
7  |  UB = cost of solution ȳ according to model M;
8  |  RC = cost of solution σ̄ evaluated through (1);
9  |  if RC > LB then
10 |  |  LB=RC;
11 |  |  ŷ = ȳ;
12 |  end
13 |  Add inequality (9) to model M;
14 end
15 Return ŷ, LB, UB.
```

The algorithm takes as input a POP instance and returns a solution, its cost (lower bound) and an upper bound for the cost of the optimal solution. The algorithm starts by building model M of the input instances, and by initializing the lower and the upper bounds (line 1–3). A loop is then entered, that will be exited once either the optimality of a solution is proven, or the maximum allowed computation time has elapsed (in this case the returned solution will be heuristic). At each iteration, the current model M is solved and the optimal solution, represented both as a sequence of customers and as a set of arcs, is stored (lines 5 and 6); the upper bound UB is updated based on the optimal solution of model M (line 7) and the variable RC takes the real value of the POP solution returned by M (line 8). If RC is higher than the current lower bound LB, the bound is updated and the solution is stored as the best one retrieved so far (lines 9–12). A new inequality (9) is finally added to model M (line 13). Once the computation is completed, the best solution retrieved is returned together with the calculated lower and upper bound (line 15).

5. Experimental results

In this section we aim at understanding the potential of the Algorithm described in Section 4. The instances commonly adopted in the literature are considered and, after a preliminary comparison between the impact of the use of models B or E, a comparison against the algorithms available in the literature is proposed. We split the comparison between exact and heuristic algorithms, in order to highlight the good performance of the new methods from both viewpoints.

The approach described in Section 4 was implemented in Python, and Google OR-Tools CP-SAT 9.9 (Perron and Didier, 2024) was used as a solver for the models faced during the computation (see Section 3).

5.1. Benchmark instances

The POP benchmarks used for the main experiments in this work are the 264 instances generated by Angelelli et al. (2017) and available from OR-Brescia - Benchmark Instances (2024). They are based on 22 TSP benchmark instances originally from the TSPLIB library (Reinelt, 1991).

The characteristics of the instances are set as follows:

- The customers information and distances are taken directly from the corresponding original TSP instances. The first customer of the original instance takes the role of the depot 0.
- The global Deadline T_{max} takes the value $\omega \cdot T_{TSP}$, with T_{TSP} being the known optimal value of the original TSP instance and $\omega = \{\frac{1}{4}, \frac{2}{4}, \frac{3}{4}\}$ representing short, medium and long deadlines. This denominator of ω is reflected in the names of the instances as "q1", "q2" or "q3".
- The prizes collected while visiting the customers are either $p_i = 1$ for all customers ("g1" in the names of the instances), or p_i is a pseudo-randomly generated integer in {1, 2, ..., 100} for each customer i ("g2" in the names of the instances).
- The probabilities of the customers requiring visits are either $\pi_i = 0.5$ for all customers ("p1" in the names of the instances), or π_i is a random number in the interval [0.25, 0.75] for each customer i ("p2" in the names of the instances).

The coefficient C used in (1) to balance between the travel time and the prize components, is taken as $C = 0.001$, as used in the previous literature (Angelelli et al., 2017). This allows for a fair comparison among the results obtained by the different methods.

5.2. Comparison of the performance of models B and E

In this section we aim to evaluate the advantages provided by the use of model E over model B, i.e. using the objective function (8) instead of (2) in the model defined in Section 3, both in isolation and when embedded within the *IterAlg* described in Section 4.

5.2.1. Models in isolation

In Fig. 2 the upper bounds provided by model B and E on all the 12 instances derived from the TSPLIB benchmark *berlin52* are depicted. The results are representative of what happens in general on the POP benchmarks. On the x axis the different instances are considered, while on the y axis the gaps from the optimal costs are reported. They are calculated as $\frac{UB(M)-Opt}{UB(M)}$, where Opt is the optimal cost of the instance under investigation and $UB(M)$ is the upper bound provided by method M, with M that is either B or E. The chart suggest first of all that for all the settings the models provide fairly good estimations, with a maximum gap of 10%–15% and approaching 0 for some of the settings. In detail, the more critical settings are those for which the prizes are small and therefore the expected travel times have a more important role ($p = 1$). The advantage of model E over B is evident, and allows a reduction of the gap up to circa 4%.

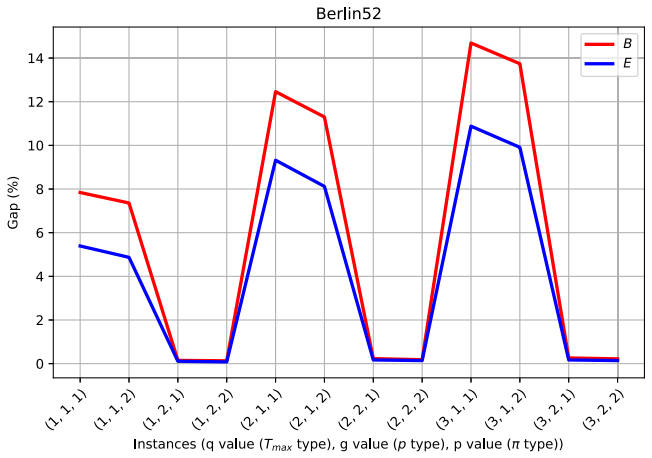


Fig. 2. Percentage gap from the optimal cost for models B and E on the instances generated from the TSPLIB benchmark *berlin52*.

