Maximizing Reward from a Team of Surveillance Drones: a Simheuristic Approach to the Stochastic Team Orienteering Problem

Javier Panadero* and Angel A. Juan

IN3 – Computer Science Department,
Av. Carl Friedrich Gauss 5,
08860 Castelldefels, SPAIN
and
Euncet Business School,
08225 Terrassa, SPAIN
E-mail: jpanaderom@uoc.edu
E-mail: ajuanp@uoc.edu
*Corresponding author

Christopher Bayliss

IN3 – Computer Science Department,
Av. Carl Friedrich Gauss 5,
08860 Castelldefels, SPAIN
E-mail: cbayliss@uoc.edu

Christine Currie

University of Southampton,
Mathematical Sciences,
SO17 1BJ Southampton, UK
E-mail: Christine.Currie@soton.ac.uk

Abstract: We consider the problem of routing a team of unmanned aerial vehicles (drones) being used to take surveillance observations of target locations, where the value of information at each location is different and not all locations need be visited. As a result, this problem can be described as a stochastic team orienteering problem (STOP), in which travel times are modeled as random variables following generic probability distributions. The orienteering problem is a vehicle-routing problem in which each of a set of customers can be visited either just once or not at all within a limited time period. In order to solve this STOP, a simheuristic algorithm based on an original and fast heuristic is developed. This heuristic is then extended into a variable neighbourhood search (VNS) metaheuristic. Finally, simulation is incorporated into the VNS framework to transform it into a simheuristic algorithm, which is then employed to solve the STOP.

Keywords: simulation-optimization; unmanned aerial vehicles; team orienteering problem; simheuristics.

Biographical notes: Javier Panadero is an Assistant Professor of Simulation and High Performance Computing in the Computer Science Dept. at the

Universitat Oberta de Catalunya (Barcelona, Spain). He is also a Lecturer at the Euncet Business School. He holds a Ph.D. and a M.S. in Computer Science. His major research areas are: high performance computing and simheuristics. He has co-authored more than 30 scientific papers. His website address is http://www.javierpanadero.com, and his email address is jpanaderom@uoc.edu

Christine Currie is Associate Professor of Operational Research in Mathematical Sciences at the University of Southampton, UK, where she also obtained her Ph.D. She is Editor-in-Chief for the Journal of Simulation. Christine chaired the 9th UK Simulation Workshop, SW18. Her research interests include mathematical modeling of epidemics, Bayesian statistics, revenue management, variance reduction methods, and optimization of simulation models. Her website address is http://www.southampton.ac.uk/maths/about/staff/ccurrie.page and her email address is Christine.Currie@soton.ac.uk.

Angel A. Juan is a Full Professor of Operations Research Industrial Engineering in the Computer Science Dept. at the Universitat Oberta de Catalunya (Barcelona, Spain). Dr. Juan holds a Ph.D. in Industrial Engineering and an M.Sc. in Mathematics. He completed a predoctoral internship at Harvard University and postdoctoral internships at Massachusetts Institute of Technology and Georgia Institute of Technology. His main research interests include applications of simheuristics and learnheuristics in computational logistics and finance. He has published more than 80 articles in JCR-indexed journals. His website address is http://ajuanp.wordpress.com and his email address is ajuanp@uoc.edu.

Christopher Bayliss is a post-doctoral researcher in the ICSO group at the IN3 - Universitat Oberta de Catalunya. His main research interests include metaheuristics, simulation optimisation, revenue management, packing problems, airline scheduling, and logistics optimisation. His email address is cbayliss@uoc.edu.

1 Introduction

The team orienteering problem (TOP) aims to maximize the reward from a team of vehicles visiting a selection of nodes, where there is a constraint on either the distance each vehicle can travel or the time that they are traveling. A solution to the problem will identify which nodes to visit and the order in which to visit them. It was initially proposed by Chao et al. (1996a), and is one realistic variation of the well-known orienteering problem (Caceres-Cruz et al., 2015). While the team orienteering problem was introduced initially to describe the home fuel delivery problem (Chao et al., 1996a) and the recruiting of college football players (Butt and Ryan, 1999), its applications now extend to the routing of technicians to service customers at geographically distributed locations (Tang and Miller-Hooks, 2005b) and, in more recent years, the routing of unmanned aerial vehicles or drones to carry out observation tasks (Juan et al., 2014). Incorporating stochasticity into the travel and service times, the stochastic team orienteering problem (STOP), allows for uncertainty in journey times due to traffic conditions for conventional vehicles or weather conditions for unmanned aerial vehicles (UAVs) and randomness in the time spent servicing a customer.

We focus here on the problem of collecting as much reward as possible from visiting nodes using a fleet of drones with driving ranges limited by the time-duration of their batteries. Drones are used to take surveillance observations of different locations after a

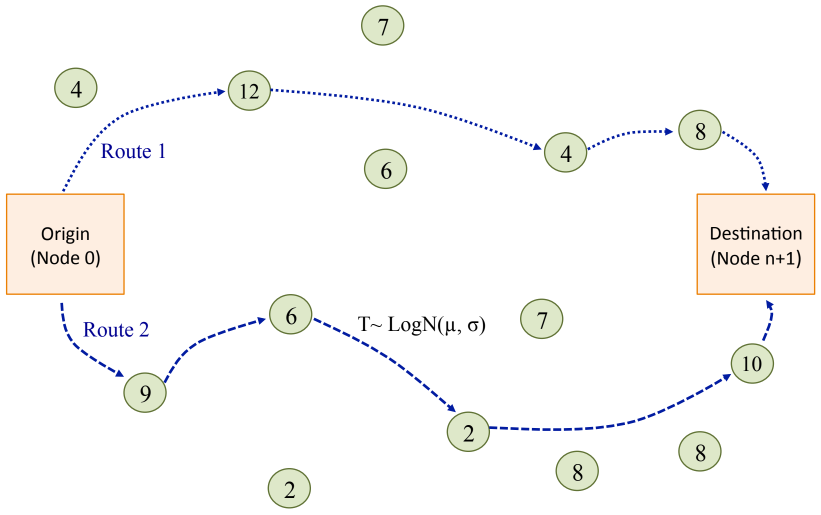

Figure 1 The team orienteering problem with stochastic travel times

natural disaster, a terrorist attack, or a humanitarian crisis. Notice that each observation can provide valuable information that can help to improve the conditions of the people affected by the event or even to save their lives by making informed decisions on the more reliable evacuation paths.

Consider the following elements: *(i)* a set of customer nodes, each of them with an associated reward score that can be collected the first time a customer is visited by any vehicle; and *(ii)* a team of m vehicles with limited driving-range capabilities. Then, the goal is to determine a set of m open routes (each of them connecting an $origin$ depot with a $destination$ depot), which maximizes the total collected reward by visiting a subset of available customers without violating the driving-range or working time constraint. This constraint can be expressed in terms of the total distance travelled or the total time spent in travelling or service, where the former perhaps better expresses the challenge of routing drones and the latter the routing of technicians. Notice that each customer can either be visited once or not visited at all. Also, due to the driving-range limitation, it is possible that not all customers can be visited.

Being an extension of the vehicle routing problem in which a subset of customers have to be selected and a set of routes covering them constructed, the TOP is also a *NP-hard* problem. Accordingly, different metaheuristic approaches have been proposed in recent years to deal with large-scale instances of the deterministic version of the problem. However, the stochastic counterpart, which considers real-life uncertainty in the form of random service and travel times, has received much less attention. This paper contributes to close this gap in the literature by analyzing a stochastic TOP variant in which travel times are modeled as random variables (Figure 1). It extends Panadero et al. (2017) by providing a significantly enhanced heuristic and simulation-optimization algorithm, as well as by adapting them to the practical example of optimizing surveillance. We also carry out a thorough test of its solutions on a selection of test instances adapted from those for the deterministic version of the TOP given in Chao et al. (1996a).

The main goal of the deterministic problem is to find a solution, in the form of a set of open routes, that maximizes the total reward. This carries over to the stochastic version where the key objective is the expected reward. Nonetheless, since solutions to the STOP are applied in a stochastic environment it is important to also consider the robustness of solutions by taking into account statistics such as the probability of violating the driving-range threshold. As part of the methodology we introduce here, simulation is used to perform a risk analysis on a subset of elite solutions proposed by the metaheuristic component.

This paper introduces and tests a novel simulation-optimization algorithm, which uses heuristics to find near-optimal solutions to the STOP. First, an original and fast heuristic is proposed, which is extended into a variable neighborhood search (VNS) metaheuristic (Hansen and Mladenović, 2001), enabling it to obtain state-of-the-art solutions for the deterministic TOP. Simulation is then incorporated into the VNS framework to transform it into a simheuristic algorithm (Juan et al., 2015). Incorporating the simulation allows for the estimation of the expected total reward and also provides feedback to the metaheuristic component, allowing it to take account of the inherent uncertainty in the problem which is used to help guide the search process.

Due to their effectiveness, simheuristic algorithms are increasingly being used for solving stochastic variants of common vehicle routing problems. For example, the stochastic inventory routing problem (Gruler et al., 2018), the stochastic waste collection problem (Gruler et al., 2017), or the stochastic arc routing problem (Gonzalez-Martin et al., 2018). This article will add to that literature, widening the range of the stochastic vehicle routing problems that simheuristics can address.

Hence, the main contributions of this paper are: *(i)* a novel and fast constructive heuristic to solve the deterministic TOP; *(ii)* a state-of-the-art VNS metaheuristic that extends this constructive heuristic to improve efficiency in solving the TOP; and *(iii)* the integration of the VNS framework with Monte Carlo simulation to efficiently deal with the STOP, including the completion of a reliability analysis on the elite solutions.

The remaining sections of this paper are structured as follows: Section 2 reviews related work on the TOP. Details of the particular problem analyzed in this paper are given in Section 3. Section 4 describes our savings-based heuristic and its extension into a VNS-based metaheuristic for solving the deterministic TOP. A first round of computational experiments is provided in Section 5, which shows the efficiency of our metaheuristic for solving the TOP. Section 6 describes our extension of the metaheuristic to a simheuristic to solve the STOP. A second round of computational experiments, this time regarding the STOP, are described in Section 7. Finally, Section 8 summarizes the highlights of this paper and proposes some future research lines.

2 Literature Review

The orienteering problem takes its name from the sport of orienteering, one variation of which involves a single runner choosing a selection of points to navigate between within a specified time limit. It was introduced to the Operations Research literature in 1987 by Golden et al. (1987), who showed that it is NP hard. Early work considered the deterministic version of the problem, setting it within the context of vehicle routing, in which one vehicle chooses the set of nodes to visit, as well as the visiting order, during a pre-defined time interval. Studies of the stochastic orienteering problem are much more recent; and Ilhan et al. (2008) were the first to incorporate stochasticity into the single vehicle orienteering problem. Allowing for uncertainty in rewards, travel and/or service times widens the applicability of the problem. Gunawan et al. (2016) provides an excellent review of the orienteering problem and its variants.

The team orienteering problem, our focus in this paper, was first introduced in Chao et al. (1996a) who extended their methodology from the single vehicle problem to consider multiple vehicles aiming to maximize their combined reward from visiting a selection of points within a given time interval. As is the case with many of the more recent authors, Chao et al. (1996a) set up the problem as a multilevel optimization, with three levels: select which points to visit; assign points to each member of the team; and finally, determine the shortest path for each team member around the points they have been assigned.

Exact solutions to the deterministic team orienteering problem have been obtained for mid-sized problems (up to 100 vertices) using an efficient column-generation algorithm (Butt and Ryan, 1999) but the vast majority of methods are based on heuristics as we discuss in what follows.

Archetti et al. (2007) propose two variants of a generalized tabu search algorithm and a variable neighborhood search algorithm to solve the deterministic TOP. The first tabu search algorithm follows a strategy which relies on exploring the set of feasible solutions, while the strategy followed by the second tabu search algorithm visits admissible, but not necessarily feasible, solutions. Their results show that, on average, the strategy that only explores feasible solutions yields higher quality results. The last algorithm proposed in Archetti et al. (2007) is a general ascending schema of a variable neighborhood search algorithm which uses the above mentioned tabu search algorithm as the local search component.

