# A Strategic Oscillation Simheuristic for the Time Capacitated Arc Routing Problem with Stochastic Demands

Peter Keenan[a], Javier Panadero[b,c,*], Angel A. Juan[b,c],
Rafael Martí[d], Seán McGarraghy[a]

*(a)* School of Business, University College Dublin, Dublin, Ireland
*(b)* IN3, Universitat Oberta de Catalunya, Barcelona, Spain
*(c)* Euncet Business School, Terrassa, Spain
*(d)* Universitat de València, Spain

{peter.keenan, sean.mcgarraghy}@ucd.ie, {jpanaderom, ajuanp}@uoc.edu, rafael.marti@uv.es

**Abstract**

The Time Capacitated Arc Routing Problem (TCARP) extends the classical Capacitated Arc Routing Problem by considering time-based capacities instead of traditional loading capacities. In the TCARP, the costs associated with traversing and servicing arcs, as well as the vehicle's capacity, are measured in time units. The increasing use of electric vehicles and unmanned aerial vehicles, which use batteries of limited duration, illustrates the importance of time-capacitated routing problems. In this paper, we consider the TCARP with stochastic demands, i.e.: the actual demands on each edge are random variables which specific values are only revealed once the vehicle traverses the arc. This variability affects the service times, which also become random variables. The main goal then is to find a routing plan that minimises the expected total time required to service all customers. Since a maximum time capacity applies on each route, a penalty time-based cost arises whenever a route cannot be completed within that limit. In this paper, a strategic oscillation simheuristic algorithm is proposed to solve this stochastic problem. The performance of our algorithm is tested in a series of numerical experiments that extend the classical deterministic instances into stochastic ones.

***Keywords:*** *Capacitated Arc Routing Problem, Time-based Capacities, Stochastic Optimization, Simheuristics.*

## 1 Introduction

As Corberán and Laporte (2014) described in their reference book in vehicle routing, "the study of modern arc routing truly started in 1960 with the first publication on the Chinese Postman Problem". Over the years, arc routing has evolved into a relevant research area, which might include real-life characteristics such as multiple criteria, soft constraints, or stochastic travel times, just to name a few. In general terms, in Arc Routing Problems (ARPs) the objective is to traverse a set of connecting edges (in the undirected case) or connecting arcs (in the directed one) at the minimum possible cost. We can find many types of constraints modeling different applications, from garbage collection to meter reading, and the advances in optimization permit to obtain high-quality solutions in most of the cases.

ARPs can be formally defined on a graph $G = (V, E)$, where $V$ is the vertex set (including the depot $d$ where routes start and end), and $E$ is the set of edges (if $G$ is undirected, or arcs if it is directed). For each edge $e \in E$, there is a non-negative value $c(e)$ giving the distance-based or time-based cost of traversing it (similarly for arcs). In the Capacitated Arc Routing Problem (CARP), capacity constraints limit the volume that can be carried on each vehicle. Vehicles are typically identical, and are based at the depot. As is the case with many ARPs, this variant is *NP-hard* and, therefore, optimal solutions are difficult to obtain for medium- and large-size instances of real applications. This is why much of the research in the scientific literature concerns methods to find good lower bounds, or heuristics to obtain near-optimal solutions (Wøhlk, 2008).

Muyldermans and Pang (2014) identified more than 80 publications dealing with different variants of the CARP, and proposed five categories to classify them according to the characteristics of the network, vehicles, facility, demand, and objective. In particular, the *network* can be *directed* or *undirected*, or combining both cases in a mixed graph. *Vehicles* are typically identical, but we can also find a *heterogeneous fleet* in some applications, in which some types of vehicle cannot service certain arcs (*vehicle dependencies*). Vehicles can also be *multi-compartment*, having the required edges a demand for different commodities that must be kept segregated during the transportation action. *Multiple or mobile depots* are characteristics related to the *facilities*, in which we consider several depots or intermediate facilities, respectively. Vehicles can be emptied or replenished in these facilities. The *demand* contains many subcategories, reflecting the interest of researchers and practitioners on these topics. Specifically, among the most important problems is that of *split delivery*, a well known problem in which the constraint that each required edge is serviced by a single vehicle is relaxed. The CARP with *time windows* has also received a lot of attention. That is also the case of the *time-dependent service cost*, where time windows with a linear penalty cost is considered. In the *periodic* CARP, each required edge has a given service frequency over a pre-specified planning horizon. A realistic CARP arises in applications that are better modeled with random variable demands on the edges. The *stochastic demand* reflects very well practical situations in waste collection or snow removal. Finally, in the fifth category, *objective*, the authors identified *multiple objectives*, including: service and labor cost, overtime and penalty costs, length and duration of routes, etc. The inclusion of *profits* in the edges constitutes also an interesting case in which vehicles do not have to service all demand edges, and profits are also collected from some edges. In line with previous studies (Dror, 2012), it is clear that the diverse nature and different characteristics of all these CARP variants means that approaches customized for one application are often less suitable to other problems.