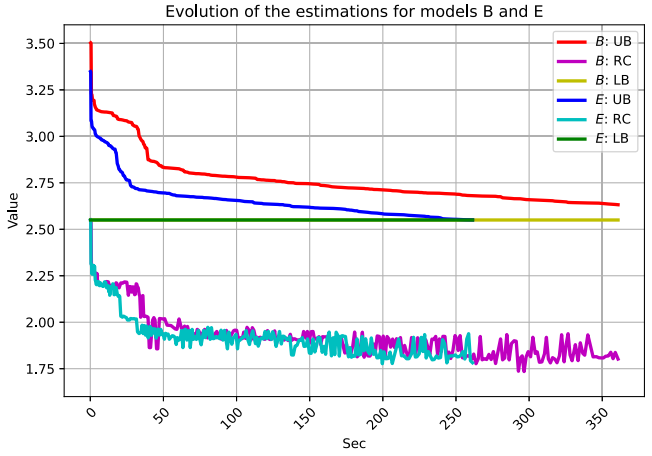


Fig. 3. Evolution of the bounds provided by models B or E within algorithm IterAlg on the POP instance *burma14_q2_g1_p2*.

5.2.2. Models within algorithm IterAlg

In Fig. 3 we chart the bounds provided by models B and E while running within the algorithm IterAlg on the instance *burma14_q2_g1_p2*. The values of the upper and lower bounds over time, together with the real cost of the solutions generated (*RC*) are plotted. The experiments are run on a computer equipped with an Apple M1 processor running at 2.0 GHz and 8 GB of RAM, and a maximum computation time of 360 s was allowed.

From Fig. 3 it is possible to appreciate the advantage provided by the use of model E over B. The better upper bound is maintained over the whole computation and allows the closure of the instance considered in approximately 260 s, while the algorithm embedding model B is not able to converge in the given time. It is interesting also to observe how the optimal solution is retrieved within the first iterations by both the instances of the *IterAlg* method, showing the potential of the method to retrieve high-quality solutions quickly. This indicates that the approximations of the POP objective function provided by models B and E (within IterAlg) tend to maintain a good ranking of the POP solutions, especially in the first iterations. Later the lines present oscillations although the descending trend remains clear.

5.3. Comparison against other approaches from the literature

In this section we compare the general performance of the *IterAlg* algorithm we propose with the other exact and heuristic methods for the POP previously appeared in the literature. We will first evaluate the performance of the new algorithm in terms of the exact solutions (or bounds) and then purely in terms of the heuristic solutions generator (without considering upper bounds on the optimal cost, but only the quality of the lower bounds).

5.3.1. Comparison against the available exact approaches

Three branch-and-cut methods, differing from each other in the branching strategy, are proposed by Angelelli et al. (2017). We will refer to them as *Chain*, *Path* and *2W-path*, from the name of the different branching strategies defining them. We refer the interested reader to Angelelli et al. (2017) for complete details of the methods. For each method considered we present the number of best-known solutions retrieved (*#bs*), the average percentage optimality gap at the end of the computation – calculated as $100 \cdot (UB(P) - LB(P))/UB(P)$, where $LB(P)$ and $UB(P)$ are the lower and upper bounds retrieved by the procedure P under investigation – and the average computation time to optimality (*Sec*). The experiments for the methods discussed in Angelelli et al. (2017) were carried out on a computer equipped with an Intel Xeon E5-1650 processor running at 3.5 GHz and 16 GB of RAM, with a maximum computation time of 7200 s, while those for the new method *IterAlg* were run on a computer running an Intel Core i7 12700F processor running at 2.1 GHz at and equipped with 32 GB of RAM with a maximum computation time of 3600 s. The different settings make the comparison more difficult, but the results remain clear. Figures are presented by aggregating the instances with similar features in terms of size, T_{max} range, prices and probabilities, for the instances. The detailed results on each instance for the method *IterAlg(E)* are provided in Appendix.

The results of Table 1 suggest that the algorithm we propose is effective in proving optimal solutions in the given time limit. In particular, both *IterAlg(B)* and *IterAlg(E)* are able to retrieve the best-known solution for substantially more instances than the other exact methods appeared in the literature, with a shorter average running time (which is a consequence of the maximum running time of the new methods being half of that of the other methods). However, the average optimality gap is worse than some of the other methods (the lower maximum allowed running time is again likely to have a role here). This suggests that the new methods are either very efficient in proving optimality, or they tend to struggle to converge. When analyzing the different groups of instances, we can observe how the new methods perform worse on the larger instances, showing therefore some scalability issues, and on the instances with flat prices ($p_i = 1$). This latter result can be explained by observing that on these instances the estimation of the travel times has a stronger impact on the final objective function. This makes the approximations we use more central, and therefore critical. It is however interesting to observe that these criticalities are common to all the methods in the table, although they appear to be mitigated more for the old methods. Finally, when comparing directly *IterAlg(B)* and *IterAlg(E)*, the advantage of the latter one is evident under any point of view, as expected.