The particle swarm optimization (PSO) algorithm that Dang et al. (2013) propose consists of a swarm intelligence algorithm, which is based on the idea of simulating the collective behavior of wild animals (Kennedy and Eberhart, 1995). The proposed approach works with a population of particles – the swarm – and uses giant tours to indirectly encode particle positions. Each particle memorizes its current position, which is a representation of a solution and its best known position. The quality of a position is determined by an evaluation process based on an interval graph model. This enables more iterations of the PSO without increasing the global computational time. In each iteration of the algorithm, the position of each particle in the swarm is updated in order to find a better solution.

The multi-start simulated annealing (SA) algorithm that Lin (2013) introduces integrates an SA stage inside a multi-start procedure to reduce the possibility of getting trapped in a local optimum. The algorithm begins with a randomly-generated initial solution before going into the iterative procedure. In each iteration, the algorithm selects a new solution from the neighborhood of the current one. If the objective-function value of the new solution is better than that of the current one, the new solution replaces the current solution and the searching process is resumed. As in any other SA structure, there is also a small probability that a new solution, with a worse objective-function value, may be accepted as the new current solution.

Ferreira et al. (2014) introduce a genetic algorithm (GA) to solve the deterministic TOP, while Ke et al. (2016) describe a Pareto mimic algorithm, which uses a mimic operator to generate a new solution by imitating an incumbent solution. The method also includes a swallow operator in order to insert an infeasible node and then repair the resulting solution. In each step, the algorithm starts from the incumbent solution and a new solution is generated by the mimic operator; subsequently, a local search and the swallow operator are employed to improve it; and finally, the newly-generated solutions and the old incumbent solutions are compared to update the set of incumbent solutions according to the Pareto concept of dominance.

Despite its practical relevance, the stochastic version of the orienteering problem has only received attention relatively recently and, to the best of our knowledge, previous work has only considered the single-tour problem rather than the team orienteering problem that we describe here. Even within the single-tour problem, there is some variation in which of the three main characteristics (travel times, service times and rewards) are assumed to be stochastic, and how to deal with tours that exceed the time limit.

The original paper on the stochastic orienteering problem assumes that only the scores associated with each node are stochastic (Ilhan et al., 2008), whereas Campbell et al. (2011); Papapanagiotou et al. (2014); Verbeeck et al. (2016); Evers et al. (2014) study the case we describe here in which service times and travel times are stochastic. Service times are often incorporated into travel times rather than treated separately, a model that we follow here.

In the sport of orienteering, exceeding the time limit typically incurs a penalty proportional to the amount by which it has been exceeded. This assumption has some practical appeal in situations where a delivery driver or repair person will earn overtime payments for any delay in their return to the depot and is used in both Teng et al. (2004) and Lau et al. (2012). An alternative concept is presented in Tang and Miller-Hooks (2005a), who instead constrain the probability of exceeding the time limit to be below a threshold value. In contrast, Campbell et al. (2011) do not force the vehicle to return to a set of depots but, instead, allow it to stop at any location once the time limit is reached. Penalties are incurred if a vehicle does not manage to visit a scheduled node within the time limit. A stricter version of the constraint is employed by Evers et al. (2014) who keep the hard constraint on the tour length used in the deterministic version of the problem and abort the route if the expected return time to the depot is equal to the remaining time. A drone that exceeds its driving range before returning to the end depot can be assumed to be lost, along with all of its rewards. As a result, we use a strict policy in our algorithms whereby drones that exceed the driving range lose all of their rewards.

Whilst this work considers the stochastic team orienteering problem, methodologies for solving the stochastic orienteering problem have included a branch and bound algorithm combined with local search (Campbell et al., 2011) and simulation with local search (Lau et al., 2012). In particular, the latter use simulation to evaluate the values of tours. However, their simulation component is only used for evaluation purposes, without providing any feedback to the optimization component as we do. Also, they do not filter out solutions provided by the optimization component as our methodology does. Both articles make use of measures that combine the reward and the probability that the deadline is exceeded. Campbell et al. (2011) denote this as the *utility* whereas Lau et al. (2012) use it in a similar way to the ratios of reward and weight in a knapsack problem. Both of these works have been refined and improved, with Varakantham and Kumar (2013) using a Mixed Integer Linear Program Sample Average Approximation technique (MILP-SAA) to improve upon the results of Lau et al. (2012) and Zhang et al. (2014) extend the method of Campbell et al. (2011) to incorporate time windows for arriving at nodes.

3 Problem Description

We assume that the team is composed of m vehicles, and there is a maximum time, t_0, for completing each open route (from the origin depot to the designated end depot). The set of possible points to visit can be described by an undirected graph $G = (N, A)$, where $N = \{0, 1, 2, \ldots, n + 1\}$ is the set of nodes, including n customers, an origin depot 0, and a destination depot $n+1$; and $A = \{(i,j)/i, j \in N, i < j\}$ is the set of edges connecting them. Each node $i \in N$ has a constant reward $u_i \geq 0$ (typically, these rewards are zero for the origin and destination depots, while they are strictly positive for the customers). Rewards can only be collected on the first visit to a node. Edges have a random travel time associated with them, T_{ij} and we assume that $T_{ij} \overset{\mathcal{D}}{=} T_{ji}$, where $\overset{\mathcal{D}}{=}$ means equality in distribution.

We define a solution to the STOP to be a set M of m open routes, where each route is defined by an array of nodes starting from the origin depot and ending at the destination depot. The objective function is given by the maximization of total expected reward,

$$\max \sum_{m \in M} E[U_m], \tag{1}$$

where U_m is a random variable representing the total reward associated with route m. Since travel times are stochastic, a *route failure* might occur whenever a vehicle cannot reach the destination depot before the driving-range threshold, t_0. For our example in which drones are being used to conduct surveillance, no reward is realised if the drone does not return to the end depot before its battery life runs out. As a result, we assume that if a vehicle does not reach the end depot before the driving-range threshold is met, all of its rewards will be lost. There will be examples where this assumption does not hold. For example, when rewards can be transmitted in real time from each depot, or in the more traditional example of carrying out repairs or sales at each node, the reward will have been collected at the end of the service rather than when the vehicle returns to the end depot.

Thus, if we consider x_{ijm} as a binary decision variable which equals 1 if the edge $(i,j) \in A$ is in the route m, and 0 otherwise, the total reward accumulated in route m is computed as:

$$U_m = \left\{ \begin{array}{ll} \sum\limits_{(i,j) \in A} u_i \cdot x_{ijm} & \text{if } \sum\limits_{(i,j) \in A} T_{ij} \cdot x_{ijm} \leq t_0 \\ 0 & \text{otherwise} \end{array} \right\}, \ \forall m \in M \tag{2}$$

We now describe the constraints, the first being that each node is visited at most once by any route:

$$\sum_{m \in M} \sum_{i \in N} x_{ijm} \leq 1, \ \forall j \in N \setminus \{0, n+1\} \tag{3}$$

A route always starts at the origin depot (node 0) and ends at the destination depot (node $n+1$):

$$\sum_{j \in N} x_{0jm} = 1, \ \forall m \in M \tag{4}$$

$$\sum_{i \in N} x_{i(n+1)m} = 1, \ \forall m \in M \tag{5}$$

Finally, the vehicle always leaves each node it visits, except in the case of the origin and destination depots:

$$\sum_{i \in N} x_{ihm} - \sum_{j \in N} x_{hjm} = 0, \ \forall h \in N \setminus \{0, n+1\}, \forall m \in M \tag{6}$$

Algorithm 1 Savings-based heuristic for the TOP

```
1: sol ← generateDummySolution(Inputs)
2: savingsList ← computeSortedSavingsList(Inputs, $\alpha$)
3: while (savingsList is not empty) do
4:     arc ← selectNextArc(savingsList)
5:     iRoute ← getStartingRoute(arc)
6:     jRoute ← getClosingRoute(arc)
7:     newRoute ← mergeRoutes(iRoute, jRoute)
8:     travelTimeNewRoute ← calcRouteTravelTime(newRoute)
9:     isMergeValid ← validateMergeDrivingConstraints(travelTimeNewRoute, drivingRange)
10:    if (isMergeValid) then
11:        sol ← updateSolution(newRoute, iRoute, jRoute, sol)
12:    end if
13:    deleteEdgeFromSavingList(arc)
14: end while
15: sortRoutesByProfit(sol)
16: deleteRoutesByProfit(sol, maxVehicles)
17: return sol
```

4 From a Fast Heuristic to a VNS Metaheuristic

While the main goal of this paper is to develop a simheuristic algorithm to solve the STOP, we first develop a metaheuristic able to efficiently solve the deterministic version of the problem (TOP). This forms the first part of the simheuristic. The metaheuristic, which we describe in this section, consists of a novel constructive heuristic embedded within a VNS-metaheuristic.

4.1 A novel savings-based heuristic for the TOP

Our heuristic is inspired by the well-known savings heuristic for the vehicle routing problem (Clarke and Wright, 1964), but adapted to consider the particular characteristics of the TOP: *(i)* the origin and destination depots may be different; *(ii)* not all of the customers need to be visited; and *(iii)* the reward collected by visiting nodes must be considered during the construction of the routing plan. The goal was to design a new savings-based heuristic able to outperform the traditional one employed for solving the TOP (Tang and Miller-Hooks, 2005b).

Algorithm 1 provides a high-level description of the constructive heuristic. It starts by generating an initial dummy solution (line 1), in which one route per customer is considered so that for each customer $i \in A$, a vehicle departs from the origin depot (node 0), visits i, and then resumes its trip towards the destination depot (node $n+1$) (Figure 2a). If any route in this dummy solution does not satisfy the driving-range constraint, the associated customer is discarded from the problem, since it cannot be reached with the current fleet of vehicles. Next, we compute the 'savings' associated with each edge connecting two different customers (line 2), i.e. the benefits obtained by visiting both customers in the same route instead of using two distinct routes.

In order to compute the savings associated with an edge, one has to consider both the travel time required to traverse that edge as well as the aggregated reward generated by visiting both customers. The savings associated with an edge ij, s'_{ij} are defined as

$$s'_{ij} = \alpha \cdot s_{ij} + (1-\alpha) \cdot (u_i + u_j) \tag{7}$$

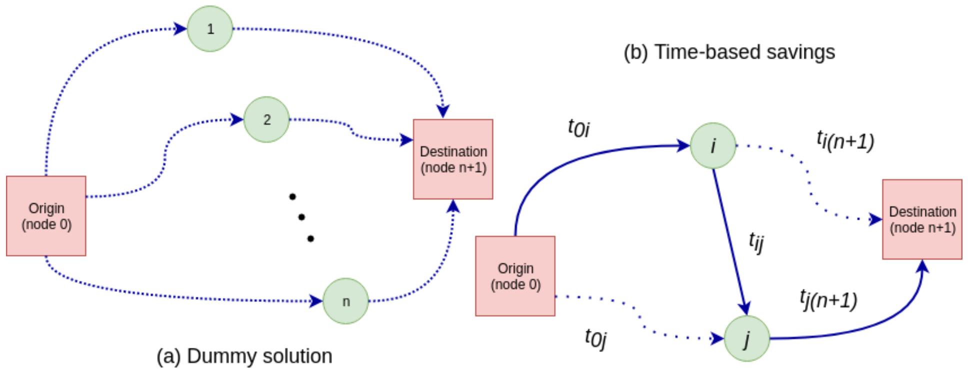

Figure 2 Dummy solution *(left)* and time-based savings *(right)*.

to account for the trade-off between time-based savings $s_{ij} = t_{i(n+1)} + t_{0j} - t_{ij}$ (Figure 2b), and the aggregated reward, $u_i + u_j$, where $\alpha \in (0,1)$ is a tuning parameter, which is dependent on the heterogeneity of customers in terms of rewards. The specific value of α needs to be empirically tuned, since it will depend on the heterogeneity of the customers in terms of rewards. Thus, in a scenario with high heterogeneity, α will be close to zero. On the contrary, α will be close to one for homogeneous scenarios. Notice that for each edge there are two associated savings, depending on the actual direction in which the edge is traversed. Thus, each edge generates two different arcs. After computing all savings, the list of arcs can be sorted from higher to lower savings. Then, a route-merging process, based on this sorted savings list, is started. In each iteration, the arc at the top of the sorted list is selected (line 4). This arc connects two routes, which are merged into a new route as far as this new route does not violate the driving-range constraint (line 9). Finally, the list of routes are sorted according to the total reward provided (line 15) to select as many routes from this list as possible taking into account the restricted number of vehicles in the fleet.

