algorithms

MDPI

Article

A Sim-Learnheuristic for the Team Orienteering Problem: Applications to Unmanned Aerial Vehicles

Mohammad Peyman [1], Xabier A. Martin [1], Javier Panadero [2] and Angel A. Juan [1,*]

[1] Research Center on Production Management and Engineering, Universitat Politècnica de València, Plaza Ferrandiz-Salvador, 03801 Alcoy, Spain; mpeyman@upv.es (M.P.); xamarsol@upv.es (X.A.M.)
[2] Department of Computer Architecture & Operating Systems, Universitat Autònoma de Barcelona, Carrer de les Sitges, 08193 Bellaterra, Spain; javier.panadero@uab.cat
* Correspondence: ajuanp@upv.es

Abstract: In this paper, we introduce a novel sim-learnheuristic method designed to address the team orienteering problem (TOP) with a particular focus on its application in the context of unmanned aerial vehicles (UAVs). Unlike most prior research, which primarily focuses on the deterministic and stochastic versions of the TOP, our approach considers a hybrid scenario, which combines deterministic, stochastic, and dynamic characteristics. The TOP involves visiting a set of customers using a team of vehicles to maximize the total collected reward. However, this hybrid version becomes notably complex due to the presence of uncertain travel times with dynamically changing factors. Some travel times are stochastic, while others are subject to dynamic factors such as weather conditions and traffic congestion. Our novel approach combines a savings-based heuristic algorithm, Monte Carlo simulations, and a multiple regression model. This integration incorporates the stochastic and dynamic nature of travel times, considering various dynamic conditions, and generates high-quality solutions in short computational times for the presented problem.

Keywords: team orienteering problem; biased randomization; learnheuristic; simheuristic

Citation: Peyman, M.; Martin, X.A.; Panadero, J.; Juan, A.A. A Sim-Learnheuristic for the Team Orienteering Problem: Applications to Unmanned Aerial Vehicles. *Algorithms* **2024**, *17*, 200. https://doi.org/10.3390/a17050200

Academic Editor: Marc Sevaux

Received: 15 April 2024
Revised: 2 May 2024
Accepted: 7 May 2024
Published: 8 May 2024

1. Introduction

The team orienteering problem (TOP) is a classic optimization problem in which a set of routes is constructed for a team of vehicles to traverse, aimed at collecting the highest possible reward from customers within the routes before a defined time limit [1]. In sectors such as search and rescue, surveillance, and logistics, addressing the TOP is crucial. One example of this could be a search and rescue mission, where a team of unmanned aerial vehicles (UAVs) must navigate through a disaster-stricken area to locate survivors and deliver supplies. While UAVs have emerged as essential assets in such missions due to their capacity to perform hazardous duties and their skill in sensing, monitoring, and navigating, the difficulties they face are difficult to overstate [2]. These challenges are multifaceted and include unpredictable environmental factors like weather or traffic conditions. Such unpredictability makes reliable behavior prediction impossible, resulting in suboptimal solutions and a reduced mission efficiency. Additionally, the natural limitations of UAVs, such as the limited battery life and cargo capacity, demand precise optimization for mission accomplishment. Despite these challenges, UAVs remain a critical technology for solving complex problems in various domains, emphasizing the importance of the TOP and its applicability to UAVs [3]. Furthermore, the integration of Internet of Things technologies stands as one of the most promising approaches for enabling UAVs to sense their surroundings and collect data. This integration facilitates UAVs to gather and exchange real-time environmental data, thereby optimizing their routes accordingly [4].

In recent years, the majority of studies focused on either deterministic or stochastic variants of the TOP. In the case of the deterministic TOP, many of the studies in the literature rely significantly on exact methods to solve this problem [5]. However, when presented with

challenges given by large-scale instances, these exact methods show limited effectiveness. On the other hand, in the case of the stochastic TOP, various studies have explored different strategies, such as stochastic programming [6], robust optimization [7], and, recently, the utilization of simheuristic approaches [8]. Furthermore, several versions of the TOP have been developed to address specific real-world scenarios, including heterogeneous vehicle capacities, customer time constraints, and unpredictable travel times and profits. For example, Kirac et al. [9] addressed the TOP with time windows and mandatory visits, Lin and Vincent [10] tackled the TOP with mandatory visits, Gunawan et al. [11] addressed the TOP with variable profits, and Panadero et al. [12] tackled the TOP with stochasticity.

Our paper focuses on a hybrid variant of the TOP, which combines deterministic, stochastic, and dynamic characteristics, addressing a gap in the scientific literature. To solve this complex problem, a new sim-learnheuristic approach is introduced, with special applications in UAVs that can be used to aid in disaster rescue. This approach combines a biased-randomized savings-based heuristic, Monte Carlo simulations, and a multiple regression model. The main objective of this novel methodology is to combine the strengths of all three components, allowing for an agile exploration of the search space to produce high-quality solutions for the presented problem. Figure 1 depicts this new variant of the TOP, where the travel times display a wide range of features. These features include certain travel times (deterministic), uncertain travel times (stochastic), and travel times that can change over time due to dynamic factors (dynamic), such as varying traffic congestion and weather conditions. This blend of characteristics results in a more precise representation of the complexities evident in real-world situations.

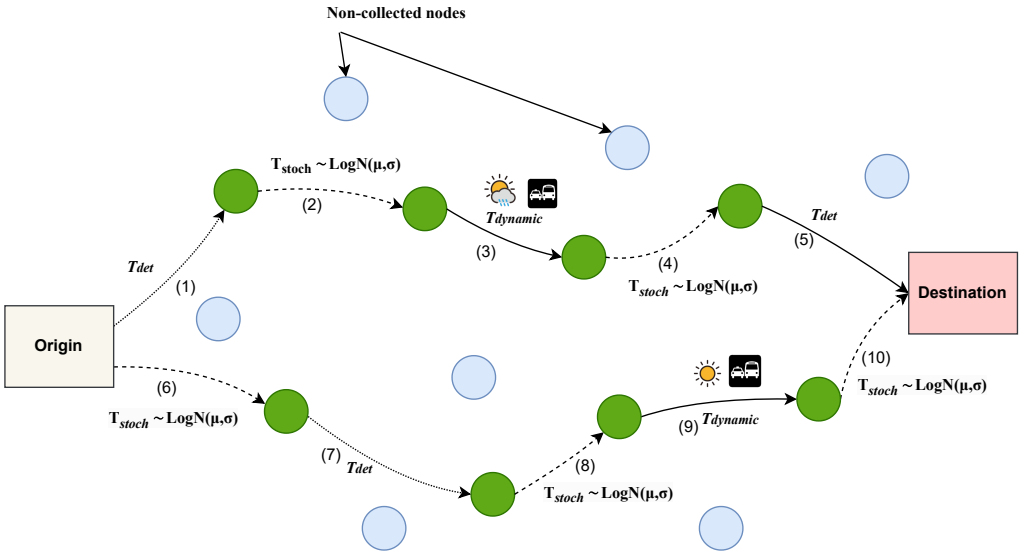

Figure 1. Hybrid variant of the TOP considering different kinds of travel times.

The main contributions of this research are as follows: (i) we present a hybrid variant of the TOP, which combines deterministic, stochastic, and dynamically changing travel times due to factors such as traffic congestion and weather conditions; and (ii) we propose a novel methodology called the sim-learnheuristic algorithm to solve this rich variant of the TOP, where the integration of three distinct components is crucial for its capability to solve increasingly complex combinatorial optimization problems. Figure 2 illustrates the integration of the three components. Initially, the metaheuristic component addresses the deterministic aspects of the problem. However, real-world scenarios often involve uncertainties. To account for this, the simulation component incorporates the stochastic nature of the environment. By combining this with the metaheuristic component, the simheuristic component effectively tackles stochastic problems. Moreover, real-world conditions are dynamic; thus, machine learning algorithms are employed to predict and

adapt to these changes. Integrating these predictions within the simheuristic framework enables it to address and solve complex dynamic problems effectively.