5.3.2. Comparison against the available heuristic approaches

Three matheuristics based on the branch-and-cut framework already encountered in Section 5.3.1 are proposed by Angelelli et al. (2017): Smart-Chain, Smart-Path and Smart-Two-Ways-Path, which are here referred to as *S-Chain*, *S-Path* and *S-2W-Path*, respectively. In these approaches heuristic branching and variable fixing rules are adopted to reduce the solution space. We refer the interested reader to Angelelli et al. (2017) for complete details of the methods. A Tabu Search algorithm (*TS*) is instead presented in Chou et al. (2021). For each method considered we present the number of best solutions retrieved (*#bs*), the average percentage gap from the best-known solution – calculated as $100 \cdot (BK - LB(P))/Best$, where BK is the cost of the best known solution and $LB(P)$ is the cost of the best solution retrieved by the procedure P under investigation – and the average computation time to retrieve the

Table 1
Comparison among different exacts for the POP.

Instances \ Methods		Basic Angelelli et al. (2017)			Chain Angelelli et al. (2017)			Path Angelelli et al. (2017)			2W-Path Angelelli et al. (2017)			IterAlg(B)			IterAlg(E)		
		# *bs*	Opt Gap %	Sec	# *bs*	Opt Gap %	Sec	# *bs*	Opt Gap %	Sec	# *bs*	Opt Gap %	Sec	# *bs*	Opt Gap %	Sec	# *bs*	Opt Gap %	Sec
$n < 30$	(84)	63	4.06	2024	67	2.55	1606	67	2.47	1516	67	2.37	1500	79	3.66	1249.54	81	2.74	1097.59
$30 \le n < 50$	(84)	58	2.79	2540	66	1.79	1443	68	1.52	1274	69	1.48	1142	80	3.73	1281.51	80	2.55	1078.20
$n \ge 50$	(96)	25	23.89	5678	62	7.27	2978	61	7.39	3213	62	7.35	3144	83	13.99	2347.92	84	10.90	2026.99
$1/4T_{max}$	(88)	62	11.62	2479	78	2.77	899	81	2.44	806	83	2.27	572	87	6.07	1064.15	87	3.29	923.94
$2/4T_{max}$	(88)	46	11.32	3775	66	4.01	1958	66	3.96	2010	66	3.98	1981	84	7.37	1566.94	85	6.34	1358.86
$3/4T_{max}$	(88)	38	9.45	4504	51	4.82	3366	49	4.89	3366	49	4.85	3405	71	8.88	2346.28	74	7.31	2005.36
$p_i = 1$	(132)	63	13.16	4162	77	7.65	3309	78	7.40	3136	80	7.27	2989	115	14.76	2647.90	118	11.21	2221.80
$p_i \in \{1, \dots, 100\}$	(132)	83	8.43	3010	118	0.08	840	118	0.13	985	118	0.13	983	127	0.12	670.34	128	0.08	636.97
$\pi_i = 0.5$	(132)	71	9.91	3674	95	3.94	2115	97	3.80	2123	98	3.73	2025	118	7.44	1741.53	121	5.59	1479.41
$\pi_i \in [0.25, 0.75]$	(132)	75	11.68	3498	100	3.80	2034	99	3.72	1999	100	3.67	1947	124	7.44	1576.71	125	5.70	1379.36
ALL	**(264)**	**146**	**10.79**	**3586**	**195**	**3.87**	**2074**	**196**	**3.76**	**2061**	**198**	**3.70**	**1986**	**242**	**7.44**	**1659.12**	**245**	**5.65**	**1429.38**

Table 2
Comparison among different heuristics for the POP.

Instances \ Methods		S-Chain Angelelli et al. (2017)			S-Path Angelelli et al. (2017)			S-2W-Path Angelelli et al. (2017)			TS Chou et al. (2021)			IterAlg(B)			IterAlg(E)		
		# *bs*	Gap %	Sec	# *bs*	Gap %	Sec	# *bs*	Gap %	Sec	# *bs*	Gap %	Sec	# *bs*	Gap %	Sec	# *bs*	Gap %	Sec
$n < 30$	(84)	63	2.52	21	56	3.93	120	60	2.72	168	77	0.02	0.09	79	0.01	0.03	81	0.00	0.03
$30 \le n < 50$	(84)	49	0.38	27	37	0.75	6	34	1.05	5	39	1.03	6.29	80	0.04	0.14	80	0.04	0.14
$n \ge 50$	(96)	41	0.77	223	26	1.17	181	23	1.34	127	20	4.26	11.66	83	0.16	0.51	84	0.16	0.50
$1/4T_{max}$	(88)	56	1.02	40	50	1.25	11	48	1.57	14	66	0.63	2.70	87	0.31	0.04	87	0.31	0.04
$2/4T_{max}$	(88)	54	0.23	22	37	0.63	94	39	0.58	17	42	2.50	7.27	84	0.02	0.20	85	0.02	0.20
$3/4T_{max}$	(88)	43	0.35	232	32	0.61	209	30	0.65	288	28	2.52	8.84	71	0.16	0.21	74	0.15	0.21
$p_i = 1$	(132)	65	0.78	136	58	0.88	174	54	1.11	153	77	1.78	6.37	115	0.15	0.24	118	0.14	0.24
$p_i \in \{1, \dots, 100\}$	(132)	88	0.29	60	61	0.78	36	63	0.75	60	59	1.98	6.16	127	0.00	0.24	128	0.00	0.24
$\pi_i = 0.5$	(132)	82	0.37	126	58	0.72	127	61	0.87	123	69	1.75	6.41	118	0.09	0.24	121	0.08	0.23
$\pi_i \in [0.25, 0.75]$	(132)	71	0.70	71	61	0.94	82	56	1.00	89	67	2.01	6.13	124	0.06	0.24	125	0.06	0.24
ALL	**(264)**	**153**	**0.53**	**98**	**119**	**0.83**	**104**	**117**	**0.93**	**107**	**136**	**1.88**	**6.27**	**242**	**0.07**	**0.24**	**245**	**0.07**	**0.24**

best solution found (*Sec*). The experiments for the methods discussed in Angelelli et al. (2017) were carried out on a computer equipped with an Intel Xeon E5-1650 processor running at 3.5 GHz and 16 GB of RAM, with a maximum computation time of 7200 s, those for the metaheuristic described in Chou et al. (2021) on a computer equipped with an Intel Core i7 9700K processor running at 3.6 GHz and 16 GB of RAM, while those for the new method *IterAlg* were run on a computer running an Intel Core i7 12700F processor running at 2.1 GHz at and equipped with 32 GB of RAM with a maximum computation time of 3600 s.