In spite of the range of problems, the CARP remains less studied than its node-based counterpart, the Capacitated Vehicle Routing Problem (CVRP or simply VRP), which in real-life applications might require to consider alternative solutions (Juan et al., 2009). In this paper, we consider the situation where the capacity constraint on the vehicle is a time limitation rather than a volume one, which is called the Time Capacitated Arc Routing Problem (TCARP). A pure TCARP arises in problems where volume constraints are not relevant, for instance: meter reading (Corberán et al., 2019), rail inspection (Batista et al., 2019), or road inspection (Marzolf et al., 2006). Modern applications of the TCARP can also involve the use of unmanned aerial vehicles (Bayliss et al., 2020). In the TCARP, each edge $e \in E$ has a non-negative travel time $t_e$ representing the time required to traverse it. It also may have a time-based demand $q_e$, which represents the time required to service it ($q_e \geq t_e$, since service time includes traversing time). In CARP, an edge may be traversed but not serviced by a route, which is referred to as deadheading by that route (Figure 1). The objective in the TCARP is to find a set of routes minimizing the total time employed in servicing required edges, where each one is serviced by exactly one route while the time capacity $Q > 0$ of each vehicle is not exceeded. Note that this total time includes both time required for servicing edges and for traversing deadhead edges. The distance traveled is not directly minimized in the TCARP. However, minimizing total time usually produces solutions (sets of routes) where distance is traveled efficiently. In situations where vehicle time capacity is critical, classic volume-based methods will tend to produce infeasible solutions, with routes exceed the available time. Although in some instances the difference between time- and volume-based approaches may be small, replacing units of time for units of volume does not recognize the specific characteristics of the TCARP (De Armas et al., 2019). Consequently, while approaches for the volume based CARP provide a guide to useful strategies for the solution of TCARP problems, they also require a significant modification.

Literature on the TCARP is scarce. Some antecedents can be traced back to Keenan and Naughton (1996), who used a heuristic approach for rural postal delivery problems. Postal delivery

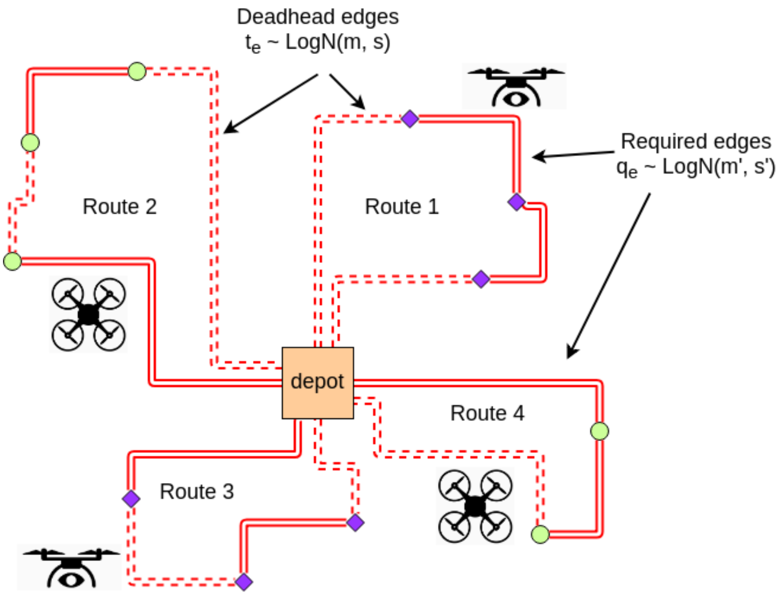



Figure 1: Example of a simple TCARP with four routes.

problems constitute one of the most important areas of application of routing models. They are usually volume constrained if, for example, delivery takes place on foot (Irnich, 2008). On the other hand, van delivery fits better in the time-capacitated model (Keenan and Naughton, 1996), where working time limits are the main constraint on the routes. Also, Keenan (2001, 2005a) tested graph theory based lower bounds. Kirlik and Sipahioglu (2012) introduced a related problem, the CARP with Deadheading Demand (CARPDD), for which Bartolini et al. (2013) proposed lower bounds and an exact algorithm. Specifically, the TCARP can be considered a particular case of the CARPDD, in which the edge demands and the objective function have the same (time) units. These authors provided data sets, lower bounds, and some optimal solutions for the TCARP. Recent research has shown an increasing interest in considering working time as an additional constraint in a volume based problem. Cortinhal et al. (2016) modeled waste collection with concern for total working time and the balance between the working time for different routes. Wøhlk and Laporte (2018) examined a refuse collection CARP with duration constraints on large networks. Tirkolaee et al. (2018) addressed a waste collection problem with driver and crew working time limits. Willemse and Joubert (2019) examined Mixed Capacitated Arc Routing Problems under time restrictions, which required consideration of both deadheading and service time on routes. There are other CARP examples where different road conditions mean differing parameters for service at different times of the day. Thus, for example, the recent paper by Jin et al. (2020) included a time-dependent penalty cost.