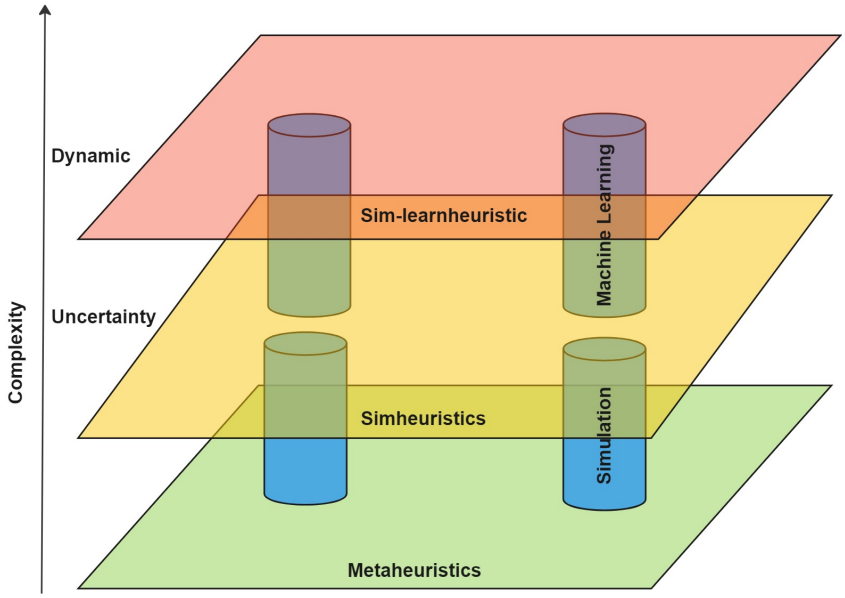

Figure 2. Integration of the sim-learnheuristic's main three components.

The remaining sections of the paper are structured as follows: Section 2 presents brief reviews of related articles. Section 3 presents the definition of the hybrid TOP, and Section 4 describes the proposed methodology of the sim-learnheuristic algorithm. Section 5 carries out a series of computational experiments to illustrate the performance of the proposed algorithm, and Section 6 presents the discussion of the obtained results. Finally, Section 7 highlights the main conclusions and future research lines of this work.

2. Related Work

In the field of optimization, simheuristics and learnheuristics are increasingly being used to solve complex combinatorial optimization problems under uncertainty. Simheuristic methods combine the advantages of both simulation and heuristics to tackle such problems. On the other hand, learnheuristics employ machine learning algorithms to improve the quality of solutions over time. Gonzalez-Neira et al. [13] proposed a hybrid simheuristic method to solve a complex multicriteria stochastic permutation flow shop problem. This method combines a greedy randomized adaptive search procedure, Monte Carlo simulations, a Pareto archived evolution strategy, and an analytic hierarchy process to handle stochastic processing times and sequence-dependent setup times. It considers quantitative criteria like earliness/tardiness and qualitative criteria like product and customer importance. The experimental results showed the significant impact of processing time distributions and coefficients of variation on decision criteria. Similarly, Caldeira and Gnanavelbabu [14] proposed a simheuristic approach to solve the stochastic flexible job shop scheduling problem. They integrated Monte Carlo simulations into a Jaya algorithm framework to minimize the expected makespan, considering uncertainties in production processes represented by random variables with known probability distributions. The computational results demonstrated its effectiveness across different variability levels using reliability-based methods, showcasing its capability in handling stochasticity in flexible job shop scheduling. Yazdani et al. [15] proposed a novel simheuristic approach to address the waste collection routing problem in the context of construction and demolition waste management. The proposed approach utilized a hybrid genetic algorithm to optimize vehicle route planning from construction projects to recycling facilities. In their research. they conducted a comparative analysis with existing approaches and demonstrated the

high performance of the proposed simheuristic algorithm. Real case studies from Sydney, Australia, were used for evaluation, contributing significantly to optimizing future waste collection problems and providing recommendations for decision-makers and practitioners. Crawford et al. [16] proposed a Q-learnheuristic framework integrating Q-Learning into metaheuristics to address the exploration–exploitation balance dilemma. This framework, applied to various metaheuristics including the Whale Optimization Algorithm and the Sine-Cosine Algorithm, enhanced the selection of operators like binarization schemes for binary combinatorial problems. This study extended the framework's applicability, demonstrating statistical improvements in both the exploration–exploitation balance and the solution quality when solving the set covering problem. Gomez et al. [17] introduced a novel approach to tackle the capacitated dispersion problem, which involves selecting elements within a network while considering their bounded service capacities. The introduction of a random Bernoulli component influences node capacities, potentially resulting in nodes with zero capacity based on environmental variables. The proposed learnheuristic approach hybridizes a heuristic algorithm with reinforcement learning, offering a promising method to handle this intricate problem variant effectively. Bullah and van Zyl [18] proposed a novel learnheuristic approach to solve a constrained multi-objective portfolio optimization problem. The authors developed a hybrid approach that combines machine learning and a metaheuristic algorithm to obtain high-quality solutions that satisfy multiple objectives, such as maximizing returns while minimizing risk and adhering to constraints such as transaction costs and diversification requirements. The learnheuristic approach involves training a neural network to predict the quality of solutions obtained by a metaheuristic algorithm and using this prediction to guide the search toward promising regions of the solution space.

Focusing on the TOP, several articles have been published in recent years. Notably, Tricoire et al. [19] employed an adaptive algorithm based on the Markov decision process to decide whether to continue the route or take a shortcut given a specific deadline. Mufalli et al. [20] discussed sensor assignment and routing for UAVs to maximize intelligence gathering, considering constraints like a limited battery and sensor weight. They proposed a detailed plan using mathematical models and local search strategies, with column generation for efficiency in larger missions. Likewise, Saeedvand et al. [21] presented a hybrid technique for solving the TOP in rescue operations using humanoid robots, aiming to optimize energy, task completion, and decision-making through a learning algorithm and an evolutionary multi-objective approach. Schmitt-Ulms et al. [22] presented a machine learning approach to solving a stochastic TOP with time windows. The authors used a reinforcement learning algorithm, specifically the Q-learning algorithm, to learn how to make routing decisions for a deliveryman facing stochastic travel times and customer demands. Also, Lee and Ahn [23] presented a data-efficient deep reinforcement learning (DRL) approach to solve a multi-start TOP for UAV mission re-planning. The authors propose a method that learns to adapt to changing conditions by iteratively updating a policy network through experience gained from previous UAV mission planning problems. Xu et al. [24] discussed a novel application of electric vehicles in TOP systems, focusing on charging energy-critical sensors using mobile chargers. Due to charger limitations, visiting all sensors was restricted, and costs were assigned considering charger energy consumption and travel. Different vehicle types and their capabilities were also considered, with sensors possibly visited by multiple vehicles, affecting profit margins. An approximation algorithm was proposed initially and later refined for handling vehicles of the same type. In contrast, Sundar et al. [25] introduced a concurrent multi-threaded branch-and-price algorithm with acceleration techniques for the TOP involving fixed-wing drones. They addressed the kinematic constraints limiting drones' maneuverability, particularly quick turns and minimum turn radii. While their approach achieved optimal solutions for most cases, its reliance on exact methods hindered real-time solutions, crucial for dynamic systems. Wang et al. [26] focused on the dynamic and stochastic orienteering problem in autonomous transportation systems. They introduced self-adaptive heuristic algorithms to

optimize task execution for intelligent agents, maximizing rewards and ensuring timely returns to the starting location within a limited time frame. The orienteering problem enables agents to selectively visit spots and optimize routing sequences for reward maximization. The study proposed a hybrid simulated annealing–tabu search algorithm for initial routing plans and three multi-stage optimization strategies for real-time adjustments. Simulation experiments demonstrated the effectiveness of the algorithm, improving the solution quality and ensuring timely arrivals for agents. Elzein and Di Caro [27] proposed a novel multi-stage metaheuristic algorithm to efficiently tackle large orienteering problem instances. The metaheuristic partitions a potentially large set of candidate sites into smaller clusters, where a solver is used to find near-optimal solutions within a bounded time. These solutions are then merged and optimized to provide a final high-quality solution. The metaheuristic significantly improves the computation time without a substantial loss in solution quality, making it suitable for online applications in robotics, logistics, and transportation scenarios with dynamic environments and a large number of candidate points. The results demonstrated the effectiveness and efficiency of the proposed approach over large benchmark instances. Le et al. [28] addressed the dynamic orienteering problem based on a VNS algorithm. The algorithm efficiently adapted existing solutions in a dynamic environment. Experimental comparisons were made with two other improvement heuristics using benchmark instances and road networks. This study evaluated the algorithm's performance, considering the impact of dynamic node values and budget changes. Fang et al. [29] investigated the use of UAVs for monitoring landslide-prone areas by solving a TOP with mandatory visits that maximize data collection with multiple UAVs. The authors incorporated a novel algorithm such as a large neighborhood search with a neural network heuristic for an enhanced efficiency and solution quality in both small- and large-scale scenarios. Lastly, Juan et al. [30] provided a hybrid methodology that combined simulation and reinforcement learning to effectively manage a vehicle's battery under dynamic settings. This approach demonstrated considerable advantages over non-informed decisions, showing its potential for routing plans that account for multiple dynamic elements such as weather, congestion, and battery conditions.