The results of Table 2 indicate that the methods we propose are dominant under any factor once compared with the previous methods available in the literature. They are able to retrieve more optimal solution, with a computation time orders of magnitude shorter than the competitors – very high quality solutions are provided almost instantaneously – and with better gaps. This confirms what has already been observed in Section 5.2.2: models B and E tend to preserve the ranking of the quality of the POP solutions, and the best solutions are consistently encountered in the very first iterations. The enhanced method *IterAlg(E)* is able to match 3 optimal solutions more than *IterAlg(B)*, without incurring any increase in the computation times.

6. Conclusion

In this work an exact model-based approach for the Probabilistic Orienteering Problem, repeatedly solving a fully deterministic problem in order to devise upper and lower bounds, was proposed. An enhancement to the deterministic inner problem, aiming at improving the quality of the bounds has also been proposed. The new approach has been tested and compared with the methods currently available in the literature, both exact and heuristic. The results show good results, and in particular the quality of the heuristic solutions provided for the instances normally adopted in the literature, appears outstanding.

Future research work will extend the results to the multi-vehicle version of the POP, which could be called Probabilistic Team Orienteering Problem, and that – up to our knowledge – has not been investigated yet. The approach we proposed could be adapted by extending the models. Moreover, the method to approximate probabilistic estimations could be adapted to other probabilistic problems in similar domains.

CRediT authorship contribution statement

Roberto Montemanni: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Conceptualization. **Derek H. Smith:** Writing – review & editing, Validation, Methodology, Conceptualization.

Acknowledgments

The authors would like to thank Mauro Dell'Amico for the useful discussions and suggestions.

Appendix

The results obtained by the *IterAlg(E)* method are presented in Table 3. The experiments were run on a computer running an Intel Core i7 12700F processor running at 2.1 GHz at and equipped with 32 GB of RAM with a maximum computation time of 3600 s. For each instance of the benchmark set (defined by the name of the original TSPLIB instance and the values of the parameters T_{max}, p and π) the values of lower and upper bounds retrieved are listed together with the computation time required to retrieve the best solution, the total computation time eventually required to close the instance, and the number of iterations carried out by the algorithm.

Table 3
Detailed results obtained by the *IterAlg(E)* method.

Name	q (T_{max})	g (p)	p (π)	LB	UB	Sec Best	Sec Tot.	# Iter
att48	1	1	1	6.15	6.89	0.34	3600.00	192
att48	1	1	2	6.43	7.09	0.37	3600.00	224
att48	1	2	1	428.60	428.60	0.36	77.28	8
att48	1	2	2	497.58	497.58	0.36	101.91	12
att48	2	1	1	10.95	12.71	0.10	3600.00	145
att48	2	1	2	10.97	12.86	0.14	3600.00	297
att48	2	2	1	817.22	817.22	0.11	6.92	3
att48	2	2	2	873.69	873.69	0.13	133.60	12
att48	3	1	1	13.59	16.08	0.15	3600.00	328
att48	3	1	2	13.23	16.18	0.12	3600.00	209