4.2 A VNS-based Metaheuristic for the TOP

The VNS framework has been chosen to extend the previously described heuristic into a complete metaheuristic. As discussed in Hansen and Mladenović (2014), VNS approaches are relatively easy-to-implement, do not contain a large number of parameters (thus avoiding time-consuming setting processes), and offer an excellent trade-off between simplicity and performance, both in terms of solutions quality as well as in terms of computing times.

Algorithm 2 describes the full simheuristic algorithm for the STOP. We describe the simulation elements of this heuristic in Section 6, focusing on the underlying metaheuristic here. First, a feasible initial solution ($initSol$) is generated using the constructive savings heuristic described in the previous section. A line search of the α value, for calculating the savings scores, is performed. In order to do this, we embed the saving heuristic in a fast iterative multistart process, varying α from 0.1 to 0.9. We select as $initSol$ the solution in which α maximizes the reward, and this α is held fixed throughout the remainder of the VNS algorithm. As shown in Algorithm 2, the *initSol* is copied into a *baseSol* and *bestSol* while the size of the neighborhood, *k*, is set to one. Then, a new solution, *newSol*, is created by shaking the current one. The shaking procedure consists of randomly deleting a percentage *k* of routes from *baseSol*. Next, new routes are generated using the constructive savings heuristic, and merged with the current routes of *baseSol*. During the generation of these new routes, the constructive heuristic is combined with biased randomization techniques, which relax somewhat the greedy behavior of the heuristic (Grasas et al., 2017). In particular, biased-randomised techniques use skewed probability distributions to induce an 'oriented' (non-uniform) random behaviour. This process turns a deterministic heuristic into a randomized algorithm whilst preserving the logic behind the original greedy heuristic. In our case, this biased randomization process is introduced by employing a Geometric probability distribution with a parameter β $(0 < \beta < 1)$ which controls the relative level of greediness present in the randomized behaviour of the algorithm. Following a tuning process, we observed a good performance for $\beta = 0.3$ and this was used in the numerical experiments.

The algorithm sequentially executes three local searches, which are described below, to find the local minimum within the defined neighbourhood structure of *newSol*.The sequence of local searches was established through experimentation during a tuning phase.

In the first local search, a traditional 2-opt local search is performed. After that, a cache memory mechanism that records best-found-so-far routes is used to achieve a faster convergence. This mechanism is implemented using a hash map data structure, which is constantly updated whenever a better route is found by the algorithm.

The second local search removes a subset of nodes from each route. The number of nodes to delete is selected randomly between 5 and 10 percent of the total number of nodes in the route. There are three different mechanisms to choose which nodes should be removed: *(i)* completely random; *(ii)* nodes with the highest rewards and; *(iii)* nodes with the lowest rewards. The specific mechanism used is selected randomly in each iteration of the algorithm.

The last local search is a biased insertion algorithm, similar to those proposed by Tang and Miller-Hooks (2005a) and Dang et al. (2013). It tries to improve the routes obtained in the previous local search. Iteratively, starting with the first node of the route, the next node is selected from the list of non-served nodes and inserted into the route (assuming this does not violate the driving-range constraint). In order to select the node to insert, the algorithm takes into account the ratio between the added duration and the additional reward, as given in Equation 8 (in this equation, it is assumed that a node i is being inserted between nodes j and h in a route).

$$(t_{ji} + t_{ih} - t_{jh})/u_i \tag{8}$$

As the algorithm uses a biased-randomized selection process, instead of selecting the node which minimizes this evaluation function at each step, the list of candidate nodes is sorted according to the previously defined ratio and, as before, the Geometric distribution is used to select the next node. The heuristic finishes when no more improvements are achieved, and the *newSol* is returned. The lines referring to the simulation component are used to deal with the stochastic version of the problem, and they will be explained in more detail in Section 6.

With the objective of reducing the odds of getting trapped in a local minimum, the algorithm can occasionally accept non-improving solutions according to an acceptance criterion. We use a classical version of simulated annealing in the algorithm to serve as the acceptance criterion. This method follows a cooling schedule with a decaying probability that is regulated with a temperature parameter (T), adjusted at each iteration by a factor of λ. The temperature T is updated in each iteration of the algorithm using Equation 9. Thus, *baseSol* can be updated with *newSol* even if *newSol* does not outperform *baseSol*. As the temperature decreases, the probability of updating *baseSol* with an inferior *newSol*

Algorithm 2 VNS-based Simheuristic

```
1: $initSol \leftarrow genInitSol(Inputs)$    ▷ Initial solution stage (Savings-based heuristic)
2: $baseSol \leftarrow initSol$
3: **fastSimulation(baseSol)**    ▷ Monte Carlo simulation
4: $bestSol \leftarrow baseSol$
5: $T \leftarrow 1000$; $\lambda \leftarrow 0.999$
6: **while** $(time \leq maxTime)$ **do**    ▷ VNS stage
7:     $k \leftarrow 1$
8:     **while** $(k \leq K_{max})$ **do**
9:         $newSol \leftarrow shaking(baseSol, k)$    ▷ biased-randomized heuristic
10:        $newSol \leftarrow localSearch1(newSol)$
11:        $newSol \leftarrow localSearch2(newSol)$
12:        $newSol \leftarrow localSearch3(newSol)$
13:        **if** $((detProfit(newSol) - detProfit(baseSol)) > 0)$ **then**
14:            **fastSimulation(newSol)**    ▷ Monte Carlo simulation
15:            **if** $((stochProfit(newSol) - stochProfit(baseSol)) > 0)$ **then**
16:                $baseSol \leftarrow newSol$
17:                **if** $((stochProfit(newSol) - stochProfit(bestSol)) > 0)$ **then**
18:                    $bestSol \leftarrow newSol$
19:                    insert(poolBestSols, bestSol)
20:                **end if**
21:                $k \leftarrow 1$
22:            **end if**
23:        **else**    ▷ SA - based acceptance criterion
24:            $updateProb \leftarrow probOfUpdating(detProfit(newSol), detProfit(baseSol), T)$
25:            **if** $(updateProb \geq Rand(0, 1))$ **then**
26:                $baseSol \leftarrow newSol$
27:                $k \leftarrow 1$
28:            **else**
29:                $k \leftarrow k + 1$
30:            **end if**
31:        **end if**
32:        $T \leftarrow T * \lambda$
33:    **end while**
34: **end while**
35: **for** (sol $\in$ poolBestSols) **do**    ▷ Refinement stage - Monte Carlo simulation
36:     **deepSimulation**(sol)
37:     **if** (stochProfit(sol) $>$ stochProfit(bestSol)) **then**
38:         bestSol $\leftarrow$ sol
39:     **end if**
40: **end for**
41: **return** bestSol
```

Table 1 Total number of nodes in each Benchmark Chao set.

Set	#Nodes
1	32
2	21
3	33
4	100
5	66
6	64
7	102

decreases. Following a tuning process we use an initial temperature of $T_0 = 1000$ and set $\lambda = 0.999$.

$$updateProb = e^{((detProfit(newSol) - detProfit(baseSol))/T)} \tag{9}$$

5 Testing the Efficiency of our VNS Metaheuristic

Our VNS-based algorithm has been implemented using Java SE 8.0, and tested on a workstation with an Intel Xeon E5-2650 v4 with 32GB RAM. We use the benchmark instances proposed in Chao et al. (1996a) as a test set in our computational experiments. These instances have been widely used in previous work to test the performance of algorithms aimed at solving the deterministic TOP. This benchmark set is composed of a total of 320 instances, which are divided into seven different sets. Table 1 shows the total number of nodes in each of the sets, including the origin and destination nodes. The node locations and rewards are identical for all of the instances within a set but the driving range and the number of vehicles vary between the instances. Each instance in a set is characterized by following the nomenclature *px.y.z*, where *x* denotes the set; *y* is the number of vehicles (drones), which varies between 2 and 4 dependent on the instance; and z indicates the maximum driving-range.

As is common in the TOP literature, all instances have been executed 5 times using different initial seeds for the solution algorithm, and the best solution found in these runs has been reported as our best solution (OBS). This is compared against the best known solution (BKS) from the literature. In particular, we measure the performance of our algorithm using the current BKS reported by Ke et al. (2016). The BKS were obtained using a PC with core i5, 3.2 GHz, and 4 GB RAM.

Figure 3 shows the average percentage gap between OBS and BKS. Notice that we reach the BKS for all of the instances in classes 1, 2, and 3, obtaining a 0.0% gap. Regarding the remaining classes, we obtain average gaps of less than 0.5% in very short computing times, which proves that our VNS-based metaheuristic is highly competitive for the deterministic version of the TOP.

The average computation times of our algorithm for each set of instances are given by the black bars in Figure 4. Notice that for classes 1, 2, and 3, less than 20 seconds are required. In the case of classes 5 and 6, average times are less than 40 seconds. Finally, for classes 4 and 7 average times are less than 225 and 160 seconds, respectively. Each circle in this figure shows the maximum computing time required to reach the OBS value for

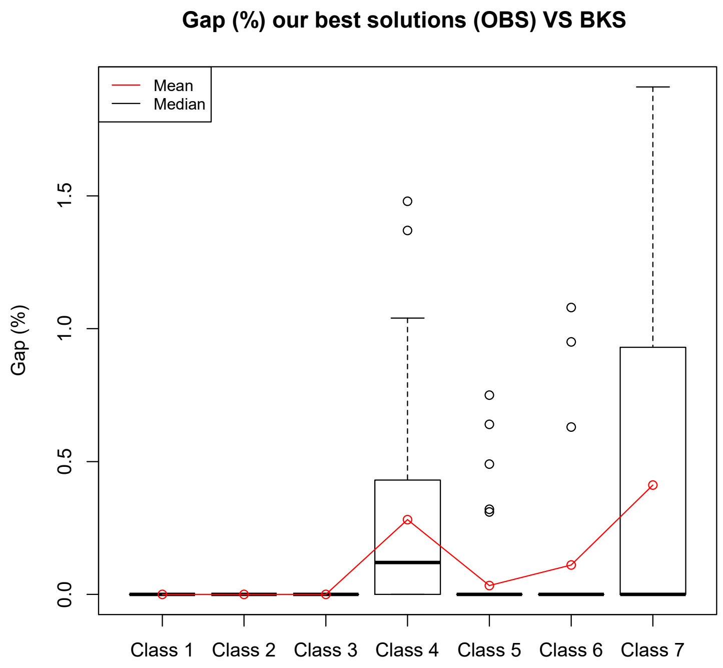

Figure 3 Percentage gap between OBS and BKS.

Table 2 Percentage gap between OBS and other competitive approaches. (Positive GAPS)

Set	Ke et al. (2016) (BKS)	Dang et al. (2013)	Dang et al. (2011)
set1	0.00	0.00	0.00
set2	0.00	0.00	0.00
set3	0.00	0.00	0.00
set4	0.30	0.20	0.00
set5	0.00	0.00	0.00
set6	0.10	0.10	0.10
set7	0.40	0.40	0.30
Average:	0.11	0.10	0.06

any instance in the set. Although classes 4 and 7 require higher CPU times than the other classes to reach the OBS value, we obtain an average time of about 65 seconds. These times are competitive in comparison with the average times required to obtain the BKS, which is about 58 seconds.

Tables 5 to 11 provide detailed information regarding computing times and OBS values for each deterministic instance (OBS-D), as well as the gap to the BKS. Moreover, we have compared our approach with two other approaches (Dang et al. (2013), and Dang et al. (2011)). Table 2 provides the percentage average gap for each set of instances.