In De Armas et al. (2019), the authors propose a metaheuristic algorithm to deal with large-scale instances of the TCARP. Our paper builds upon their approach by extending the previous metaheuristic into a simheuristic algorithm (Juan et al., 2018) able to tackle a stochastic version of the TCARP in which demands are random variables following a known probability distribution. Thus, the exact value of these demands is not discovered until the edge has been traversed, which has a direct influence on the service times. Notice that these time-based demands, $q_e$, are now considered as random variables (which are usually represented by uppercase letters), i.e., $Q_e = t_e + D_e$, being $D_e \geq 0$ a random variable representing the pure service time (without considering the traversing time, $t_e$). Working under more realistic uncertainty conditions (e.g., stochastic demands and service times), might impose additional difficulties whenever a maximum time per route is considered. In effect, there are scenarios in which drivers cannot work for more than a given number of hours, and others in which a time limitation is imposed by the fact of using batteries – as in the case of electric vehicles and unmanned aerial vehicles. In those scenarios, random service times might generate route failures, i.e., routes that cannot be completed due to the total time exceeding a time threshold. The existence of these route failures might require corrective actions which, in turn, will increase the time-based cost of the routing solution. This justifies the need for reliable solutions with low expected time-based costs. Hence, the main contributions of this paper are: *(i)* it introduces a more realistic version of the TCARP in which random demands are considered; *(ii)* it proposes a simheuristic algorithm, which integrates simulation into a metaheuristic framework, to deal with the TCARP with Stochastic Demands; *(iii)* it proposes a set of stochastic benchmarks that extend, in a natural way, the deterministic ones from the TCARP literature; and *(iv)* it introduces reliability concepts to better assess the quality of the obtained solutions.

The remaining of the paper is structured as follows: Section 2 provides a literature review on stochastic CARP variants. Section 3 describes the constructive heuristic that we apply to solve the deterministic version of the TCARP. Section 4 can be considered the core of the paper, since it is devoted to describe the simheuristic algorithm that we propose to solve the TCARP with Stochastic Demands. Finally, Section 5 presents a number of computational experiments to asses the merit of our proposal, while the results are analyzed in Section 6. Section 7 discusses how our algorithm can be extended to consider loading capacity constraints as well. The paper finishes with the associated conclusions and future work in Section 8.

# 2 Related Work on the Stochastic CARP

Classical optimization methods encounter grave difficulties when dealing with optimization problems with uncertainties. It is well documented that stochastic elements make practical problems much more challenging, making them inaccessible to modeling except by resorting to more comprehensive tools, such as computer simulation. This is probably why the literature on stochastic versions of the CARP is limited, when compared with the deterministic versions reviewed above. We now describe the few previous publications.

In their seminal paper, Fleury et al. (2002) modeled the random quantities in the stochastic CARP by Gaussian random variables, based on the central limit theorem. Note that the demand on each edge is the sum of multiple elementary demands (say for instance waste containers or plastic bags in waste collection). Therefore, the application of the theorem results in a Normal distribution. The main objective was to evaluate the robustness of the solutions obtained when the demands are stochastic. Fleury et al. (2005) proposed a hybrid Genetic Algorithm to obtain robust solutions to the stochastic CARP. The authors customized the Genetic Algorithm framework proposed by Lacomme et al. (2001) for the deterministic CARP to deal with random demands. Specifically, the Normal distribution describing the demand in each edge is truncated, avoiding negative values or demands exceeding the vehicle capacity. The method was tested on classical CARP instances (Belenguer and Benavent, 2003).