Most of the reviewed papers are summarized in Table 1, which discuss deterministic, stochastic, and dynamic versions of the TOP. This paper considers the combination of stochastic and dynamic scenarios of the TOP, and introduces a novel sim-learnheuristic methodology designed to address these combined scenarios.

Table 1. Summary of reviewed papers.

Paper	Year	Proposed Approach	Problem Domain	Key Contributions
Gonzalez-Neira et al. [13]	2021	Combined approach using GRASP, MSC, PAES, and AHP	Permutation flow shop problem	GRASP, simulation-based optimization, PAES, AHP
Yazdani et al. [15]	2021	Hybrid genetic algorithm combined with Monte Carlo simulation	Waste collection routing problem	Simheuristic approach to address a real case study of waste collection planning
Crawford et al. [16]	2021	Combination of Q-Learning with metaheuristics to address the exploration–exploitation dilemma	Set covering problem	Exploration–exploitation balance, Q-learnheuristics framework
Gomez et al. [17]	2023	Combination of heuristics with reinforcement learning	Capacitated dispersion problem	Learnheuristic algorithm using reinforcement learning
Bullah and van Zyl [18]	2023	Hybrid approach combining machine learning and metaheuristic optimization	Constrained portfolio optimization	Neural network prediction, guided search in solution space
Tricoire et al. [19]	2010	Adaptive algorithm based on a Markov decision process	TOP	Decision-making based on deadlines
Mufalli et al. [20]	2012	Combination of a mathematical method with local search strategies	TOP	Efficient algorithm for the TOP in critical situations
Saeedvand et al. [21]	2020	Hybrid learning algorithm with an evolutionary multi-objective approach	TOP with time windows	Hybrid learning algorithm for dynamic decision-making
Schmitt-Ulms et al. [22]	2022	Reinforcement learning using the Q-learning algorithm	Stochastic TOP	Machine learning approach for routing decisions in the TOP
Lee and Ahn [23]	2024	Data-efficient DRL approach	Multi-start TOP	Adaptive policy network through DRL for UAV mission re-planning
Xu et al. [24]	2020	An approximation algorithm refined for handling vehicles of the same type	TOP	Application of electric vehicles, focusing on charging energy-critical sensors using mobile chargers
Sundar et al. [25]	2022	Concurrent multi-threaded branch-and-price algorithm with acceleration techniques	TOP	TOP variant involving fixed-wing drones with kinematic constraints
Wang et al. [26]	2023	Self-adaptive heuristic algorithms	Dynamic and stochastic orienteering problem	Introduced algorithms to boost adaptability and efficiency in autonomous transportation.
Elzein and Di Caro [27]	2022	Multi-stage metaheuristic	Orienteering problem	Merges and optimizes solutions for a high-quality parallelizable final path

Table 1. *Cont.*

Paper	Year	Proposed Approach	Problem Domain	Key Contributions
Le et al. [28]	2021	Improve the heuristic based on a VNS algorithm	Dynamic orienteering problem	Proposed an efficient VNS-based heuristic for handling dynamic environments
Fang et al. [29]	2023	Large neighborhood search algorithm embedding a neural network heuristic	Dynamic TOP	Verified the algorithm on synthetic datasets and a real-world large-scale case
Juan et al. [30]	2023	Hybrid methodology combining simulation with reinforcement learning	Dynamic TOP	Efficient routing plans that significantly outperform non-informed decisions

3. Problem Definition

The TOP is an NP-hard optimization problem [1], with a formal definition on a directed graph $G = (V, A)$. The graph has $V = \{0, 1, 2, \ldots, n+1\}$ nodes, where the origin depot is node 0, the destination depot is node $n+1$, and $N = \{1, \ldots, n\}$ are the intermediate nodes. The edges in $A = \{(i, j) | i, j \in V, i \neq j\}$ represent the connections between the nodes. We consider a set of homogeneous UAVs denoted as D, and each UAV $d \in D$ begins its journey from the origin depot, serves a subset of intermediate nodes, and finally reaches the destination depot. The objective is to maximize the total reward collected by all UAVs. Following the mixed-integer linear programming model for the TOP proposed by [12], the hybrid version of the problem, where the travel times are either deterministic, stochastic, or dynamic, can be expressed formally as follows:

$$\max \sum_{d \in D} \sum_{(i,j) \in A} u_j x_{ij}^d \tag{1}$$

where the objective is to maximize the aggregated deterministic reward of visited nodes over the set of edges. For each visited node in the route, it yields a reward $u_i \geq 0$. Note that these rewards are zero for both the origin and destination depots, while they hold strictly positive values for the customers. More specifically, for each edge $(i, j) \in A$ and each UAV $d \in D$, we consider the binary variable x_{ij}^d, which is equal to 1 if UAV d traverses through edge (i, j), and takes the value 0 otherwise.

Below are the provided constraints with their explanations. In the course of the tour, each point is visited at most once to ensure that no point is revisited (2).

$$\sum_{d \in D} \sum_{i \in V} x_{ij}^d \leq 1 \quad \forall j \in N \tag{2}$$

The variable y_i^d is introduced to indicate the position of node i in the tour made by vehicle d, and constraint (3) makes sure that there are no sub-tours.

$$y_i^d - y_j^d + 1 \leq (1 - x_{ij}^d) \mid N \mid \quad \forall i, j \in N, \ \forall d \in D \tag{3}$$

Constraint (4) states that the total travel time of each vehicle should not be more than its threshold (T_{max}), since each UAV $d \in D$ starts its route and can only serve some intermediate nodes due to the limited travel time. The total travel time is the sum of the travel times of deterministic, dynamic, and stochastic edges, which are included in A_{det}, A_{dyn}, and A_{stoch}, respectively. These are disjoint sets that include all the edges of set A. For each edge $(i, j) \in A_{det}$, we assume that the travel time of each edge is deterministic and predefined ($t_{ij} = t_{ji} > 0$). Similarly, for each edge $(i, j) \in A_{stoch}$, we assume the travel time T_{ij} is stochastic and can be modeled using a probability distribution function. Lastly, for each edge $(i, j) \in A_{dyn}$, we assume the travel time $\hat{t}_{ij}$ is dynamic, which depends on various factors such as weather and traffic conditions. Since the stochastic and dynamic travel times bring in a level of uncertainty, this constraint introduces a probabilistic approach to account for such unpredictability in travel times. The parameter γ, which ranges between 0 and 1, represents a threshold that defines the acceptable level of probability for keeping the travel time within the specified travel limit.

$$P\left(\sum_{(i,j) \in A_{det}} t_{ij} x_{ij}^d + \sum_{(i,j) \in A_{dyn}} \hat{t}_{ij} x_{ij}^d + \sum_{(i,j) \in A_{stoch}} T_{ij} x_{ij}^d \leq T_{max} \right) \geq \gamma \quad \forall d \in D \tag{4}$$