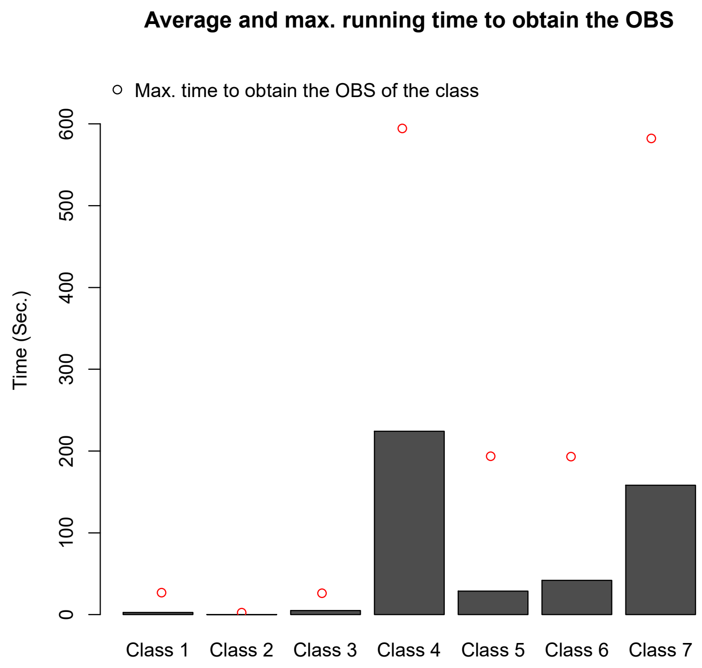

Figure 4 Average running time to obtain the OBS for each class.

6 Extending the VNS Metaheuristic into a Simheuristic

Algorithm 2 provides an overview of our multi-stage simheuristic approach, which extends the VNS metaheuristic to the STOP. In the first stage, a feasible initial solution is constructed using the savings-based heuristic described in Algorithm 1. During the second stage, the adaptive VNS metaheuristic enhances the initial feasible solution by iteratively exploring the search space and conducting a 'reduced' number of simulation runs that allow us to: *(i)* obtain observations on the total length of the tour defined by the current solution (from which the expected time and other statistics can be estimated); and *(ii)* provide feedback that can be used by the metaheuristic to better guide the search (e.g., by updating the base solution according to the estimated statistics). From this stage, a reduced set of 'elite' solutions is obtained. By limiting the size of this 'elite' solutions, we ensure we only keep track of the 'elite' solutions as the algorithm evolves.

Notice that during the second stage, whenever a *newSol* is 'promising', it is subject to a fast Monte Carlo Simulation (MCS) process to test how it deals with the stochastic nature of the proposed problem. This fast simulation process consists of a small number of simulation runs accounting for travel time variability between nodes, to estimate the following values: *(i)* the expected return; and *(ii)* its reliability, measured in terms of the percentage of routes that are completed without violating the driving-range constraint. Also, whenever the stochastic value of the *newSol* outperforms that of the *baseSol* and / or that of some elite solution, these solutions are updated to *newSol*.

Once the second stage of the algorithm is finished, a deep MCS process is launched to better assess the elite solutions pool before reporting the final results. This process carries out a large number of MCS runs, to refine the estimated values of the 'elite' solutions obtained in the previous step. Since the number of generated solutions during the search can be large and the simulation process is time-consuming, we limit the number of MCS iterations to be executed. For our approach, the number of iterations for the fast and deep MCS stages were set to 1,000 and 50,000, respectively.

7 Computational Experiments for the Stochastic TOP

To the best of our knowledge, previous work in the literature has only considered the deterministic TOP or the stochastic single-tour orienteering problem. Thus, we have compared our approach with the method proposed by Campbell et al. (2011) for solving the stochastic orienteering problem. Within our algorithm we set the number of vehicles in the team, $m = 1$ for a fair comparison. We chose to compare with Campbell's method because it solves the most closely related problem to our work. The authors use the instances proposed in Chao et al. (1996b). These sets contain 66 (set 5), and 64 customers (set 6), respectively, and 26, and 14 different maximum travel times, respectively. They assume that the customers are fully connected and that the travel times on the arcs are gamma distributed. We have adapted our algorithm using the same parameters and conditions than described in Campbell et al. (2011), to compute the instances. Tables 3 and 4 show for each set: *(i)* the maximum travel time (T_{max}); *(ii)* the the best-found stochastic solution obtained in Campbell et al. (2011) approach; *(iii)* the best-found stochastic solution obtained using our approach (OBS_S); and *(iv)* the gap between both approaches. As can been seen, our approach is highly competitive, improving the previous results by an average of 1.59% for set 5, and 1.68% for set 6.

Once we have validated the quality of our approach, we have extended the deterministic instances introduced by Chao et al. (1996a) for the TOP, and described previously in Section 5, to incorporate stochastic travel times.

We assume in the computational experiments that the travel times T_{ij} follow Log-Normal probability distributions. In a real-world application, historical data could be used to determine the most appropriate distribution for each of the T_{ij} but, as discussed in Juan et al. (2011), the Log-Normal distribution is a natural choice for describing non-negative random variables, such as travel times. The Log-Normal distribution has two parameters, namely: the location parameter, μ, and the scale parameter, σ, which relate to the expected value $E[T_{ij}]$ and the variance $Var[T_{ij}]$ as follows:

$$\mu_{ij} = \ln(E[T_{ij}]) - \frac{1}{2}\ln\left(1 + \frac{Var[T_{ij}]}{E[T_{ij}]^2}\right) \tag{10}$$

$$\sigma_{ij} = \left|\sqrt{\ln\left(1 + \frac{Var[T_{ij}]}{E[T_{ij}]^2}\right)}\right| \tag{11}$$

In setting up the stochastic instances, we assume that $E[T_{ij}] = t_{ij}$ $(\forall i, j \in N)$, where t_{ij} the travel time for the corresponding deterministic instance. We set the variability in the travel times with reference to the deterministic travel time such that $Var[T_{ij}] = c \cdot t_{ij}$ and $c \geq 0$. Notice that the deterministic instances correspond to stochastic instances with $c = 0$. In our experiments, we have used the value $c = 0.05$.

The classic deterministic benchmark dataset consists of 7 different classes. Tables 5 to 11 show, for each class: *(i)* the maximum travel time (T_{max}) for each drone, *(ii)* the best-known solution for the deterministic variant of the problem, obtained from the existing

T_{max}	Campbell et al. 2011 [1]	OBS_S [3]	GAP (%)[2-1]
20	294.95	312.50	5.95
25	378.91	393.65	3.89
30	463.40	481.10	3.82
35	548.13	586.40	6.98
40	630.54	647.40	2.67
45	717.67	730.30	1.76
50	799.85	809.70	1.23
55	878.77	891.45	1.44
60	954.15	967.50	1.40
65	1026.11	1030.70	0.45
70	1094.11	1094.40	0.03
75	1156.66	1167.05	0.90
80	1224.79	1228.85	0.33
85	1285.42	1284.59	-0.06
90	1340.92	1343.00	0.16
95	1396.09	1399.95	0.28
100	1438.59	1447.15	0.60
105	1483.26	1504.90	1.46
110	1522.61	1530.95	0.55
115	1555.89	1565.20	0.60
120	1584.68	1585.85	0.07
130	1627.53	1666.65	2.40
Average:	1063.77	1075.87	1.68

Table 3 Comparison between our approach and Campbell et al. (2011) approach for dataset 5 of Chao's instances.

T_{max}	Campbell et al 2011 [1]	OBS_S [2]	GAP(%)[2-1]
15	264.46	273.53	3.43
20	363.93	388.80	6.83
25	452.88	471.60	4.13
30	561.32	564.30	0.53
35	666.27	680.06	2.07
40	766.36	775.50	1.19
45	853.50	853.79	0.03
55	1011.81	1014.88	0.30
60	1073.23	1079.14	0.55
65	1135.31	1138.72	0.30
70	1179.92	1190.56	0.90
75	1230.45	1230.36	-0.01
80	1264.52	1269.02	0.36
Average:	832.61	840.79	1.59

Table 4 Comparison between our approach and Campbell et al. (2011) approach for dataset 6 of Chao's instances.

Table 5 Results for class 1 benchmark instances.

		Deterministic execution				Stochastic execution				
		BKS	OBS_D	Time (s)	GAP(%)	OBS_D_S	Reliability	OBS_S	Reliability	Time (s)
Instance	Tmax	Reward [1]	Reward [2]	[3]	[4] (1-2)	E[Reward] [5]	[6]	E[Reward] [7]	[8]	[9]
p1.2.b	5.0	15	15	0	0.0	14.3	0.86	14.3	0.86	4
p1.2.c	7.5	20	20	0	0.0	18.2	0.75	18.2	0.75	4
p1.2.d	10.0	30	30	0	0.0	26.0	0.74	26.3	0.75	4
p1.2.e	12.5	45	45	0	0.0	41.7	0.87	42.2	0.89	4
p1.2.f	15.0	80	80	0	0.0	64.6	0.60	70.5	0.90	5
p1.2.g	17.5	90	90	0	0.0	75.8	0.62	83.7	0.97	4
p1.2.h	20.0	110	110	0	0.0	97.5	0.76	103.6	0.89	17
p1.2.i	23.0	135	135	0	0.0	94.5	0.48	122.9	0.97	4
p1.2.j	25.0	155	155	0	0.0	123.1	0.64	141.0	0.88	5
p1.2.k	27.5	175	175	0	0.0	124.2	0.49	163.3	0.98	7
p1.2.l	30.0	195	195	5	0.0	125.8	0.39	179.5	0.99	9
p1.2.m	32.5	215	215	9	0.0	170.2	0.64	201.4	0.97	16
p1.2.n	35.0	235	235	0	0.0	152.9	0.43	218.8	0.99	5
p1.2.o	36.5	240	240	4	0.0	204.4	0.74	230.7	0.97	7
p1.2.p	37.5	250	250	0	0.0	167.3	0.45	230.9	0.96	5
p1.2.q	40.0	265	265	1	0.0	205.4	0.55	247.8	0.94	7
p1.2.r	42.5	280	280	21	0.0	168.9	0.37	259.9	1.00	4
p1.3.c	5.0	15	15	0	0.0	14.3	0.86	14.3	0.86	6
p1.3.d	6.7	15	15	0	0.0	15.0	1.00	15.0	1.00	6
p1.3.e	8.3	30	30	0	0.0	24.9	0.62	27.0	0.70	6
p1.3.f	10.0	40	40	0	0.0	31.8	0.43	32.6	0.47	23
p1.3.g	11.7	50	50	0	0.0	44.2	0.84	48.4	0.86	6
p1.3.h	13.3	70	70	1	0.0	66.3	0.83	66.3	0.83	11
p1.3.i	15.3	105	105	0	0.0	87.2	0.47	94.1	0.83	6
p1.3.j	16.7	115	115	0	0.0	90.7	0.45	102.2	0.74	6
p1.3.k	18.3	135	135	0	0.0	109.2	0.52	121.4	0.90	17
p1.3.l	20.0	155	155	0	0.0	124.0	0.48	138.8	0.89	13
p1.3.m	21.7	175	175	16	0.0	127.7	0.39	159.5	0.91	9
p1.3.n	23.3	190	190	1	0.0	152.2	0.49	175.1	0.93	15
p1.3.o	24.3	205	205	0	0.0	163.5	0.51	183.0	0.90	11
p1.3.p	25.0	220	220	1	0.0	177.5	0.53	199.0	0.78	32
p1.3.q	26.7	230	230	1	0.0	190.0	0.49	218.0	0.97	6
p1.3.r	28.3	250	250	17	0.0	207.2	0.54	233.5	0.87	27
p1.4.d	5.0	15	15	0	0.0	14.3	0.86	14.3	0.85	8
p1.4.e	6.2	15	15	0	0.0	15.0	1.00	15.0	1.00	8
p1.4.f	7.5	25	25	0	0.0	23.2	0.74	23.2	0.74	8
p1.4.g	8.8	35	35	0	0.0	30.1	0.72	34.3	0.88	8
p1.4.h	10.0	45	45	0	0.0	37.9	0.44	40.8	0.60	9
p1.4.i	11.5	60	60	0	0.0	48.5	0.53	55.0	0.70	8
p1.4.j	12.5	75	75	8	0.0	63.1	0.41	67.2	0.61	8
p1.4.k	13.8	100	100	27	0.0	86.3	0.55	86.3	0.74	8
p1.4.l	15.0	120	120	0	0.0	107.1	0.64	116.0	0.86	9
p1.4.m	16.2	130	130	0	0.0	112.0	0.71	124.9	0.81	10
p1.4.n	17.5	155	155	0	0.0	121.4	0.30	135.7	0.88	8
p1.4.o	18.2	165	165	0	0.0	129.6	0.38	149.0	0.84	28
p1.4.p	18.8	175	175	6	0.0	152.2	0.58	159.1	0.87	16
p1.4.q	20.0	190	190	6	0.0	167.1	0.59	171.9	0.84	8
p1.4.r	21.2	210	210	0	0.0	170.8	0.44	191.1	0.83	9
	Average:	126	126	3	0.0	99.6	0.60	116.0	0.86	10

TOP literature; *(iii)* our best solution for the deterministic variant of the problem (OBS-D); *(iv)* the expected reward of OBS-D when it is applied as a solution of the stochastic variant of the problem (OBS-D-S); *(v)* the reliability of the OBS-D-S (i.e., the percentage of routes that are effectively completed without violating the driving-range constraint in a stochastic environment); *(vi)* the expected reward for our best solution for the stochastic variant of the problem (OBS-S); and *(vii)* the reliability associated with OBS-S. To compute the OBD-D we have executed just the VNS stage of the metaheuristic, disabling the sim parts (fastSimulation process). Notice that the refinement stage, where a deep simulation process is carried out, is not required for a deterministic scenario.