att48	3	2	1	1011.11	1011.11	0.09	254.07	11
att48	3	2	2	1130.66	1130.66	0.14	220.99	14
bayg29	1	1	1	3.20	3.20	0.07	13.74	11
bayg29	1	1	2	3.70	3.70	0.09	1.29	2
bayg29	1	2	1	215.22	215.22	0.08	2.71	3
bayg29	1	2	2	266.09	266.09	0.08	1.47	2
bayg29	2	1	1	7.42	7.42	0.03	81.78	54
bayg29	2	1	2	8.23	8.23	0.04	25.40	14
bayg29	2	2	1	461.90	461.90	0.03	3.89	4
bayg29	2	2	2	546.72	546.72	0.03	4.10	4
bayg29	3	1	1	10.63	10.63	0.06	926.70	446
bayg29	3	1	2	11.30	11.34	0.08	3600.00	629
bayg29	3	2	1	627.61	627.61	0.03	0.38	2
bayg29	3	2	2	713.71	713.71	0.10	0.50	2
bays29	1	1	1	3.59	3.59	0.08	1.47	4
bays29	1	1	2	3.89	3.89	0.06	5.03	5
bays29	1	2	1	223.65	223.65	0.08	1.59	2
bays29	1	2	2	268.41	268.41	0.07	1.25	2
bays29	2	1	1	7.73	7.73	0.07	79.23	50
bays29	2	1	2	8.48	8.48	0.06	410.52	109
bays29	2	2	1	481.21	481.21	0.05	3.91	3
bays29	2	2	2	573.46	573.46	0.06	2.89	3
bays29	3	1	1	10.88	10.88	0.03	33.25	43
bays29	3	1	2	11.44	11.54	0.07	3600.00	740
bays29	3	2	1	638.37	638.37	0.09	8.75	9
bays29	3	2	2	723.82	723.82	0.06	7.25	9
berlin52	1	1	1	11.05	11.22	0.32	3600.00	381
berlin52	1	1	2	11.69	11.97	0.31	3600.00	352
berlin52	1	2	1	632.52	632.52	0.33	27.88	6
berlin52	1	2	2	735.46	735.46	0.30	405.88	83
berlin52	2	1	1	14.91	16.02	0.13	3600.00	84
berlin52	2	1	2	15.35	16.60	0.16	3600.00	111
berlin52	2	2	1	907.96	907.96	0.17	1507.08	51
berlin52	2	2	2	1061.21	1061.21	0.15	15.00	2
berlin52	3	1	1	17.55	19.68	0.17	3600.00	288
berlin52	3	1	2	18.04	20.04	0.21	3600.00	300
berlin52	3	2	1	1137.64	1137.64	0.14	519.28	56
berlin52	3	2	2	1278.61	1278.61	0.14	112.81	22
brazil58	1	1	1	4.99	7.96	0.40	3600.00	23
brazil58	1	1	2	5.22	8.76	0.34	3600.00	16
brazil58	1	2	1	503.94	503.94	0.43	1717.06	9
brazil58	1	2	2	558.94	558.94	0.43	300.89	3
brazil58	2	1	1	11.91	16.99	0.20	3600.00	73
brazil58	2	1	2	11.82	17.22	0.20	3600.00	93
brazil58	2	2	1	1061.53	1063.40	0.19	3600.00	65
brazil58	2	2	2	1156.72	1162.12	0.19	3600.00	61
brazil58	3	1	1	12.34	18.93	0.35	3600.00	1407
brazil58	3	1	2	12.14	19.22	0.25	3600.00	621
brazil58	3	2	1	1296.34	1302.95	0.21	3600.00	202
brazil58	3	2	2	1423.10	1430.31	0.22	3600.00	152
burma14	1	1	1	1.57	1.57	0.01	0.20	7
burma14	1	1	2	1.25	1.25	0.01	0.20	7
burma14	1	2	1	105.02	105.02	0.01	0.05	2
burma14	1	2	2	98.14	98.14	0.01	0.03	2
burma14	2	1	1	3.15	3.15	0.01	13.77	62
burma14	2	1	2	2.55	2.55	0.01	214.89	271
burma14	2	2	1	178.15	178.15	0.01	0.52	3
burma14	2	2	2	183.82	183.82	0.01	1.11	5
burma14	3	1	1	3.81	4.10	0.04	3600.00	1738

(*continued on next page*)

Table 3 (*continued*).

Name	q (T_{max})	g (p)	p (π)	LB	UB	Sec Best	Sec Tot.	# Iter
burma14	3	1	2	3.20	3.73	0.04	3600.00	1584
burma14	3	2	1	264.64	264.64	0.01	1.81	9
burma14	3	2	2	277.73	277.73	0.01	3.20	15
dantzig42	1	1	1	5.36	5.43	0.24	3600.00	192
dantzig42	1	1	2	5.71	5.71	0.22	127.67	14
dantzig42	1	2	1	320.35	320.35	0.21	16.88	3
dantzig42	1	2	2	346.21	346.21	0.23	14.11	3
dantzig42	2	1	1	12.72	12.72	0.12	458.06	109
dantzig42	2	1	2	13.01	13.01	0.08	30.36	9
dantzig42	2	2	1	670.22	670.22	0.11	19.19	5
dantzig42	2	2	2	746.15	746.15	0.11	23.97	5
dantzig42	3	1	1	16.12	16.30	0.10	3600.00	188
dantzig42	3	1	2	16.75	16.83	0.13	3600.00	229
dantzig42	3	2	1	896.57	896.57	0.12	2.75	2
dantzig42	3	2	2	996.17	996.17	0.11	4.40	2.00
eil51	1	1	1	6.91	6.91	0.17	27.88	7
eil51	1	1	2	8.01	8.01	0.18	2.34	2
eil51	1	2	1	400.41	400.41	0.20	10.67	3