Constraint (5) ensures that for every arrival at a node, there is a corresponding departure from that node.

$$\sum_{i \in V} x_{ij}^d = \sum_{h \in V} x_{jh}^d \quad \forall d \in D \qquad \forall j \in N \tag{5}$$

Constraint (6) indicates that the tour always commences at the starting node 0 and ends at the ending node $n+1$.

$$\sum_{j \in N} x_{0jd} = \sum_{j \in N} x_{j(n+1)d} = 1 \qquad \forall d \in D \tag{6}$$

Constraints (7) and (8) refer to the nature of y_j^d and x_{ij}^d variables. Finally, the vehicle departs from each node it visited, except for the end node.

$$y_j^d \geq 0 \quad \forall j \in N \qquad \forall d \in D \tag{7}$$

$$x_{ij}^d \in \{0,1\} \quad \forall i,j \in A, \forall d \in D \tag{8}$$

Although the deterministic version of the TOP has been widely studied in the literature, the high degree of uncertainty in real-life applications of the TOP makes it a good candidate for our purpose. In this work, both stochastic and dynamic travel times are introduced in the TOP. Hence, we will consider that the travel time of some edges can be modeled as random variables following a theoretical probability distribution. Likewise, we will consider that other edges have uncertain travel times that have to be predicted using a multiple regression model. As far as we know, no other authors have addressed this realistic version of the TOP in the past.

4. Our Sim-Learnheuristic Approach

To effectively solve the hybrid TOP described in Section 3, we propose a novel sim-learnheuristic method that combines a biased-randomized heuristic multi-start (BR-MS) framework, Monte Carlo simulations (MCSs), and a multiple regression model. This methodology aims to combine the advantages of all elements, enabling efficient exploration of the search space for optimal solutions [31].

This sim-learnheuristic methodology extends the simheuristic approach proposed in [8] to solve the TOP with stochastic processing times. Figure 3 outlines the main components of the sim-learnheuristic methodology. The sim-learnheuristic method can be summarized in the following six main steps, which can be identified by labels S1 through S6. First, the optimization problem is initially converted to its deterministic version, where variables are replaced by their expected or most likely values, effectively eliminating uncertainty. This deterministic problem is then solved using a BR-MS framework, leading to the generation of multiple high-quality solutions. These deterministic solutions are subsequently subjected to a comprehensive evaluation, considering the various sources of uncertainty inherent in the problem. This evaluation involves a short number of MCS runs, during which random variables and dynamic elements are assigned different values based on probability distributions or machine learning models, respectively. Additionally, constraints are assessed under uncertain conditions. Descriptive statistics are computed for each solution, yielding detailed insights into their performance. The examination of these solutions informs the generation of new solutions within the BR-MS framework until a predefined stopping criterion, such as a maximum execution time, is met. Following the previous stage, a subset of top-performing solutions, referred to as 'elite' solutions, are further examined under uncertainty. This second examination involves a longer number of MCS runs than the initial evaluations. Based on this intensive examination, the solutions are ranked, and the best solution is recommended. The selection of the best solution may take into consideration measures of interest beyond just the expected value, such as variance or reliability, to ensure a more robust decision-making process.

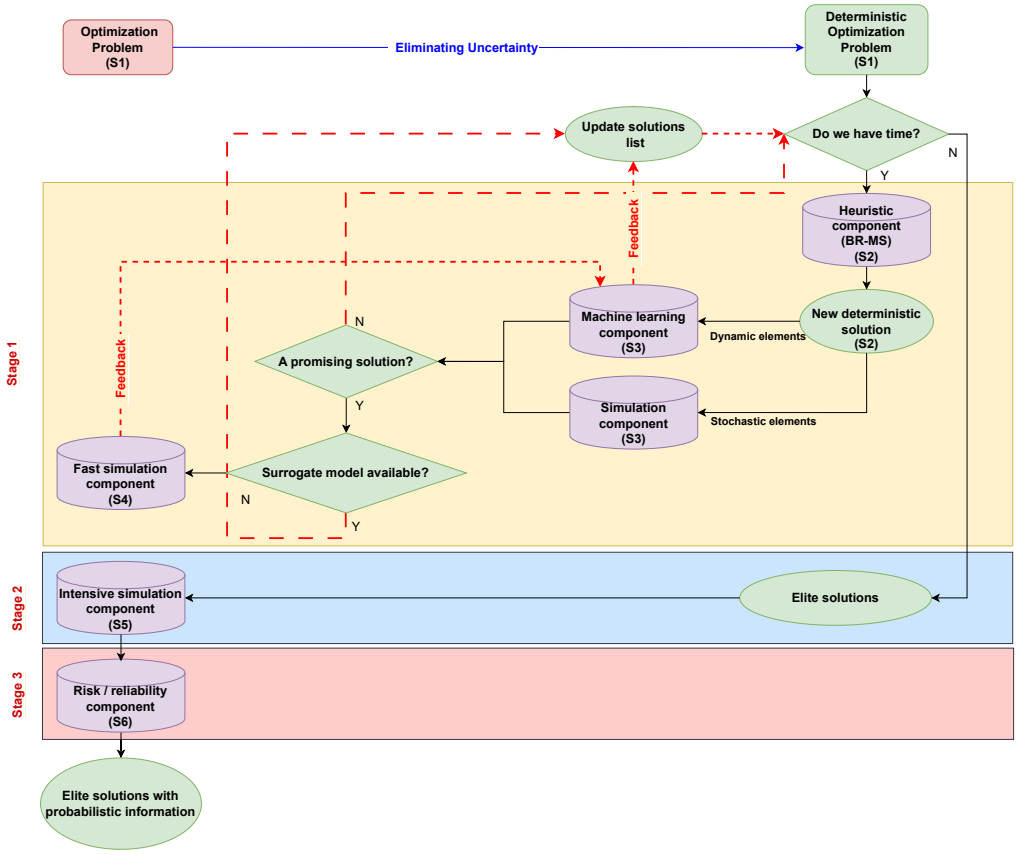

Figure 3. Schema of the sim-learnheuristic methodology.

Algorithm 1 outlines the main steps of our sim-learnheuristic approach. The sim-learnheuristic algorithm receives an instance as input, comprising the nodes, the maximum number of vehicles, and the maximum travel time. First, a feasible initial solution ($initSol$) is generated by applying a savings-based heuristic (line 1). The savings-based heuristic was first proposed by Panadero et al. [8]. Then, a short number of MCS runs (n_{short}) are performed to determine the expected reward of the initial solution (line 2). At this point, the initial solution becomes the best-known solution so far ($bestSol$) and the pool of best stochastic solutions ($eliteSols$) is initialized with this solution (lines 3–4). Next, the algorithm performs an iterative procedure to generate new solutions until a maximum execution time is reached (lines 6–16). At each iteration, the algorithm generates a new solution ($newSol$) using a biased-randomized version of the savings-based heuristic (line 7). The biased-randomized heuristic introduces a slight modification in the greedy behavior, providing a certain degree of randomness while maintaining the logic behind the savings-based heuristic. The biased-randomized version considers each element in the edges list with a probability that follows a geometric distribution with a single parameter $\beta \in (0, 1)$, which controls the relative level of greediness in the randomized behavior of the algorithm. By employing a biased-randomized version of the savings-based heuristic, multiple alternative solutions can be generated without losing the logic of the original heuristic. Next, the algorithm compares the newly generated solution to the best-known solution. If the new solution has a higher reward value (line 8), a short number of MCS runs are performed on the new solution to calculate its expected reward (line 9). In addition, if the newly generated solution obtains a higher expected reward (line 10), the best-known solution is updated and the new solution is added to the pool of elite solutions (lines 11–12). The computational time employed to generate new solutions is updated before checking if the maximum execution time (t_{max}) is reached (line 15). Once the stopping criteria are met, the algorithm conducts a refinement procedure using a larger number of MCS runs (n_{long}) over the pool of elite solutions (lines 17–22). This allows for more accurate estimations of the

solutions in the elite pool. Finally, the solution with the highest expected reward is selected from the pool of elite solutions and returned by the algorithm (line 23).