Figure 5 shows, for each class, the percentage gaps between: *(i)* the best-found deterministic solution when applied into stochastic conditions (OBS-D-S) and itself when applied in a deterministic environment (OBS-D); and *(ii)* the best-found stochastic solution when applied into stochastic conditions (OBS-S) and the best-found solution for the deterministic version (OBS-D). Notice that, for each class, the OBS-S boxplot is always

Table 6 Results for class 2 benchmark instances.

		Deterministic execution				Stochastic execution				
		BKS	OBS_D	Time (s)	GAP(%)	OBS_D_S	Reliability	OBS_S	Reliability	Time (s)
Instance	Tmax	Reward [1]	Reward [2]	[3]	[4] (1-2)	E[Reward] [5]	[6]	E[Reward] [7]	[8]	[9]
p2.2.a	7.5	90	90	0	0.0	78.0	0.76	81.0	0.80	4
p2.2.b	10.0	120	120	0	0.0	108.5	0.81	116.5	0.93	4
p2.2.c	11.5	140	140	0	0.0	112.0	0.65	122.0	0.76	5
p2.2.d	12.5	160	160	0	0.0	126.4	0.59	134.6	0.69	6
p2.2.e	13.5	190	190	0	0.0	133.6	0.49	165.8	0.85	10
p2.2.f	15.0	200	200	0	0.0	193.3	0.97	197.7	0.98	7
p2.2.g	16.0	200	200	0	0.0	200.0	1.00	200.0	1.00	10
p2.2.h	17.5	230	230	0	0.0	191.8	0.65	219.3	0.89	10
p2.2.i	19.0	230	230	0	0.0	224.8	0.96	229.6	1.00	6
p2.2.j	20.0	260	260	0	0.0	217.3	0.70	220.3	0.69	4
p2.2.k	22.5	275	275	0	0.0	207.4	0.54	260.0	1.00	4
p2.3.a	5.0	70	70	0	0.0	54.0	0.54	59.4	0.52	10
p2.3.b	6.7	70	70	0	0.0	64.8	0.89	67.0	0.93	10
p2.3.c	7.7	105	105	0	0.0	78.9	0.48	97.9	0.74	18
p2.3.d	8.3	105	105	0	0.0	74.6	0.49	104.5	0.98	11
p2.3.e	9.0	120	120	0	0.0	110.6	0.80	117.1	0.90	13
p2.3.f	10.0	120	120	0	0.0	109.2	0.79	116.2	0.92	10
p2.3.g	10.7	145	145	0	0.0	103.7	0.28	137.8	0.91	9
p2.3.h	11.7	165	165	0	0.0	131.8	0.48	155.7	0.92	31
p2.3.i	12.7	200	200	0	0.0	148.9	0.35	171.7	0.87	7
p2.3.j	13.3	200	200	1	0.0	125.0	0.32	196.3	0.93	14
p2.3.k	15.0	200	200	3	0.0	187.9	0.85	198.6	0.99	15
p2.4.a	3.8	10	10	0	0.0	10.0	1.00	10.0	1.00	9
p2.4.b	5.0	70	70	0	0.0	70.0	1.00	70.0	1.00	9
p2.4.c	5.8	70	70	0	0.0	68.7	0.96	68.8	0.97	11
p2.4.d	6.2	70	70	0	0.0	58.4	0.74	69.9	0.99	10
p2.4.e	6.8	70	70	0	0.0	61.5	0.92	67.6	0.95	14
p2.4.f	7.5	105	105	0	0.0	72.0	0.29	78.3	0.45	13
p2.4.g	8.0	105	105	0	0.0	92.6	0.75	102.7	0.91	27
p2.4.h	8.8	120	120	0	0.0	103.0	0.64	112.0	0.93	8
p2.4.i	9.5	120	120	0	0.0	110.0	0.87	112.0	0.93	9
p2.4.j	10.0	120	120	0	0.0	110.0	0.87	112.0	0.93	8
p2.4.k	11.2	180	180	0	0.0	138.7	0.37	157.0	0.91	19
	Average:	140	140	0	0.0	117.5	0.69	131.2	0.88	11

closer to the OBS-D value than the OBS-D-S boxplot. In other words, employing the best-found deterministic plan into a stochastic environment usually leads to suboptimal solutions. Notice also that the OBS-D value can be seen as a reference reward value in a scenario with perfect information (i.e., without uncertainty) for the expected reward under stochastic conditions.

Table 7 Results for class 3 benchmark instances.

		Deterministic execution				Stochastic execution				
Instance	Tmax	BKS Reward [1]	OBS_D Reward [2]	Time (s) [3]	GAP(%) [4] (1-2)	OBS_D_S E[Reward] [5]	Reliability [6]	OBS_S E[Reward] [7]	Reliability [8]	Time (s) [9]
p3.2.a	7.5	90	90	0	0.0	68.8	0.59	81.3	0.85	5
p3.2.b	10.0	150	150	0	0.0	143.7	0.91	143.8	0.91	5
p3.2.c	12.5	180	180	0	0.0	171.5	0.90	179.5	1.00	4
p3.2.d	15.0	220	220	0	0.0	181.7	0.75	207.1	0.96	4
p3.2.e	17.5	260	260	26	0.0	187.3	0.49	251.9	0.98	18
p3.2.f	20.0	300	300	0	0.0	186.6	0.38	285.9	0.97	4
p3.2.g	22.5	360	360	10	0.0	222.2	0.39	327.4	0.98	6
p3.2.h	25.0	410	410	2	0.0	233.2	0.34	375.9	0.92	19
p3.2.i	27.5	460	460	13	0.0	360.7	0.59	412.8	0.91	4
p3.2.j	30.0	510	510	2	0.0	351.2	0.47	466.3	0.92	21
p3.2.k	32.5	550	550	0	0.0	473.6	0.75	511.4	0.88	5
p3.2.l	35.0	590	590	0	0.0	393.4	0.48	544.4	0.95	19
p3.2.m	37.5	620	620	17	0.0	546.7	0.68	599.9	0.97	6
p3.2.n	40.0	660	660	10	0.0	541.6	0.54	614.1	0.93	18
p3.2.o	42.5	690	690	11	0.0	556.6	0.57	655.8	0.96	15
p3.2.p	45.0	720	720	1	0.0	459.9	0.43	691.9	0.98	8
p3.2.q	47.5	760	760	4	0.0	500.2	0.41	728.5	1.00	5
p3.2.r	50.0	790	790	2	0.0	492.9	0.39	741.1	0.97	4
p3.2.s	52.5	800	800	3	0.0	782.5	0.94	791.9	0.98	6
p3.2.t	55.0	800	800	0	0.0	730.5	0.86	799.5	1.00	4
p3.3.a	5.0	30	30	0	0.0	22.6	0.75	29.0	0.95	6
p3.3.b	6.7	90	90	0	0.0	89.9	0.99	89.9	0.98	6
p3.3.c	8.3	120	120	0	0.0	101.1	0.62	111.4	0.72	6
p3.3.d	10.0	170	170	0	0.0	162.7	0.90	163.9	0.90	7
p3.3.e	11.7	200	200	0	0.0	190.1	0.68	197.6	0.96	6
p3.3.f	13.3	230	230	0	0.0	177.4	0.38	208.4	0.71	25
p3.3.g	15.0	270	270	0	0.0	234.8	0.65	255.3	0.93	19
p3.3.h	16.7	300	300	1	0.0	288.2	0.85	295.2	0.94	7
p3.3.i	18.3	330	330	0	0.0	311.9	0.77	319.4	0.99	6
p3.3.j	20.0	380	380	0	0.0	261.9	0.28	356.7	0.96	7
p3.3.k	21.7	440	440	5	0.0	324.8	0.46	414.2	0.90	18
p3.3.l	23.3	480	480	11	0.0	431.8	0.75	456.8	0.92	9
p3.3.m	25.0	520	520	17	0.0	400.8	0.46	488.4	0.99	10
p3.3.n	26.7	570	570	0	0.0	514.1	0.70	514.6	0.70	25
p3.3.o	28.3	590	590	11	0.0	477.4	0.50	568.0	0.93	6
p3.3.p	35.0	640	640	4	0.0	543.8	0.58	586.7	0.94	15
p3.3.q	31.7	680	680	22	0.0	467.9	0.37	648.9	0.99	13
p3.3.r	33.3	710	710	5	0.0	580.5	0.51	665.1	0.79	7
p3.3.s	35.0	720	720	3	0.0	606.0	0.62	693.2	0.95	9
p3.3.t	36.7	760	760	23	0.0	549.4	0.39	699.3	0.89	19
p3.4.a	3.8	20	20	0	0.0	15.7	0.79	15.9	0.79	8
p3.4.b	5.0	30	30	0	0.0	22.2	0.74	28.6	0.93	8
p3.4.c	6.2	90	90	0	0.0	86.3	0.77	86.3	0.74	8
p3.4.d	7.5	100	100	0	0.0	79.7	0.59	97.6	0.76	14
p3.4.e	8.5	140	140	0	0.0	129.0	0.59	132.6	0.64	8
p3.4.f	10.0	190	190	0	0.0	181.6	0.88	183.5	0.88	24
p3.4.g	11.2	220	220	0	0.0	193.3	0.63	204.7	0.76	32
p3.4.h	12.5	240	240	0	0.0	228.6	0.63	228.8	0.65	8
p3.4.i	13.8	270	270	0	0.0	211.7	0.38	247.3	0.96	36
p3.4.j	15.0	310	310	0	0.0	262.6	0.47	291.5	0.87	8
p3.4.k	16.2	350	350	0	0.0	292.1	0.34	300.3	0.39	8
p3.4.l	17.5	380	380	0	0.0	344.3	0.56	361.0	0.87	8
p3.4.m	18.8	390	390	21	0.0	337.9	0.58	381.4	0.89	9
p3.4.n	20.0	440	440	10	0.0	365.9	0.38	411.6	0.88	8
p3.4.o	21.2	500	500	22	0.0	419.5	0.50	465.5	0.87	22
p3.4.p	22.5	560	560	1	0.0	456.8	0.48	511.9	0.92	34
p3.4.q	23.8	560	560	18	0.0	522.5	0.71	555.3	0.96	14
p3.4.r	25.0	600	600	18	0.0	446.7	0.36	562.8	0.89	10
p3.4.s	26.2	670	670	0	0.0	501.7	0.23	586.3	0.75	22
p3.4.t	27.5	670	670	0	0.0	593.9	0.59	664.9	0.97	10
	Average:	415	415	5	0.0	333.6	0.59	391.5	0.89	12

Table 8 Results for class 4 benchmark instances.