eil51	1	2	2	461.02	461.02	0.19	12.90	2
eil51	2	1	1	13.83	13.83	0.13	84.95	24
eil51	2	1	2	15.67	15.67	0.13	7.66	3
eil51	2	2	1	803.33	803.33	0.14	7.96	4
eil51	2	2	2	959.33	959.33	0.14	2.33	2
eil51	3	1	1	19.75	19.75	0.16	3107.78	287
eil51	3	1	2	21.09	21.09	0.15	91.24	20
eil51	3	2	1	1077.74	1077.74	0.15	61.07	9
eil51	3	2	2	1228.19	1228.19	0.14	215.77	31
eil76	1	1	1	10.89	10.93	0.43	3600.00	55
eil76	1	1	2	11.57	11.57	0.45	316.34	7
eil76	1	2	1	640.39	640.39	0.44	32.64	3
eil76	1	2	2	754.72	754.72	0.43	24.33	2
eil76	2	1	1	22.26	22.26	0.45	136.20	9
eil76	2	1	2	23.22	23.22	0.48	265.38	13
eil76	2	2	1	1243.77	1243.77	0.42	11.33	2
eil76	2	2	2	1434.14	1434.14	0.48	8.52	2
eil76	3	1	1	31.17	31.30	0.31	3600.00	8
eil76	3	1	2	32.42	32.42	0.31	3256.18	95
eil76	3	2	1	1693.16	1693.16	0.29	3.63	2
eil76	3	2	2	1887.55	1887.55	0.31	28.56	5
fri26	1	1	1	1.81	1.81	0.03	0.25	5
fri26	1	1	2	2.02	2.02	0.03	0.14	3
fri26	1	2	1	111.30	111.30	0.03	0.13	3
fri26	1	2	2	123.95	123.95	0.03	0.13	3
fri26	2	1	1	6.13	6.13	0.04	392.85	239
fri26	2	1	2	6.25	6.25	0.03	744.90	255
fri26	2	2	1	352.61	352.61	0.03	1.97	3
fri26	2	2	2	394.72	394.72	0.03	2.98	3
fri26	3	1	1	9.46	9.69	0.07	3600.00	686
fri26	3	1	2	10.06	10.06	0.06	870.92	262
fri26	3	2	1	552.92	552.92	0.05	30.96	26
fri26	3	2	2	624.32	624.32	0.04	42.56	26
gr17	1	1	1	3.12	3.12	0.01	0.08	2
gr17	1	1	2	3.00	3.00	0.01	0.11	2
gr17	1	2	1	179.62	179.62	0.01	0.16	2
gr17	1	2	2	188.91	188.91	0.01	0.11	2
gr17	2	1	1	4.73	4.73	0.03	631.86	1017
gr17	2	1	2	4.78	4.86	0.06	3600.00	1941
gr17	2	2	1	292.23	292.23	0.04	643.48	1017
gr17	2	2	2	314.33	314.33	0.03	886.82	1017
gr17	3	1	1	5.84	5.93	0.05	3600.00	1781
gr17	3	1	2	5.81	6.03	0.04	3600.00	1494
gr17	3	2	1	349.84	349.84	0.02	97.00	232
gr17	3	2	2	383.95	383.95	0.02	415.11	409
gr21	1	1	1	3.03	3.03	0.02	1.13	9
gr21	1	1	2	3.07	3.07	0.02	1.37	9
gr21	1	2	1	199.02	199.02	0.02	0.52	3
gr21	1	2	2	212.43	212.43	0.03	5.19	29
gr21	2	1	1	4.98	4.98	0.03	1932.78	444
gr21	2	1	2	5.23	5.23	0.06	2203.25	486
gr21	2	2	1	356.94	356.94	0.02	0.98	3

(*continued on next page*)

Table 3 (*continued*).

Name	q (T_{max})	g (p)	p (π)	LB	UB	Sec Best	Sec Tot.	# Iter
gr21	2	2	2	402.95	402.95	0.03	1.44	3
gr21	3	1	1	6.47	6.75	0.05	3600.00	1130
gr21	3	1	2	6.92	7.31	0.04	3600.00	734
gr21	3	2	1	441.93	441.93	0.02	1.08	3
gr21	3	2	2	491.52	491.52	0.02	1.88	3
gr24	1	1	1	2.77	2.77	0.04	6.10	26
gr24	1	1	2	2.99	2.99	0.04	1.53	6
gr24	1	2	1	188.73	188.73	0.04	0.25	2
gr24	1	2	2	217.23	217.23	0.04	0.30	2
gr24	2	1	1	5.55	5.72	0.06	3600.00	643
gr24	2	1	2	6.34	6.34	0.04	1023.65	364
gr24	2	2	1	357.51	357.51	0.02	1.41	3
gr24	2	2	2	413.86	413.86	0.04	1.64	3
gr24	3	1	1	8.29	8.29	0.04	438.88	238
gr24	3	1	2	8.86	8.86	0.03	190.24	90
gr24	3	2	1	460.26	460.26	0.04	1.02	2
gr24	3	2	2	519.88	519.88	0.03	1.16	2
gr48	1	1	1	6.99	7.03	0.26	3600.00	156
gr48	1	1	2	6.63	6.86	0.30	3600.00	128
gr48	1	2	1	447.91	447.91	0.29	14.32	2
gr48	1	2	2	490.81	490.81	0.24	12.39	2
gr48	2	1	1	12.94	13.43	0.12	3600.00	108
gr48	2	1	2	12.68	13.30	0.12	3600.00	253
gr48	2	2	1	841.42	841.42	0.12	4.65	2
gr48	2	2	2	899.07	899.07	0.13	5.60	2
gr48	3	1	1	16.59	17.90	0.12	3600.00	51
gr48	3	1	2	16.80	17.94	0.15	3600.00	218
gr48	3	2	1	1059.47	1059.47	0.11	24.49	5
gr48	3	2	2	1168.32	1168.32	0.12	22.98	5.00
gr96	1	1	1	3.08	9.46	1.40	3600.00	14
gr96	1	1	2	3.46	11.28	1.44	3600.00	10
gr96	1	2	1	736.82	738.18	1.40	3600.00	4