Algorithm 1 Sim-learnheuristic algorithm

1: $initSol \leftarrow$savingsHeuristic$(inputs)$
2: simulation$(initSol, n_{short})$
3: $bestSol \leftarrow initSol$
4: add$(eliteSols, bestSol)$
5: $time \leftarrow 0$
6: **while** $time \leq t_{max}$ **do**
7: $newSol \leftarrow$savingsHeuristicBR$(inputs, \beta)$
8: **if** $reward(newSol) > reward(bestSol)$ **then**
9: simulation$(newSol, n_{short})$
10: **if** $expReward(newSol) > expReward(bestSol)$ **then**
11: $bestSol \leftarrow newSol$
12: add$(eliteSols, bestSol)$
13: **end if**
14: **end if**
15: updateTime$(time)$
16: **end while**
17: **for** $sol \in eliteSols$ **do**
18: simulation(sol, n_{long})
19: **if** $expReward(sol) > expReward(bestSol)$ **then**
20: $bestSol \leftarrow sol$
21: **end if**
22: **end for**
23: **return** $bestSol$

Algorithm 2 depicts the main components of the savings-based heuristic. The heuristic starts by constructing a dummy solution (*sol*) with as many routes as nodes (line 1). Specifically, each route goes from the origin depot to a node before concluding its route at the destination depot. Notice that the constructed routes that exceed the maximum travel time are deleted from the solution. Then, the savings $s_{i,j}$ associated with each edge connecting two different nodes i and j are generated as follows:

$$s_{i,j} = \alpha\left(t_{0,j} + t_{i,n+1} - t_{i,j}\right) + (1-\alpha)\left(u_i + u_j\right) \tag{9}$$

where $t_{i,j}$ is the travel time between nodes i and j, $t_{i,n+1}$ is the travel time between node i and the destination depot, $t_{0,j}$ is the travel time between the origin depot and node j, u_i and u_j are the rewards at nodes i and j, and α is a parameter used to balance travel time and collected reward. Notably, each pair of nodes has two different savings, one for each direction or edge. The savings associated with each edge are sorted in descending order, from the highest to the lowest saving value (line 2). This sorted list of savings is iterated, and a merging process is started (lines 3–15). At each iteration, a savings edge is extracted from the list of savings, and the two routes associated with the savings edge are obtained (lines 4–6). These two routes are merged, and the resulting route is checked to see if the merge does not exceed the maximum travel time constraint (line 7). If the merge is feasible, the two routes are removed from the solution, and the merged route is inserted instead (lines 8–13). Once the merging process finishes, the routes are sorted based on their collected reward in decreasing order from higher to lower rewards (lines 16). Then, the number of routes equal to the number of vehicles in the fleet (v_{max}) constitutes the solution proposed by the heuristic (line 17). Finally, the resulting solution with the highest rewarded routes is returned by the procedure (line 18).

Algorithm 2 Savings-based heuristic

```
1: sol ←genDummySolution(inputs)
2: savingsList ←genSortedSavingsList(inputs, α)
3: while savingsList ≠ ∅ do
4:     ijEdge ←selectNext(savingsList)
5:     iRoute ←getOriginRoute(ijEdge)
6:     jRoute ←getEndRoute(ijEdge)
7:     newRoute ←mergeRoutes(iRoute, jRoute)
8:     isMergePossible ←checkMergeConditions(newRoute)
9:     if isMergePossible = true then
10:        deleteRoute(sol, iRoute)
11:        deleteRoute(sol, jRoute)
12:        addRoute(sol, newRoute)
13:    end if
14:    deleteEdge(savingsList, ijEdge)
15: end while
16: sortRoutesByProfit(sol)
17: deleteRoutesByProfit(sol, v_max)
18: return sol
```

The simulation process is depicted in Algorithm 3. The simulation procedure takes two input parameters: a solution (*sol*) and the number of simulation replications (n_{runs}) to estimate the performance of the solution under stochastic and dynamic conditions. In the procedure, a variable keeps track of the accumulated expected reward (*sumReward*), which is initialized to zero at the start of the simulation process (line 1). The simulation is run it_{max} times, and at each run, the expected reward of the solution (*solReward*) is added to the accumulated expected reward (lines 2–28). For each route in the solution, the route's expected reward (*routeReward*) is computed along with the route's expected cost (*routeCost*) (lines 4–26). If the expected route cost exceeds the maximum travel time, a penalty is applied to the route, resulting in a loss of all the collected rewards from the visited nodes (lines 22–24). To compute the route's expected reward, we iterate through the route's composing edges, and for each edge, we obtain its expected cost (lines 7–21). The expected cost of an edge is computed based on its deterministic, stochastic, or dynamic characteristics. The expected cost of a deterministic edge is predefined and specific to each particular instance (line 12). Moreover, the expected cost of a stochastic edge is sampled using a non-negative probability distribution, such as the Weibull or Log-Normal distributions, to simulate the uncertainty in the route plan (line 14). Finally, the expected cost of a dynamic edge is predicted using a multiple regression model considering the current dynamic conditions at this time (lines 16–17). Once the simulation replications are complete, the expected reward is computed as the average accumulated expected reward across all the simulation replications and is assigned as the expected reward of the solution (lines 29–30).

Algorithm 3 Simulation procedure

```
sumReward ← 0
for i ∈ {1, ..., n_runs} do
    solReward ← 0
    for route ∈ getRoutes(sol) do
        routeReward ← 0
        routeCost ← 0
        for edge ∈ getEdges(route) do
            edgeCost ← 0
            node ←getEndNode(edge)
            type ←getType(edge)
            if type = DETERMINISTIC then
                edgeCost ←getCost(edge)
            else if type = STOCHASTIC then
                edgeCost ←getStochasticValue(edge, varLevel)
            else if type = DYNAMIC then
                conditions ←getDynamicConditions(time)
                edgeCost ←getDynamicValue(edge, conditions)
            end if
            routeCost ← routeCost + edgeCost
            routeReward ← routeReward+getReward(node)
        end for
        if routeCost > routeMaxCost then
            routeReward ← 0
        end if
        solReward ← solReward + routeReward
    end for
    sumReward ← sumReward + solReward
end for
expReward ← sumReward/it_max
setExpReward(sol, expReward)
```

5. Computational Experiments

The proposed sim-learnheuristic algorithm was implemented using Julia 1.8.2 and tested on a personal computer equipped with an Intel Core i7 processor operating at 2.8 GHz and 16 GB of RAM. The algorithm's computational time was set to a time limit of 60 s for all instances. The β parameter for the geometric distribution was randomly assigned from the interval $(0.1, 0.3)$ after a quick tuning process over a random sample of instances which established a good performance. For the exploratory and refinement stages of the simulation phase, we conducted 100 and 1000 simulation runs, respectively. To validate the proposed methodology, we randomly selected 23 instances from a well-known benchmark presented next and conducted 10 separate executions for each instance, utilizing varying initial seeds for the algorithm. Among the results from these runs, we reported the best solutions based on the highest collected reward.

Given the unavailability of publicly accessible instances for the dynamic and stochastic TOP, we decided to enhance the benchmark instances introduced by Chao et al. [1] for the deterministic version of the TOP. The benchmark collection consists of 320 examples, categorized into seven distinct sets. Each example within a set is identified using the naming convention px.y.z, where x represents the set number, y denotes the number of drones, ranging from two to four depending on the specific instance, and z indicates the maximum driving range. These benchmark instances have been widely used in previous research to assess the performance of algorithms in solving the deterministic TOP [32–34]. However, for our study, we extended these instances to encompass four different scenarios: deterministic, stochastic, dynamic, and hybrid scenarios.

In the deterministic scenario, the travel time for each route is predefined. The travel time is given by the Euclidean distance defined between each pair of nodes of the benchmark instances.