		Deterministic execution				Stochastic execution				
		BKS	OBS_D	Time (s)	GAP(%)	OBS_D_S	Reliability	OBS_S	Reliability	Time (s)
Instance	Tmax	Reward [1]	Reward [2]	[3]	[4] (1-2)	E[Reward] [5]	[6]	E[Reward] [7]	[8]	[9]
p4.2.a	25.0	206	206	2	0.0	133.8	0.42	164.4	0.81	11
p4.2.b	30.0	341	341	1	0.0	232.2	0.46	298.3	0.95	19
p4.2.c	35.0	452	452	39	0.0	251.9	0.31	377.5	0.82	25
p4.2.d	40.0	531	531	50	0.0	325.3	0.38	476.7	0.90	60
p4.2.e	45.0	618	618	33	0.0	391.5	0.39	539.6	0.83	88
p4.2.f	50.0	687	687	278	0.0	379.9	0.30	601.5	0.95	264
p4.2.g	55.0	757	756	470	0.1	47.8	0.41	653.0	0.94	431
p4.2.h	60.0	835	827	169	1.0	441.5	0.29	726.0	0.96	220
p4.2.i	65.0	918	918	522	0.0	500.2	0.30	744.2	0.94	427
p4.2.j	70.0	965	962	185	0.3	632.6	0.43	790.6	0.97	199
p4.2.k	75.0	1022	1020	240	0.2	560.4	0.31	849.1	0.92	254
p4.2.l	80.0	1074	1068	247	0.6	590.5	0.31	894.4	0.94	453
p4.2.m	85.0	1132	1124	192	0.7	572.2	0.29	927.0	0.82	120
p4.2.n	90.0	1174	1166	181	0.7	587.6	0.25	1027.2	0.95	187
p4.2.o	95.0	1218	1211	479	0.6	700.9	0.34	1098.5	0.93	327
p4.2.p	100.0	1242	1225	475	1.4	739.1	0.36	1126.0	0.91	394
p4.2.q	105.0	1268	1262	137	0.5	699.1	0.31	1135.1	0.94	294
p4.2.r	110.0	1292	1281	504	0.9	705.4	0.30	1098.8	0.90	391
p4.2.s	115.0	1304	1302	444	0.2	853.2	0.42	1119.1	0.91	330
p4.2.t	120.0	1306	1306	100	0.0	914.8	0.49	1096.8	0.86	241
p4.3.b	20.0	38	38	0	0.0	24.5	0.38	21.3	0.34	7
p4.3.c	23.3	193	193	0	0.0	144.5	0.35	177.0	0.82	129
p4.3.d	26.7	335	335	1	0.0	188.0	0.17	298.5	0.95	179
p4.3.e	30.0	468	468	11	0.0	301.0	0.25	416.7	0.94	261
p4.3.f	33.3	579	579	7	0.0	326.2	0.18	491.9	0.81	40
p4.3.g	36.7	653	651	134	0.3	441.3	0.31	544.8	0.86	313
p4.3.h	40.0	729	723	522	0.8	422.8	0.19	643.1	0.95	193
p4.3.i	43.3	809	797	407	1.5	512.8	0.26	689.1	0.90	182
p4.3.j	46.7	861	853	333	0.9	478.9	0.18	773.0	0.96	327
p4.3.k	50.0	919	918	27	0.1	572.3	0.22	846.0	0.98	280
p4.3.l	53.3	979	976	296	0.3	538.6	0.17	898.9	0.96	141
p4.3.m	56.7	1063	1063	292	0.0	625.7	0.20	925.7	0.99	320
p4.3.n	60.0	1121	1118	345	0.3	669.4	0.22	969.6	0.81	102
p4.3.o	63.3	1172	1165	491	0.6	654.7	0.18	1027.7	0.86	496
p4.3.p	63.7	1222	1220	320	0.2	690.3	0.18	1040.3	0.86	212
p4.3.q	70.0	1253	1240	121	1.0	897.8	0.36	1126.0	0.88	141
p4.3.r	73.3	1273	1270	324	0.2	782.4	0.23	1199.7	0.92	496
p4.3.s	76.7	1295	1290	450	0.4	842.2	0.26	1173.9	0.92	123
p4.3.t	80.0	1305	1304	595	0.1	1046.7	0.52	1254.3	0.94	388
p4.4.d	20.0	38	38	0	0.0	21.3	0.34	26.0	0.64	10
p4.4.e	22.5	183	183	4	0.0	141.2	0.22	160.7	0.76	345
p4.4.f	25.0	324	324	348	0.0	213.6	0.17	287.8	0.90	278
p4.4.g	27.5	461	461	5	0.0	269.0	0.12	372.7	0.65	116
p4.4.h	30.0	571	571.0	5	0.0	360.5	0.16	499.5	0.89	577
p4.4.i	32.5	657	657	3	0.0	372.1	0.10	558.4	0.92	551
p4.4.j	35.0	732	732	177	0.0	485.0	0.20	632.9	0.81	164
p4.4.k	37.5	821	821	117	0.0	447.6	0.24	680.0	0.89	134
p4.4.l	40.0	880	879	43	0.1	492.4	0.09	782.7	0.77	490
p4.4.m	42.5	919	916	590	0.3	572.0	0.15	786.0	0.72	459
p4.4.n	45.0	977	970	7	0.7	611.2	0.15	870.9	0.78	288
p4.4.o	47.5	1061	1061	149	0.0	565.2	0.08	893.8	0.83	192
p4.4.p	50.0	1124	1124	543	0.0	666.1	0.12	955.3	0.78	327
p4.4.q	47.5	1161	1161	160	0.0	676.5	0.12	1006.5	0.87	338
p4.4.r	55.0	1216	1213	464	0.2	727.0	0.12	1018.7	0.83	333
p4.4.s	57.5	1260	1256	456	0.3	720.2	0.11	1087.8	0.86	316
p4.4.t	60.0	1285	1281	65	0.3	708.7	0.10	1178.6	0.84	503
	Average:	862	859	224	0.3	508.4	0.26	751.1	0.87	259

Table 9 Results for class 5 benchmark instances.

		Deterministic execution				Stochastic execution				
		BKS	OBS_D	Time (s)	GAP(%)	OBS_D_S	Reliability	OBS_S	Reliability	Time (s)
Instance	Tmax	Reward [1]	Reward [2]	[3]	[4] (1-2)	E[Reward] [5]	[6]	E[Reward] [7]	[8]	[9]
p5.2.b	5.0	20	20	0	0.0	16.4	0.86	19.4	0.94	5
p5.2.c	7.5	50	50	0	0.0	41.4	0.69	45.3	0.85	20
p5.2.d	10.0	80	80	0	0.0	76.0	0.95	80.0	1.00	5
p5.2.e	12.5	180	180	0	0.0	120.9	0.47	142.3	0.72	65
p5.2.f	15.0	240	240	0	0.0	121.1	0.25	180.0	1.00	6
p5.2.g	17.5	320	320	20	0.0	309.3	0.93	310.6	0.94	27
p5.2.h	20.0	410	410	38	0.0	248.5	0.37	352.4	0.82	52
p5.2.i	22.5	480	480	6	0.0	244.3	0.26	434.4	1.00	16
p5.2.j	25.0	580	580	0	0.0	334.1	0.33	467.0	0.87	13
p5.2.k	27.5	670	670	1	0.0	466.3	0.48	631.2	0.94	55
p5.2.l	30.0	800	800	49	0.0	472.4	0.35	723.7	0.96	35
p5.2.m	32.5	860	860	1	0.0	524.2	0.37	746.9	0.93	54
p5.2.n	35.0	925	925	14	0.0	600.1	0.40	819.6	0.92	16
p5.2.o	37.5	1020	1020	14	0.0	532.0	0.27	862.6	0.93	57
p5.2.p	40.0	1150	1150	0	0.0	672.2	0.34	857.3	0.57	6
p5.2.q	42.5	1195	1195	34	0.0	656.0	0.30	1146.8	0.98	39
p5.2.r	45.0	1260	1260	4	0.0	972.1	0.60	1184.1	0.99	9
p5.2.s	47.5	1340	1330	1	0.7	748.9	0.32	1229.5	0.98	21
p5.2.t	50.0	1400	1400	75	0.0	818.2	0.34	1304.8	0.93	50
p5.2.u	52.5	1460	1460	65	0.0	833.7	0.33	1236.3	0.99	76
p5.2.v	55.0	1505	1505	87	0.0	954.7	0.39	1336.0	0.95	50
p5.2.w	57.5	1565	1560	57	0.3	983.5	0.40	1411.0	0.93	43
p5.2.x	60.0	1610	1610	16	0.0	927.4	0.33	1389.8	0.93	65
p5.2.y	60.5	1645	1645	97	0.0	905.5	0.30	1456.2	0.87	16
p5.2.z	65.0	1680	1680	124	0.0	906.4	0.23	1453.5	0.92	157
p5.3.b	3.3	15	15	0	0.0	13.2	0.67	13.8	0.72	97
p5.3.c	5.0	20	20	0	0.0	19.4	0.94	19.4	0.94	7
p5.3.d	6.7	60	60	0	0.0	40.8	0.33	44.3	0.40	20
p5.3.e	8.3	95	95	0	0.0	60.4	0.32	66.8	0.37	8
p5.3.f	10.0	110	110	0	0.0	110.0	1.00	110.0	1.00	7
p5.3.g	11.7	185	185	0	0.0	163.4	0.69	164.6	0.70	18
p5.3.h	13.3	260	260	1	0.0	249.5	0.88	251.3	0.90	25
p5.3.i	15.0	335	335	55	0.0	177.3	0.15	267.2	0.92	11
p5.3.j	16.7	470	470	1	0.0	354.4	0.43	356.3	0.44	15
p5.3.k	18.3	495	495	13	0.0	322.6	0.28	463.6	0.96	53
p5.3.l	20.0	595	595	133	0.0	421.1	0.35	518.7	0.73	50
p5.3.m	21.7	650	650	2	0.0	630.8	0.91	630.8	0.91	10
p5.3.n	23.3	755	755	23	0.0	457.1	0.22	616.8	0.66	17
p5.3.o	25.0	870	870	0	0.0	487.5	0.18	678.5	0.68	48
p5.3.p	26.7	990	990	0	0.0	614.8	0.24	798.6	0.61	30
p5.3.q	28.3	1070	1070	3	0.0	637.8	0.21	972.7	0.99	7
p5.3.r	30.0	1125	1125	110	0.0	629.8	0.18	983.3	0.73	10
p5.3.s	31.7	1190	1190	96	0.0	668.1	0.18	1067.7	0.88	38
p5.3.t	33.3	1260	1260	11	0.0	1094.1	0.65	1089.8	0.87	24
p5.3.u	35.0	1345	1345	130	0.0	805.5	0.21	1148.7	0.86	21
p5.3.v	36.7	1425	1425	167	0.0	854.5	0.21	1207.3	0.85	94
p5.3.w	38.3	1485	1485	48	0.0	826.9	0.17	1253.7	0.99	7
p5.3.x	40.0	1555	1545	146	0.6	1025.6	0.28	1335.2	0.89	31
p5.3.y	41.7	1595	1590	50	0.3	1361.1	0.62	1484.7	0.95	15
p5.3.z	43.3	1635	1635	24	0.0	1011.0	0.23	1420.6	0.84	10
p5.4.b	2.5	15	15	0	0.0	0.0	1.00	0.0	1.00	10
p5.4.c	3.8	20	20	0	0.0	19.9	0.97	19.9	0.97	10
p5.4.d	5.0	20	20	0	0.0	19.3	0.93	19.4	0.94	10
p5.4.e	6.2	20	20	0	0.0	20.0	1.00	20.0	1.00	10
p5.4.f	7.5	80	80	0	0.0	79.3	0.96	79.5	0.98	10
p5.4.g	8.8	140	140	0	0.0	121.7	0.58	122.3	0.58	13
p5.4.h	10.0	140	140	1	0.0	140.0	1.00	140.0	1.00	10
p5.4.i	11.2	240	240	0	0.0	138.5	0.11	178.1	0.58	23
p5.4.j	12.5	340	340	0	0.0	225.3	0.19	272	0.84	195
p5.4.k	13.8	340	340	30	0.0	323.2	0.91	338.6	0.98	16
p5.4.l	15.0	430	430	0	0.0	224.6	0.07	335.4	0.80	36
p5.4.m	16.2	555	555	8	0.0	322.3	0.11	447.6	0.86	27
p5.4.n	17.5	620	620	17	0.0	551.3	0.59	600.0	0.88	28
p5.4.o	18.8	690	690	12	0.0	430.7	0.15	608.7	0.93	20
p5.4.p	20.0	765	765	194	0.0	511.6	0.18	690.2	0.84	85
p5.4.q	21.2	860	860	31	0.0	720.6	0.47	759.1	0.71	87
p5.4.r	22.5	960	960	27	0.0	504.2	0.08	822.6	0.83	85
p5.4.s	23.8	1030	1025	16	0.5	559.3	0.09	893.6	0.87	79
p5.4.t	25.0	1160	1160	0	0.0	650.5	0.10	854.1	0.84	140
p5.4.u	26.2	1300	1300	0	0.0	769.3	0.12	1087.3	0.89	219
p5.4.v	27.5	1320	1320	8	0.0	1184.7	0.65	1274.1	0.91	96
p5.4.w	28.8	1390	1390	19	0.0	886.4	0.15	1299.4	1.00	35
p5.4.x	30.0	1450	1450	40	0.0	894.1	0.14	1300.0	1.00	93
p5.4.y	31.2	1520	1520	5	0.0	831.7	0.09	1402.9	0.86	202
p5.4.z	32.5	1620	1620	25	0.0	962.4	0.12	1481.7	0.85	179
	Average:	787	787	29	0.0	501.5	0.43	691.1	0.86	45

Table 10 Results for class 6 benchmark instances.