gr96	1	2	2	827.14	829.92	1.42	3600.00	5
gr96	2	1	1	7.12	20.06	0.86	3600.00	31
gr96	2	1	2	8.30	20.51	0.86	3600.00	60
gr96	2	2	1	1636.37	1646.45	0.83	3600.00	40
gr96	2	2	2	1877.96	1884.57	0.84	3600.00	56
gr96	3	1	1	10.39	25.91	0.83	3600.00	86
gr96	3	1	2	10.64	26.32	0.84	3600.00	57
gr96	3	2	1	2182.51	2196.89	0.83	3600.00	54
gr96	3	2	2	2436.16	2449.62	0.85	3600.00	71
hk48	1	1	1	3.07	3.79	0.36	3600.00	58
hk48	1	1	2	3.25	4.01	0.34	3600.00	88
hk48	1	2	1	314.08	314.08	0.34	20.98	2
hk48	1	2	2	399.08	399.08	0.39	20.28	3
hk48	2	1	1	9.73	11.50	0.13	3600.00	191
hk48	2	1	2	9.62	11.54	0.15	3600.00	224
hk48	2	2	1	765.03	765.03	0.12	9.34	2
hk48	2	2	2	862.53	862.53	0.12	11.17	2
hk48	3	1	1	13.06	15.37	0.12	3600.00	119
hk48	3	1	2	12.80	15.36	0.13	3600.00	98
hk48	3	2	1	1005.48	1005.48	0.12	23.51	4
hk48	3	2	2	1130.66	1130.66	0.14	31.12	5
pr76	1	1	1	0.00	0	0.47	0.00	1
pr76	1	1	2	0.00	0	0.45	0.00	1
pr76	1	2	1	569.64	569.64	0.48	2017.56	29
pr76	1	2	2	659.20	659.20	0.48	1001.86	17
pr76	2	1	1	0.00	1.15	0.43	3600.00	110
pr76	2	1	2	0.00	1.76	0.46	3600.00	235
pr76	2	2	1	1272.16	1283.98	0.33	3600.00	169
pr76	2	2	2	1472.13	1484.74	0.39	3600.00	229
pr76	3	1	1	0.00	1.21	0.40	3600.00	128
pr76	3	1	2	0.00	1.82	0.46	3600.00	293
pr76	3	2	1	1636.59	1668.38	0.42	3600.00	4
pr76	3	2	2	1824.80	1852.96	0.44	3600.00	30
rat99	1	1	1	11.74	11.91	1.01	3600.00	11
rat99	1	1	2	12.77	12.77	1.59	2311.63	19
rat99	1	2	1	701.74	701.74	1.55	49.10	2
rat99	1	2	2	813.39	813.39	1.65	68.45	2
rat99	2	1	1	24.97	25.21	0.93	3600.00	10
rat99	2	1	2	26.62	26.76	0.95	3600.00	10

(*continued on next page*)

Table 3 (*continued*).

Name	q (T_{max})	g (p)	p (π)	LB	UB	Sec Best	Sec Tot.	# Iter
rat99	2	2	1	1428.97	1428.97	0.88	168.67	2
rat99	2	2	2	1662.79	1662.79	0.93	673.51	3
rat99	3	1	1	37.73	37.73	0.96	2266.90	25
rat99	3	1	2	39.43	39.71	1.00	3600.00	32
rat99	3	2	1	2092.74	2092.74	0.93	110.62	3
rat99	3	2	2	2386.06	2386.06	0.98	243.16	4
st70	1	1	1	10.36	10.36	0.59	1077.92	89
st70	1	1	2	9.94	9.94	0.51	366.61	7
st70	1	2	1	516.85	516.85	0.51	277.51	3
st70	1	2	2	611.74	611.74	0.52	180.43	3
st70	2	1	1	20.22	20.35	0.32	3600.00	91
st70	2	1	2	21.03	21.03	0.31	930.50	65
st70	2	2	1	1096.71	1096.71	0.30	77.47	2
st70	2	2	2	1275.41	1275.41	0.29	57.04	3
st70	3	1	1	28.09	28.28	0.33	3600.00	21
st70	3	1	2	29.05	29.13	0.36	3600.00	117
st70	3	2	1	1499.08	1499.08	0.32	2022.50	63
st70	3	2	2	1679.98	1679.98	0.34	212.65	9
swiss42	1	1	1	6.75	6.75	0.18	53.27	16
swiss42	1	1	2	7.22	7.22	0.19	38.59	14
swiss42	1	2	1	406.74	406.74	0.19	1.81	2
swiss42	1	2	2	451.37	451.37	0.17	4.42	2
swiss42	2	1	1	12.99	12.99	0.09	0.97	3
swiss42	2	1	2	12.88	12.88	0.09	135.34	36
swiss42	2	2	1	719.47	719.47	0.09	6.49	3
swiss42	2	2	2	821.40	821.40	0.10	9.79	3
swiss42	3	1	1	16.74	16.74	0.08	35.62	13
swiss42	3	1	2	17.11	17.21	0.12	3600.00	339
swiss42	3	2	1	933.25	933.25	0.11	4.42	3
swiss42	3	2	2	1023.66	1023.66	0.11	8.67	3
ulysses16	1	1	1	2.12	2.12	0.01	18.58	124
ulysses16	1	1	2	1.85	1.85	0.01	34.80	181
ulysses16	1	2	1	172.12	172.12	0.01	0.28	3
ulysses16	1	2	2	187.11	187.11	0.01	0.44	5
ulysses16	2	1	1	3.09	3.86	0.05	3600.00	1840
ulysses16	2	1	2	3.14	3.88	0.04	3600.00	1910
ulysses16	2	2	1	280.61	280.61	0.01	0.93	4
ulysses16	2	2	2	325.51	325.51	0.01	0.63	3
ulysses16	3	1	1	3.60	5.06	0.11	3600.00	4335
ulysses16	3	1	2	3.59	4.96	0.08	3600.00	3107
ulysses16	3	2	1	319.60	321.08	0.08	3600.00	3080