In the stochastic scenario, we introduce uncertainty into the travel times between nodes. Each edge $(i, j) \in A$ in the directed graph $G = (V, A)$ is now defined by a travel time, $T_{ij} = T_{ji} > 0$, which is not deterministic but follows a best-fit probability distribution function with a mean value $E[t_{ij}] > 0$. In our computational experiments, we used the Log-Normal probability distribution to model the random travel times. The Log-Normal distribution is preferred over the Normal distribution when modeling non-negative random variables. It has two parameters, namely the location parameter μ and the scale parameter σ. These parameters can be determined based on the properties of the Log-Normal distribution considering the stochastic travel times between nodes i and j are assumed to be as follows: $E[T_{ij}] = t_{ij}$ (i.e., the travel costs of the deterministic instances), and $\mathrm{Var}[T_{ij}] = \mathrm{c} \cdot t_{ij}$ for all $i, j \in \{0, 1, 2, \ldots, n+1\}$. The parameter c serves as a design parameter enabling us to control the level of uncertainty. As c approaches zero, the results of the stochastic scenario are expected to converge with those obtained in the deterministic scenario. For our analysis, we have employed the value $c = 1$, which introduces a significant level of uncertainty as it uses the deterministic travel time as variance. A more comprehensive discussion on the implications of varying the parameter c can be found in Panadero et al. [12]. Specifically, the authors consider three different levels of uncertainty: low $(c = 0.05)$, medium $(c = 0.25)$, and large $(c = 0.75)$. In addition, the edges are categorized into two parts for the stochastic scenario following these conditions. If the node index is even, we classify the corresponding edge as stochastic. Otherwise, we classify the edge as deterministic.

In the dynamic scenario, the travel times between nodes are predicted using a multiple regression model that takes into account the current dynamic conditions. This means that instead of relying on fixed or predefined travel times, the travel times between nodes are now estimated based on the real-time or dynamic information available at the time of planning the routes. This multiple regression model takes into account external factors such as weather and congestion conditions to calculate the dynamic cost. The weather adversity level w and congestion adversity level c, ranging from 0 (low) to 1 (high), serve as input variables for the regression model. These two parameters are generated from a uniform distribution, which indicates that extreme weather and traffic conditions are as likely to occur as more standard ones. This generation matches the dynamic nature of the system, as every condition is likely to occur at any point in time. Thus, the dynamic travel times are calculated with Equation (10) as follows:

$$\hat{t}_{ij} = b_e \cdot t_{ij} + (b_1 w + b_2 c + \epsilon) \tag{10}$$

where t_{ij} represents the deterministic time of the edge connecting nodes i and j, b_e is a coefficient for edge standard time, b_1 represents the coefficient for the weather, and b_2 represents the coefficient of the congestion. In addition, the term ϵ represents the independent error term, assumed to be 0 to maintain the deterministic travel time value when the conditions are ideal (w and c parameters are 0). For our analysis, we have employed the values $b_e = 1$, $b_1 = 0.005$, and $b_2 = 0.0075$, which was designed so the dynamic travel time varies between the deterministic travel time and 1.125 times the deterministic travel time. In addition, the edges are categorized into two parts for the dynamic scenario following these conditions. If the node index is even, we classify the corresponding edge as dynamic. Otherwise, we classify the edge as deterministic.

In the hybrid scenario, we combine three distinct edge types: deterministic, stochastic, and dynamic edges. In determining the type of edge, we implement these specific conditions. If the node index is even, we classify the corresponding edge as stochastic. Otherwise, if the node index is divisible by three, we classify the edge as dynamic. If neither of these conditions is met, we assign the edge as a deterministic edge. This systematic approach ensures the accurate classification of edges within our model.

Due to the uncertain nature of travel times, there is a possibility of route failure when a UAV is unable to reach the destination depot within the designated time limit. Therefore, the reward collected by a particular UAV, r_d, is defined as follows:

$$r_d = \begin{cases} \sum\limits_{(i,j)\in A} u_i \cdot x_{ij}^d & \text{if } \sum\limits_{(i,j)\in A_{det}} t_{ij}x_{ij}^d + \sum\limits_{(i,j)\in A_{dyn}} \hat{t}_{ij}x_{ij}^d + \sum\limits_{(i,j)\in A_{stoch}} T_{ij}x_{ij}^d \leq T_{max} \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

Hence, when a UAV embarks on a route, it faces uncertainty in the time it takes to travel between nodes due to the probabilistic nature of the travel times. If the UAV fails to complete the planned route within the limited time, it forfeits all the rewards it accumulated during its journey. It is important to note that any partial rewards obtained are considered valid solely if the UAV manages to reach the destination node before the end of its travel time.

6. Results and Discussion

Table 2 shows the experimental results for each instance. The first column identifies the instances, while the remaining columns show the obtained results under the four considered scenarios. The first section of the table reports the best-known solutions for the deterministic variant of the problem obtained from the literature (*BKS*), our best-found solutions for the deterministic version of the problem (*OBD*), and the percentage gaps when both solutions are compared. The second section of the table reports the obtained results for the stochastic scenario. First, the best-found solutions for the deterministic scenario when evaluated in a stochastic environment are reported (*OBD-S*). To compute this solution, an intensive simulation process has been applied to the *OBD*. The main idea behind this process is to assess the best-found deterministic solutions under stochastic uncertainty. Next, the best-found stochastic solutions obtained using our sim-leanheuristic approach are reported (*OBS*). This approach considers the stochastic elements during the search for a high-quality solution. The next section of the table shows the obtained results for the dynamic scenario. First, the best-found solutions for the deterministic version when evaluated under a dynamic environment are shown (*OBD-Dy*). Similarly, the best-found solutions for the dynamic scenario are reported (*OBDy*). Lastly, the results for the hybrid scenario are reported. First, the best-found solutions for the deterministic version when evaluated in a hybrid scenario are shown (*OBD-H*). Next, the best-found solutions for the hybrid scenario are reported (*OBH*).

It is noteworthy that for each instance, the best-found solutions obtained by our sim-learnheuristic approach are consistently closer to the *BKS* when compared with the best-found deterministic solutions simulated under the corresponding types of uncertainty. In other words, our best-found solutions for the deterministic version of the problem are sub-optimal for the different stochastic, dynamic, and hybrid scenarios. For instance, considering the dynamic scenario, when simulating the *OBD*, solutions often exhibit significant fluctuations. This is exemplified by the *p5.3.z* instance, as exemplified by p5.3.z, where *OBD-Dy* yields an expected reward of 2.34 and *OBDy* yields an expected reward of 1525.00. This difference can be attributed to the way our sim-learnheuristic approach explores the search space for solutions. Specifically, our novel methodology considers this uncertainty throughout the search for high-quality solutions. In other words, incorporating uncertainty elements in the search process leads to superior results compared to solely solving a deterministic version of the problem and then applying the solution in a real-life scenario with dynamic uncertainty. This fundamental difference serves as the key factor behind the substantial disparities observed between the best-found solutions obtained using our sim-learnheuristic method and the results obtained simulating the best-found deterministic solutions under the different uncertainty scenarios.

Table 2. Experimental results for the different scenarios.