		Deterministic execution				Stochastic execution				
		BKS	OBS_D	Time (s)	GAP(%)	OBS_D_S	Reliability	OBS_S	Reliability	Time (s)
Instance	Tmax	Reward [1]	Reward [2]	[3]	[4] (1-2)	E[Reward] [5]	[6]	E[Reward] [7]	[8]	[9]
p6.2.d	15	192	192	1	0.0	114.4	0.36	143.3	0.68	5
p6.2.e	17.5	360	360	0	0.0	217.6	0.37	276.6	0.73	11
p6.2.f	20	588	588	0	0.0	363.4	0.38	445.7	0.84	5
p6.2.g	22.5	660	660	8	0.0	528.0	0.64	531.0	0.78	5
p6.2.h	25	780	780	30	0.0	512.9	0.43	648.0	1.00	9
p6.2.i	27.5	888	888	10	0.0	477.7	0.29	725.8	0.86	97
p6.2.j	30	948	942	64	0.6	599.6	0.40	900.9	0.93	8
p6.2.k	32.5	1032	1032	96	0.0	586.7	0.32	911.3	0.84	12
p6.2.l	37.5	1116	1104	40	1.1	908.5	0.68	1001.2	0.92	18
p6.2.m	37.5	1188	1188	80	0.0	865.5	0.53	1037.3	0.86	13
p6.2.n	40	1260	1248	167	1.0	794.4	0.41	1042.5	0.81	12
p6.3.g	15	282	282	0	0.0	169.3	0.21	196.6	0.45	54
p6.3.h	16.7	444	444	2	0.0	279.8	0.25	331.9	0.51	8
p6.3.i	18.3	642	642	0	0.0	379.0	0.21	448.1	0.40	36
p6.3.j	20	828	828	0	0.0	501.0	0.22	612.4	0.58	15
p6.3.k	21.7	894	894	8	0.0	573.0	0.27	781.8	0.86	43
p6.3.l	23.3	1002	1002	43	0.0	708.2	0.31	829.1	0.91	33
p6.3.m	25	1080	1080	38	0.0	673.3	0.23	858.0	1.00	24
p6.3.n	26.7	1170	1170	193	0.0	925.2	0.43	1028.5	0.82	230
p6.4.j	15	366	366	0	0.0	215.7	0.12	276.3	0.64	83
p6.4.k	16.2	528	528	5	0.0	314.6	0.13	367.4	0.52	207
p6.4.l	17.5	696	696	37	0.0	471.1	0.21	514.4	0.74	259
p6.4.m	18.8	912	912	0	0.0	629.4	0.32	836.3	0.78	45
p6.4.n	20	1068	1068	184	0.0	634.7	0.36	816.2	0.81	272
	Average:	789	787	42	0.1	518.5	0.34	648.3	0.76	63

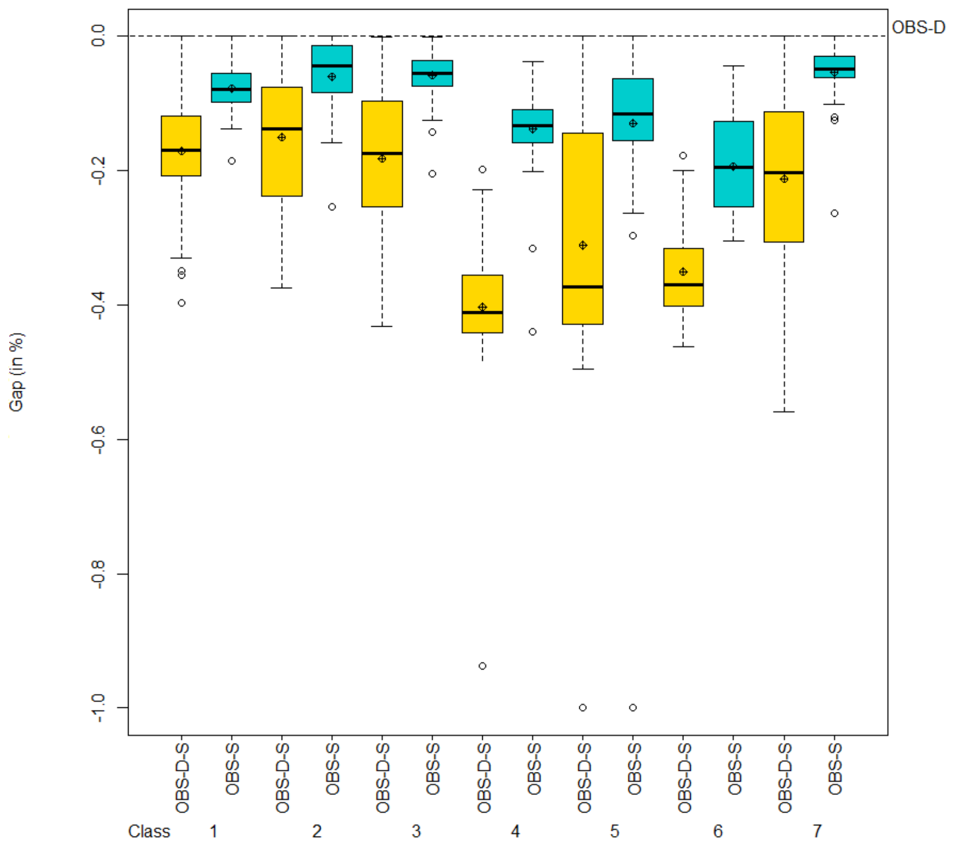

Figure 5 Boxplot comparison of gaps OBS-D-S and OBS-S w.r.t. OBS-D.

Table 11 Results for class 7 benchmark instances.

		Deterministic execution				Stochastic execution				
		BKS	OBS_D	Time (s)	GAP(%)	OBS_D_S	Reliability	OBS_S	Reliability	Time (s)
Instance	Tmax	Reward [1]	Reward [2]	[3]	[4] (1-2)	E[Reward] [5]	[6]	E[Reward] [7]	[8]	[9]
p7.2.a	10.0	30	30	0	0.0	22.0	0.50	22.1	0.50	4
p7.2.b	20.0	64	64	0	0.0	64.0	1.00	64.0	1.00	4
p7.2.c	30.0	101	101	0	0.0	92.0	0.74	98.0	1.00	4
p7.2.d	40.0	190	190	0	0.0	162.9	0.75	173.0	0.93	4
p7.2.e	50.0	290	290	1	0.0	233.8	0.69	267.9	0.95	5
p7.2.f	60.0	387	387	1	0.0	233.8	0.69	366.8	0.96	6
p7.2.g	70.0	459	459	31	0.0	316.4	0.49	440.8	0.99	38
p7.2.h	80.0	521	521	57	0.0	437.8	0.70	514.0	1.00	133
p7.2.i	90.0	580	580	35	0.0	414.5	0.52	550.5	0.99	118
p7.2.j	100.0	646	639	87	1.1	502.4	0.64	574.1	0.88	86
p7.2.k	110.0	705	698	27	1.0	412.5	0.34	654.3	1.00	192
p7.2.l	120.0	767	758	579	1.2	543.7	0.49	721.8	0.98	287
p7.2.m	130.0	827	824	487	0.4	555.6	0.44	782.5	0.98	347
p7.2.n	140.0	888	871	410	1.9	583.3	0.44	826.1	0.98	552
p7.2.o	150.0	945	928	114	1.8	858.0	0.89	879.5	1.00	394
p7.2.p	160.0	1002	989	260	1.3	436.4	0.24	894.2	0.98	561
p7.2.q	170.0	1044	1026	208	1.7	758.3	0.42	1009.4	1.00	514
p7.2.r	180.0	1094	1082	104	1.1	657.3	0.56	1009.2	0.98	197
p7.2.s	190.0	1136	1116	58	1.8	695.3	0.40	976.1	0.85	88
p7.2.t	200.0	1179	1162	216	1.4	563.6	0.56	1023.1	0.96	167
p7.3.b	13.3	46	46	0	0.0	45.7	0.98	46.0	0.98	6
p7.3.c	20.0	79	79	0	0.0	79.0	1.00	79.0	1.00	6
p7.3.d	26.7	117	117	0	0.0	103.8	0.66	106.7	0.82	56
p7.3.e	33.3	175	175	0	0.0	173.6	0.97	174.3	0.99	6
p7.3.f	40.0	247	247	0	0.0	229.7	0.81	231.1	0.94	6
p7.3.g	46.7	344	344	0	0.0	239.4	0.29	310.5	0.98	11
p7.3.h	53.3	425	425	5	0.0	354.7	0.52	401.4	0.88	20
p7.3.i	60.0	487	487	29	0.0	372.7	0.49	464.9	0.97	253
p7.3.j	66.7	564	562	233	0.4	459.5	0.54	541.4	0.92	248
p7.3.k	73.3	633	633	483	0.0	478.4	0.45	614.9	0.99	165
p7.3.l	80.0	684	681	28	0.4	548.6	0.43	666.9	0.99	58
p7.3.m	86.7	762	762	185	0.0	486.8	0.25	724.7	1.00	426
p7.3.n	93.3	820	812	517	1.0	662.3	0.55	774.3	0.97	289
p7.3.o	100.0	874	874	231	0.0	651.4	0.42	824.7	0.97	432
p7.3.p	106.7	929	919	491	1.1	836.1	0.75	884.7	0.99	597
p7.3.q	113.3	987	978	242	0.9	633.8	0.27	918.4	0.98	495
p7.3.r	120.0	1026	1016	521	1.0	701.1	0.31	962.0	0.99	454
p7.3.s	126.7	1081	1063	449	1.7	916.6	0.60	1009.5	1.00	341
p7.3.t	133.3	1120	1111	445	0.8	945.7	0.55	1056.9	0.98	292
p7.4.b	10.0	30	30	0	0.0	22.0	0.50	22.1	0.50	11
p7.4.c	15.0	46	46	0	0.0	46.0	1.00	46.0	1.00	11
p7.4.d	20.0	79	79	1	0.0	79.0	0.78	79.0	1.00	11
p7.4.e	25.0	123	123	5	0.0	119.5	0.92	120.3	0.93	12
p7.4.f	30.0	164	164	60	0.0	153.7	0.77	158.2	0.83	89
p7.4.g	35.0	217	217	60	0.0	192.9	0.60	209.2	0.95	121
p7.4.h	40.0	285	285	60	0.0	255.0	0.57	281.2	0.95	96
p7.4.i	45.0	366	366	60	0.0	309.3	0.50	348.7	0.99	69
p7.4.j	50.0	462	462	8	0.0	368.6	0.43	441.5	0.95	50
p7.4.k	55.0	520	520	22	0.0	349.3	0.21	502.7	0.99	34
p7.4.l	60.0	590	590	2	0.0	409.6	0.22	557.9	0.95	27
p7.4.m	65.0	646	646	75	0.0	501.5	0.32	637.3	0.99	234
p7.4.n	70.0	730	726	289	0.5	534.2	0.33	681.7	0.97	364
p7.4.o	75.0	781	780	208	0.1	468.5	0.14	753.9	0.97	193
p7.4.p	80.0	846	846	54	0.0	721.4	0.49	792.5	0.92	210
p7.4.q	85.0	909	909	437	0.0	773.8	0.56	887.0	0.97	359
p7.4.r	90.0	970	966	512	0.4	783.2	0.44	912.0	0.94	382
p7.4.s	95.0	1022	1022	207	0.0	813.7	0.39	985.1	0.99	283
p7.4.t	100.0	1077	1067	582	0.9	846.5	0.38	980.5	0.92	391
	Average:	588	584	158	0.4	434.7	0.55	552.4	0.95	186