ulysses16	3	2	2	350.96	352.36	0.06	3600.00	2274
ulysses22	1	1	1	2.70	2.86	0.04	3600.00	820
ulysses22	1	1	2	2.78	2.78	0.03	2006.15	501
ulysses22	1	2	1	225.70	225.70	0.02	5.07	5
ulysses22	1	2	2	241.02	241.02	0.02	5.75	5
ulysses22	2	1	1	5.70	6.81	0.04	3600.00	778
ulysses22	2	1	2	5.61	7.34	0.04	3600.00	719
ulysses22	2	2	1	423.70	423.70	0.05	64.70	45
ulysses22	2	2	2	463.07	463.07	0.04	2.28	3
ulysses22	3	1	1	6.32	8.12	0.08	3600.00	1854
ulysses22	3	1	2	6.29	8.12	0.07	3600.00	1512
ulysses22	3	2	1	466.82	468.62	0.07	3600.00	1693

Data availability

Data will be made available on request.

References

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.

Angelelli, E., Archetti, C., Vindigni, M., 2014. The clustered orienteering problem. European J. Oper. Res. 238, 404–414.

Archetti, C., Carrabs, F., Cerulli, R., 2018. The set orienteering problem. European J. Oper. Res. 267 (1), 264–272.

Archetti, C., Carrabs, F., Cerulli, R., Laureana, F., 2024. A new formulation and a branch-and-cut algorithm for the set orienteering problem. European J. Oper. Res. 314 (2), 446–465.

Archetti, C., Speranza, M.G., Vigo, D., 2014. Vehicle routing problems with profits. In: Toth, P., Vigo, D. (Eds.), Vehicle Routing: ProbLems, Methods, and Applications. In: MOS-SIAM Series on Optimization, pp. 273–298.

Chou, X., Gambardella, L.M., Montemanni., R., 2018. Monte Carlo sampling for the probabilistic orienteering problem. In: P., D., L., S. (Eds.), New Trends in Emerging Complex Real Life Problems. In: AIRO Springer Series, Vol. 1, Springer, Cham, pp. 169–177.

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

Fischetti, M., Salazar-Gonzalez, J., Toth, P., 2007. The generalized traveling salesman and orienteering problems. In: The Traveling Salesman Problem and Its Variations. Springer, pp. 609–662.

Golden, B.L., Levy, L., Vohra, R., 1987. The orienteering problem. Naval Res. Logist. 34 (3), 307–318.

Gunawan, A., Lau, H.C., Vansteenwegen, P., 2016. Orienteering problem: A survey of recent variants, solution approaches and applications. Eur. J. Oper. Res. 255 (2), 315–332.

Montemanni, R., Dell'Amico, M., 2023. Solving the parallel drone scheduling traveling salesman problem via constraint programming. Algorithms 16 (1), 40.

Montemanni, R., Dell'Amico, M., Corsini, A., 2024. Parallel drone scheduling vehicle routing problems with collective drones. Comput. Oper. Res. 163, 106514.

2024. OR-Brescia - benchmark instances. https://or-brescia.unibs.it/instances/. (Accessed 14 March 2024).

Perron, L., Didier, F., 2024. Google OR-Tools - CP-SAT. https://developers.google.com/optimization/cp/cp_solver/. (Accessed 14 March 2024).

Reinelt, G., 1991. TSPLIB – A traveling salesman problem library. ORSA J. Comput. 3 (4), 376–384.

Santini, A., Archetti, C., 2023. The hazardous orienteering problem. Networks 31 (2), 235–252.

Statista, 2024. E-commerce. https://www.statista.com/markets/413/e-commerce/. (Accessed 25 April 2024).

Tsiligirides, T., 1984. Heuristic methods applied to orienteering. J. Oper. Res. Soc. 35, 797–809.

Vansteenwegen, P., Oudheusden, D.V., 2007. The mobile tourist guide: An OR opportunity. OR Insights 20 (3), 21–27.

Vansteenwegen, P., Souffriau, W., van Oudheusden, D., 2011. The orienteering problem: A survey. European J. Oper. Res. 209 (1), 1–10.