Instances	Deterministic Scenario			Stochastic Scenario		Dynamic Scenario		Hybrid Scenario	
	BKS (1)	**OBD (2)**	**GAP% (1)–(2)**	**OBD-S**	**OBS**	**OBD-Dy**	**OBDy**	**OBD-H**	**OBH**
p1.2.r	280	275	1.78	166.60	167.74	33.24	36.84	150.86	159.23
p1.2.h	110	110	0.00	70.21	73.96	84.70	105.00	68.46	72.72
p1.2.p	250	245	2.00	134.29	148.34	0.00	57.20	130.55	143.48
p1.4.k	100	100	0.00	69.85	71.01	52.48	58.62	68.47	70.60
p2.3.h	165	165	0.00	104.68	105.01	72.72	76.56	102.30	105.06
p2.3.e	120	120	0.00	79.83	83.77	58.02	63.10	80.69	80.72
p2.3.f	120	120	0.00	91.09	91.55	120.00	120.00	90.48	90.84
p2.2.i	230	230	0.00	173.92	176.50	146.62	147.21	163.42	172.80
p2.4.j	120	120	0.00	91.09	91.55	120.00	120.00	90.48	90.84
p3.3.j	380	380	0.00	258.12	267.25	130.36	130.63	262.35	262.82
p3.3.o	590	590	0.00	346.61	379.20	129.58	309.12	339.07	359.68
p3.2.a	90	90	0.00	58.10	61.26	54.69	56.39	58.21	58.62
p3.2.d	220	220	0.00	142.13	155.44	105.25	193.86	137.19	141.27
p3.4.j	310	310	0.00	200.25	225.31	159.61	251.68	194.09	222.26
p3.4.k	350	350	0.00	247.31	257.19	218.01	221.11	242.13	249.29
p3.4.i	270	270	0.00	170.68	178.18	90.48	117.33	166.88	173.25
p5.3.z	1635	1620	0.91	943.60	1235.26	2.34	1525.00	850.78	1240.57
p5.3.o	870	870	0.00	495.03	520.64	0.00	270.00	485.46	487.78
p5.2.i	480	450	6.25	297.43	306.67	90.22	424.38	265.05	292.62
p6.2.g	660	660	0.00	431.97	436.26	417.12	446.49	406.56	411.51
p6.2.f	588	588	0.00	336.04	361.62	1.18	2.65	327.81	332.51
p7.2.c	101	101	0.00	80.75	82.03	48.05	48.16	66.18	66.20
p7.4.e	123	123	0.00	113.46	113.81	113.90	115.47	76.49	77.01
Average:	354.86	352.47	0.47	221.87	243.02	97.76	212.90	209.73	233.10

Moreover, if we compare the computed averages for the different scenarios, we observe that we can classify the four different scenarios based on their uncertainty levels. Firstly, the deterministic scenario can be considered a reference scenario with perfect information (i.e., without uncertainty) for the expected reward under different uncertainty conditions. Conversely, in the stochastic scenario, we observe a lower uncertainty level, resulting in an average expected reward of 221.87 and 243.02 for the *OBD-S* and *OBS*, respectively. This can be attributed to half of the edges' travel times being stochastic in nature, while the other half being deterministic. Similarly, for the hybrid scenario, we can observe a medium level of uncertainty which results in an average expected reward of 209.73 and 233.10 for the *OBD-H* and *OBH*, respectively. Lastly, considering the dynamic scenario, it demonstrates the highest level of uncertainty, driven by the dynamic conditions that result in highly variable travel times. The resulting average expected reward is 97.76 and 243.02 for the *OBD-Dy* and *OBDy*, respectively. The dynamic scenario comprises half the dynamic and half the deterministic travel time, while the hybrid scenario consists of roughly one-third of the dynamic and one-third of the stochastic travel times. In other words, the combined uncertainty of the travel times in the dynamic scenario surpasses that of the hybrid and stochastic scenarios.

Figure 4 shows an overview of Table 2 which details the performance of our sim-learnheuristic approach for all considered scenarios. The horizontal and vertical axes represent the four uncertainty scenarios and the percentage gap obtained with respect to the *BKS* reported in the literature, respectively. Median values are represented by triangles and lines, while outliers are presented by circles. The average percentage gap for the *OBD-S* is 33.48, while the average percentage gap for the *OBS* is 30.16. This means that, on average, the *OBS* solutions are closer to the *BKS* solutions in the stochastic scenario. In the hybrid scenario, if we compare the percentage gaps of the *OBD-H* with the *OBH*, we can see that the *OBH* outperforms the *OBD-H* and is closer to the *BKS*. However, the average gap for *OBD-Dy* is 55.12. This very high gap indicates that the deterministic approach

does not adapt well to the changing dynamics of the system and results in considerably worse outcomes.

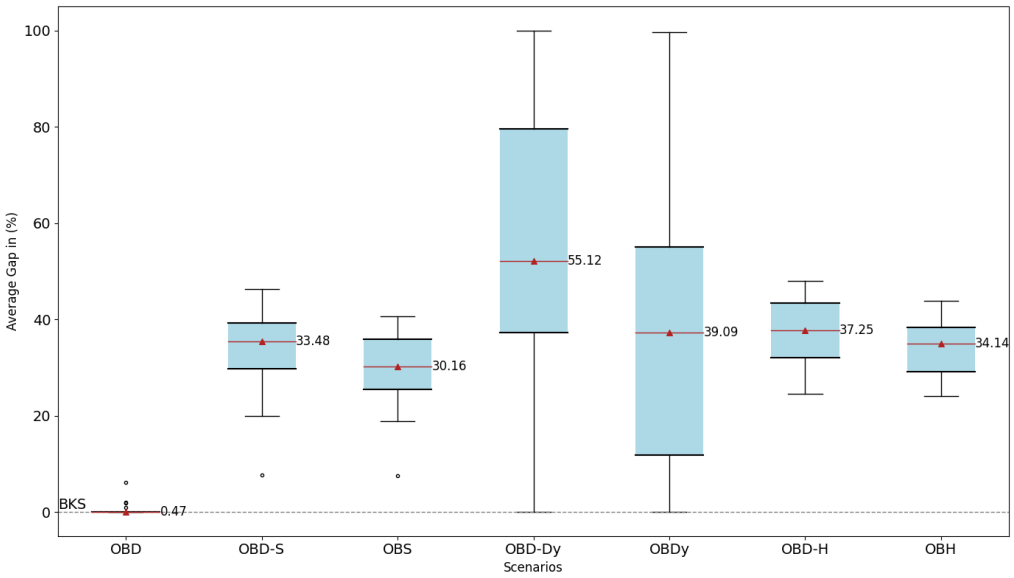

Figure 4. Gaps with respect to *BKS* for the different scenarios.

7. Conclusions and Future Work

This paper presents a hybrid version of the TOP encompassing deterministic, stochastic, and dynamic travel times. To address this problem, a novel sim-learnheuristic methodology is proposed, effectively combining a savings-based heuristic algorithm, Monte Carlo simulations, and a multiple regression model. By integrating stochastic and dynamic components in the sim-learnheuristic methodology, this approach captures the impact of randomness and variability, obtaining high-quality solutions in scenarios with uncertainty. This combination allows for the modeling of scenarios with both stochastic and dynamic components simultaneously. The computational experiments reveal that relying on the best solutions found for the deterministic version of the problem can lead to high variance and unreliability in realistic scenarios with uncertainties. These uncertainties can result in increasing travel times due to factors like traffic congestion and weather conditions. In contrast, the sim-learnheuristic approach is able to find high-quality solutions in uncertainty scenarios as it incorporates the stochastic and dynamic components in the search for solutions that maximize the collected rewards. This approach ensures the generated solutions are both reliable and robust, even in uncertain conditions with stochastic and dynamic travel times.

Moving forward, our future work encompasses two key aspects. Firstly, we aim to solve the multi-depot variant of the TOP. Secondly, we intend to enhance the realism of our approach by employing discrete event simulations and more intricate machine learning models. This will enable us to account for diverse interactions and complex environmental conditions, such as synchronization between various vehicles or UAV battery levels.