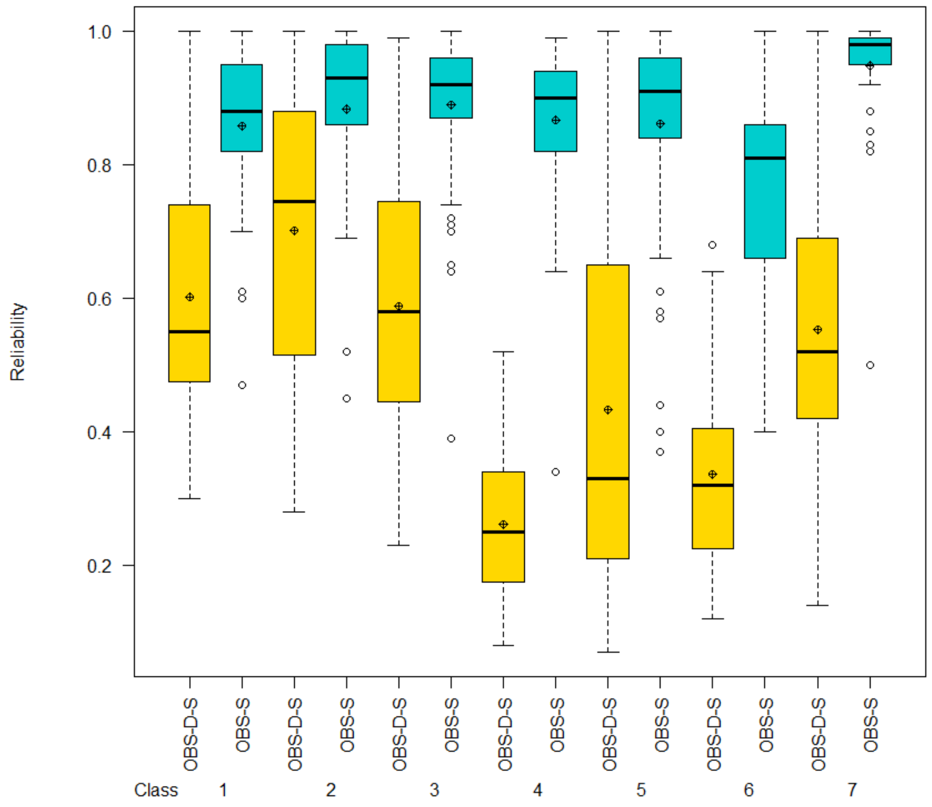

Figure 6 Boxplot comparison of reliabilities associated with OBS-D-S and OBS-S, where the reliability is defined as the percentage of routes that are effectively completed without violating the driving-range constraint.

Finally, Figure 6 shows, a boxplot comparison of reliability values associated with OBS-D-S and OBS-S, for each of the problem instances. Notice that the OBS-S also outperforms the OBS-D-S in terms of reliability, i.e. while near-optimal solutions for the deterministic TOP might be unreliable for the STOP, the proposed simheuristic is able to provide solutions with a superior performance in this key indicator.

8 Conclusions

Developing algorithms to guide the behavior of drones or other autonomous vehicles is becoming increasingly important as their usage becomes more widespread. In this article we consider the particular problem of surveillance, where different nodes or locations within a space provide differing rewards in terms of the quality of information that they provide. This problem can be viewed as a version of the stochastic team orienteering problem (STOP), a variant of the TOP that incorporates stochastic travel times.

To the best of our knowledge, previous work in the literature has only considered the deterministic TOP or the stochastic single-tour orienteering problem. Incorporating uncertainty is particularly important when modelling surveillance drones as it allows the optimization to account for variations in travel time due to weather but it is also important in more conventional vehicle routing problems where traffic conditions can affect journey durations. As discussed previously, different authors have treated violations of the driving range constraint in different ways and we use an especially strict condition here. Investigating how our method works for other interpretations of the constraint could be interesting; for example, the use of a penalty function for exceeding the driving-range.

We propose a novel savings-based heuristic approach to solve the deterministic TOP. This heuristic is then embedded into a variable neighborhood search (VNS) framework to build a competitive metaheuristic. The algorithm uses components inspired by simulated annealing and biased-randomization techniques. A series of computational experiments allow us to validate the quality of the VNS metaheuristic for solving the deterministic TOP. Finally, this VNS is extended into a simheuristic to cope with stochastic travel times. Our algorithm integrates Monte Carlo simulation inside the VNS framework.

The experimental results show that the best solutions for the deterministic TOP are likely to be suboptimal solutions when stochasticity is taken into account, both in terms of total expected rewards and reliability. In contrast, the solutions generated using our simheuristic offer a much better performance in the presence of uncertainty. The level of uncertainty used in the experiments considered here is relatively low, as befits the practical application we are considering. With higher levels of variability, we expect the difference between the simheuristic algorithm and the deterministic algorithm to be even greater.

The proposed methodology does not rely on any structural properties of the particular travel time distributions and will work for any travel time distributions. In this work we consider benchmark test instances with log-normal travel times distributions and also test instances with gamma travel time distributions, in both cases the proposed algorithm finds new best-known solutions.

As future research lines, one can consider the following ones: *(i)* consider a multi-objective version of the STOP; and *(ii)* to use distributed and parallel computing techniques to extend our approach to include a more complex simulation model, for example an agent-based simulation. The ideas we describe here also have the potential to be adapted for use in dynamic routing – updating the set of nodes to visit and/or the order of the nodes within the tour based on the current status of the vehicles. This ability to reroute vehicles during the tour could be an effective way of accounting for the stochasticity in the travel times.

There are also several aspects of the problem that we would like to consider in future work to make the work more practically applicable. The first concerns the affect of weather and particularly wind on the behavior of the drones, and the second the effect of ascent and descent on the rate of battery usage. To a certain extent, these are taken into account through assuming stochastic travel times between nodes, but there are refinements that we could make to the modelling of the problem that would better describe these issues.

References

Archetti, C., Hertz, A., and Speranza, M. G. (2007). Metaheuristics for the team orienteering problem. *Journal of Heuristics*, 13(1):49–76.

Butt, S. and Ryan, D. (1999). An optimal solution procedure for the multiple tour maximum collection problem using column generation. *Computers and Operations Research*, 26:427–441.

Caceres-Cruz, J., Arias, P., Guimarans, D., Riera, D., and Juan, A. A. (2015). Rich vehicle routing problem: Survey. *ACM Computing Surveys (CSUR)*, 47(2):32.

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

Chao, I.-M., Golden, B., and Wasil, E. (1996a). The team orienteering problem. *European Journal of Operational Research*, 88:464–474.

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

Clarke, G. and Wright, J. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:pp. 568–581.

Dang, D.-C., Guibadj, R. N., and Moukrim, A. (2011). A pso-based memetic algorithm for the team orienteering problem. In Di Chio, C., Brabazon, A., Di Caro, G. A., Drechsler, R., Farooq, M., Grahl, J., Greenfield, G., Prins, C., Romero, J., Squillero, G., Tarantino, E., Tettamanzi, A. G. B., Urquhart, N., and Uyar, A. Ş., editors, *Applications of Evolutionary Computation*, pages 471–480, Berlin, Heidelberg. Springer Berlin Heidelberg.

Dang, D.-C., Guibadj, R. N., and Moukrim, A. (2013). An effective pso-inspired algorithm for the team orienteering problem. *European Journal of Operational Research*, 229(2):332–344.

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

Ferreira, J., Quintas, A., and Oliveira, J. (2014). Solving the team orienteering problem: developing a solution tool using a genetic algorithm approach. In *Soft computing in industrial applications. Advances in Intelligent Systems and Computing: 223*, pages 365–375. Springer.

Golden, B., Levy, L., and Vohra, R. (1987). The orienteering problem. *Naval Research Logistics*, 34:307–318.

Gonzalez-Martin, S., Juan, A. A., Riera, D., Elizondo, M. G., and Ramos, J. J. (2018). A simheuristic algorithm for solving the arc routing problem with stochastic demands. *Journal of Simulation*, 12(1):53–66.

Grasas, A., Juan, A. A., Faulin, J., de Armas, J., and Ramalhinho, H. (2017). Biased randomization of heuristics using skewed probability distributions: a survey and some applications. *Computers & Industrial Engineering*, 110:216–âŁ“228.

Gruler, A., Fikar, C., Juan, A. A., Hirsch, P., and Contreras-Bolton, C. (2017). Supporting multi-depot and stochastic waste collection management in clustered urban areas via simulation–optimization. *Journal of simulation*, 11(1):11–19.

Gruler, A., Panadero, J., de Armas, J., Pérez, J. A. M., and Juan, A. A. (2018). A variable neighborhood search simheuristic for the multiperiod inventory routing problem with stochastic demands. *International Transactions in Operational Research*.

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

Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467.

Hansen, P. and Mladenović, N. (2014). *Variable Neighborhood Search*, pages 313–337. Springer US, Boston, MA.

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

Juan, A. A., Faulin, J., Grasman, S. E., Rabe, M., and Figueira, G. (2015). A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, 2(1):62–72.

Juan, A. A., Faulin, J., Grasman, S. E., Riera, D., Marull, J., and Mendez, C. (2011). Using safety stocks and simulation to solve the vehicle routing problem with stochastic demands. *Transportation Research Part C: Emerging Technologies*, 19(5):751–765.

Juan, A. A., Goentzel, J., and Bektaş, T. (2014). Routing fleets with multiple driving ranges: Is it possible to use greener fleet configurations? *Applied Soft Computing*, 21:84–94.

Ke, L., Zhai, L., Li, J., and Chan, F. T. (2016). Pareto mimic algorithm: An approach to the team orienteering problem. *Omega*, 61:155 – 166.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4.

Lau, H. C., Yeoh, W., Varakantham, P., Nguyen, D. T., and Chen, H. (2012). Dynamic stochastic orienteering problems for risk-aware applications. *CoRR*, abs/1210.4874.

Lin, S. (2013). Solving the team orienteering problem using effective multi-start simulated annealing. *Applied Soft Computing*, 13:1064–1073.

Panadero, J., de Armas, J., Currie, C., and Juan, A. (2017). A simheuristic approach for the stochastic team orienteering problem. In Chan, W., D'Ambrogio, A., Zacharewicz, G., Mustafee, N., Wainer, G., and Page, E., editors, *Proceedings of the Winter Simulation Conference*, pages 3208–3217, Piscataway, New Jersey. Institute of Electrical and Electronics Engineers, Inc.

Papapanagiotou, V., Montemanni, R., and Gambardella, L. (2014). Objective function evaluation methods for the orienteering problem with stochastic travel and service times. *Journal of Applied Operations Research*, 6:16–29.

Tang, H. and Miller-Hooks, E. (2005a). Algorithms for a stochastic selective travelling salesperson problem. *Journal of the Operational Research Society*, 56:439–452.

Tang, H. and Miller-Hooks, E. (2005b). A tabu search heuristic for the team orienteering problem. *Computers & Operations Research*, 32(6):1379–1407.

Teng, S., Ong, H., and Huang, H. (2004). An integer l-shaped algorithm for time-constrained traveling salesman problem with stochastic travel and service times. *Asia-Pacific Journal of Operational Research*, 21:241–257.

Varakantham, P. and Kumar, A. (2013). Optimization approaches for solving chance-constrained stochastic orienteering problems. In *Algorithmic Decision Theory*, volume 8176, pages 387–398. Springer.

Verbeeck, C., Vansteenwegen, P., and Aghezzaf, E.-H. (2016). Solving the stochastic time-dependent orienteering problem with time windows. *European Journal of Operational Research*, 255:699–718.

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