Author Contributions: Conceptualization, M.P. and A.A.J.; methodology, M.P. and X.A.M.; software, M.P. and X.A.M.; validation, M.P., X.A.M. and J.P.; writing—original draft preparation, M.P., X.A.M., J.P. and A.A.J.; writing—review and editing, A.A.J.; supervision, A.A.J. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially funded by the Spanish Ministry of Science and Innovation (PRE2020-091842, PID2022-138860NB-I00, RED2022-134703-T) and the Horizon Europe program (HORIZON-CL4-2022-HUMAN-01-14-101092612).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Chao, I.M.; Golden, B.L.; Wasil, E.A. The team orienteering problem. *Eur. J. Oper. Res.* **1996**, *88*, 464–474. [CrossRef]
2. Gupta, A.; Afrin, T.; Scully, E.; Yodo, N. Advances of UAVs toward future transportation: The state-of-the-art, challenges, and opportunities. *Future Transp.* **2021**, *1*, 326–350. [CrossRef]
3. Bayliss, C.; Juan, A.A.; Currie, C.S.; Panadero, J. A learnheuristic approach for the team orienteering problem with aerial drone motion constraints. *Appl. Soft Comput.* **2020**, *92*, 106280. [CrossRef]
4. Poudel, S.; Moh, S. Hybrid path planning for efficient data collection in UAV-aided WSNs for emergency applications. *Sensors* **2021**, *21*, 2839. [CrossRef] [PubMed]
5. Gunawan, A.; Lau, H.C.; Vansteenwegen, P. Orienteering problem: A survey of recent variants, solution approaches and applications. *Eur. J. Oper. Res.* **2016**, *255*, 315–332. [CrossRef]
6. Evers, L.; Glorie, K.; Van Der Ster, S.; Barros, A.I.; Monsuur, H. A two-stage approach to the orienteering problem with stochastic weights. *Comput. Oper. Res.* **2014**, *43*, 248–260. [CrossRef]
7. Yu, Q.; Cheng, C.; Zhu, N. Robust team orienteering problem with decreasing profits. *INFORMS J. Comput.* **2022**, *34*, 3215–3233. [CrossRef]
8. Panadero, J.; Juan, A.A.; Bayliss, C.; Currie, C. Maximising reward from a team of surveillance drones: A simheuristic approach to the stochastic team orienteering problem. *Eur. J. Ind. Eng.* **2020**, *14*, 485–516. [CrossRef]
9. Kirac, E.; Gedik, R.; Oztanriseven, F. Solving the team orienteering problem with time windows and mandatory visits using a constraint programming approach. *Int. J. Oper. Res.* **2023**, *46*, 20–42. [CrossRef]
10. Lin, S.W.; Vincent, F.Y. Solving the team orienteering problem with time windows and mandatory visits by multi-start simulated annealing. *Comput. Ind. Eng.* **2017**, *114*, 195–205. [CrossRef]
11. Gunawan, A.; Ng, K.M.; Kendall, G.; Lai, J. An iterated local search algorithm for the team orienteering problem with variable profits. *Eng. Optim.* **2018**, *50*, 1148–1163. [CrossRef]
12. Panadero, J.; Juan, A.A.; Ghorbani, E.; Faulin, J.; Pagès-Bernaus, A. Solving the stochastic team orienteering problem: Comparing simheuristics with the sample average approximation method. *Int. Trans. Oper. Res.* **2023**, *31*, 3039–3060. [CrossRef]
13. Gonzalez-Neira, E.M.; Montoya-Torres, J.R.; Jimenez, J.F. A multicriteria simheuristic approach for solving a stochastic permutation flow shop scheduling problem. *Algorithms* **2021**, *14*, 210. [CrossRef]
14. Caldeira, R.H.; Gnanavelbabu, A. A simheuristic approach for the flexible job shop scheduling problem with stochastic processing times. *Simulation* **2021**, *97*, 215–236. [CrossRef]
15. Yazdani, M.; Kabirifar, K.; Frimpong, B.E.; Shariati, M.; Mirmozaffari, M.; Boskabadi, A. Improving construction and demolition waste collection service in an urban area using a simheuristic approach: A case study in Sydney, Australia. *J. Clean. Prod.* **2021**, *280*, 124138. [CrossRef]
16. Crawford, B.; Soto, R.; Lemus-Romani, J.; Becerra-Rozas, M.; Lanza-Gutiérrez, J.M.; Caballé, N.; Castillo, M.; Tapia, D.; Cisternas-Caneo, F.; García, J.; et al. Q-learnheuristics: Towards data-driven balanced metaheuristics. *Mathematics* **2021**, *9*, 1839. [CrossRef]
17. Gomez, J.F.; Uguina, A.R.; Panadero, J.; Juan, A.A. A Learnheuristic Algorithm for the Capacitated Dispersion Problem under Dynamic Conditions. *Algorithms* **2023**, *16*, 532. [CrossRef]
18. Bullah, S.; van Zyl, T.L. A Learnheuristic Approach to A Constrained Multi-Objective Portfolio Optimisation Problem. In Proceedings of the 2023 7th International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence, Virtual, 23–24 April 2023; pp. 58–65.
19. Tricoire, F.; Romauch, M.; Doerner, K.F.; Hartl, R.F. Heuristics for the multi-period orienteering problem with multiple time windows. *Comput. Oper. Res.* **2010**, *37*, 351–367. [CrossRef]
20. Mufalli, F.; Batta, R.; Nagi, R. Simultaneous sensor selection and routing of unmanned aerial vehicles for complex mission plans. *Comput. Oper. Res.* **2012**, *39*, 2787–2799. [CrossRef]
21. Saeedvand, S.; Aghdasi, H.S.; Baltes, J. Novel hybrid algorithm for Team Orienteering Problem with Time Windows for rescue applications. *Appl. Soft Comput.* **2020**, *96*, 106700. [CrossRef]
22. Schmitt-Ulms, F.; Hottung, A.; Sellmann, M.; Tierney, K. Learning to solve a stochastic orienteering problem with time windows. In Proceedings of the International Conference on Learning and Intelligent Optimization, Milos Island, Greece, 5–10 June 2022; Springer: Cham, Switzerland, 2022; pp. 108–122.
23. Lee, D.H.; Ahn, J. Multi-start team orienteering problem for UAS mission re-planning with data-efficient deep reinforcement learning. *Appl. Intell.* **2024**, *54*, 4467–4489. [CrossRef]
24. Xu, W.; Xu, Z.; Peng, J.; Liang, W.; Liu, T.; Jia, X.; Das, S.K. Approximation algorithms for the team orienteering problem. In Proceedings of the IEEE INFOCOM 2020—IEEE Conference on Computer Communications, Toronto, ON, Canada, 6–9 July 2020; pp. 1389–1398.
25. Sundar, K.; Sanjeevi, S.; Montez, C. A branch-and-price algorithm for a team orienteering problem with fixed-wing drones. *EURO J. Transp. Logist.* **2022**, *11*, 100070. [CrossRef]

26. Wang, B.; Bian, Z.; Mansouri, M. Self-adaptive heuristic algorithms for the dynamic and stochastic orienteering problem in autonomous transportation system. *J. Heuristics* **2023**, *29*, 77–137. [CrossRef]
27. Elzein, A.; Di Caro, G.A. A clustering metaheuristic for large orienteering problems. *PLoS ONE* **2022**, *17*, e0271751. [CrossRef] [PubMed]
28. Le, H.T.; Middendorf, M.; Shi, Y. An improvement heuristic based on variable neighborhood search for a dynamic orienteering problem. In Proceedings of the Evolutionary Computation in Combinatorial Optimization: 21st European Conference, EvoCOP 2021, Held as Part of EvoStar 2021, Virtual Event, 7–9 April 2021; Springer: Cham, Switzerland, 2021; pp. 68–83.
29. Fang, C.; Han, Z.; Wang, W.; Zio, E. Routing UAVs in landslides Monitoring: A neural network heuristic for team orienteering with mandatory visits. *Transp. Res. Part E Logist. Transp. Rev.* **2023**, *175*, 103172. [CrossRef]
30. Juan, A.A.; Marugan, C.A.; Ahsini, Y.; Fornes, R.; Panadero, J.; Martin, X.A. Using Reinforcement Learning to Solve a Dynamic Orienteering Problem with Random Rewards Affected by the Battery Status. *Batteries* **2023**, *9*, 416. [CrossRef]
31. Martí, R.; Resende, M.G.; Ribeiro, C.C. Multi-start methods for combinatorial optimization. *Eur. J. Oper. Res.* **2013**, *226*, 1–8. [CrossRef]
32. Tang, H.; Miller-Hooks, E. A tabu search heuristic for the team orienteering problem. *Comput. Oper. Res.* **2005**, *32*, 1379–1407. [CrossRef]
33. Ke, L.; Archetti, C.; Feng, Z. Ants can solve the team orienteering problem. *Comput. Ind. Eng.* **2008**, *54*, 648–665. [CrossRef]
34. Dang, D.C.; Guibadj, R.N.; Moukrim, A. An effective PSO-inspired algorithm for the team orienteering problem. *Eur. J. Oper. Res.* **2013**, *229*, 332–344. [CrossRef]