Christiansen and Lysgaard (2007) proposed a column generation method based on the following formulation as a set partitioning problem. The authors first define $R$ as the set of routes $r = (d, i_1, \ldots, i_k, d)$, where $i_1, \ldots, i_k \in V$. Each route $r$ has an expected cost, $c_r$. The expected demand is computed as the sum of the expected values $E[D_e]$ in each edge $e = (i_j, i_{j+1})$ of the route. Only feasible routes are considered in $R$, i.e.: routes with expected demands within the capacity limit $Q$. Let $a_{ir}$ be a parameter of value 1 if route $r$ visits node $i$, and 0 otherwise. Then the problem can be formulated as follows:

$$\begin{aligned}
\min \quad & z(x) = \min_{r \in R} c_r x_r \\
\text{s.t.} \quad & \sum_{r \in R} a_{ir} x_r = 1 \quad \forall i \in V \\
& x_r = \{0, 1\} \qquad \forall r \in R
\end{aligned}$$

Christiansen et al. (2009) extended their previous work and proposed a Branch-and-Price algorithm in which the pricing incorporates demands with stochastic nature. We can also find developments in the metaheuristic arena for the stochastic version of the CARP. Laporte et al. (2010) applied the Adaptive Large Neighborhood Search (ALNS) framework to solve the undirected CARP in the context of garbage collection. Specifically, the authors included, in their ALNS method, the expected cost of recourse by extending the concept of route failure commonly found in stochastic node-routing problems. As it is customary in heuristic papers, the authors presented computational results to show the merit of their ALNS. Their analysis reveals that ALNS solutions are better than those obtained when considering first a deterministic CARP and then computing the expected cost of recourse by using random variables for the demands.

As may be expected, we can find different applications in which the stochastic CARP is adapted to model a particular problem. Chen et al. (2009) solved a real-world application in small-package delivery, where uncertainty is addressed into a model called Probabilistic Arc Routing Problem. The authors adapted a local search that was primarily designed for the Probabilistic Traveling Salesman Problem (Bertsimas and Howell, 1993). Another interesting application can be found in Ismail and Ramli (2011), who considered a rich CARP based on waste collection operations. The authors studied weather-base variants of routing problems. Specifically, they addressed the particular case of how rain drops affect the weight of the collected waste. Their heuristic method is built from a constructive heuristic called Nearest Procedure Based on Highest Demand/Cost. Also, Gonzalez-Martin et al. (2018) introduced a simheuristic algorithm for the CARP with Stochastic Demands. The authors proposed to adapt the metaheuristic framework to deal with uncertainties. In particular, their method is based on the well-tested RandSharp algorithm for the deterministic CARP (Gonzalez-Martin et al., 2012). In the present work, we consider stochastic demands for the TCARP.

# 3 A Savings-based Heuristic

This section describes the constructive heuristic that we apply to solve the deterministic version of the TCARP. This heuristic for the TCARP is based on the savings heuristic for the CARP proposed in Gonzalez-Martin et al. (2012). The resulting adaptation is then extended by combining it with the augment-merge heuristic proposed by Lacomme et al. (2004). Algorithm 1 depicts the pseudo-code of our approach, which we describe now in more detail. Given a graph $G$, which might be complete or not, the Floyd-Warshall algorithm (Pallottino, 1984) is used to compute the shortest paths –based on traversing time– for all pairs of nodes, $i, j \in G$ (see line 1 in Algorithm 1). This allows us to consider the graph as a complete one. At this point, we compute the savings associated with each path connecting two different nodes related to a requiring edge, $e = (i, j)$ (lines 2 and 3). The savings associated with a requiring edge $e$ are computed using the following expression:

$$s(e) = t(depot, i) + t(j, depot) - t_e$$

Then, required edges are sorted in a decreasing order according to the value of their associated savings (line 4). At this point, we construct a 'dummy' set of single-edge routes (line 5) by considering each requesting edge, and their connection paths to the depot. The following *augment* procedure is now applied over the dummy solution (line 6): *(i)* routes in the dummy solution are sorted by total time (from higher to lower); *(ii)* each route is scanned to identify those edges –belonging to the scanned route– that are also requesting edges in other routes of the dummy solution; and *(iii)* whenever one of these edges is identified, it is serviced in the scanned route if that does not violate any constraint –i.e., the largest route 'absorbs' the shortest one if possible. In our experiments, we run the algorithm with and without this augment procedure, since its performance depends on the specific instance being solved. Now, a route-merging process starts. This is an iterative process (lines 7 to 22). In this process, for each edge $e$ in the list of savings, its two extreme nodes, $i$ and $j$ are considered. All routes (if any) traversing each of these extremes, and having the corresponding node as an 'exterior' one, are merged to construct a new route $mR$ traversing $e$. In this context, an exterior node in a route $R$ is one that is connected to the depot

**Algorithm 1** Heuristic approach

```
1: times ← algFloyd-Warshall(graph) % Use Floyd-Warshall algorithm to compute the shortest
   paths for all pair of nodes i, j in the graph
2: reqEdges ← getReqEdges(graph) %Obtain the requesting Edges
3: savings ← computeSavings(reqEdges, times, depot)
4: sortSavingByDecreasingOrder(savings)
5: routes ← genDummyPartialRoutes(reqEdges) %Generate dummy 'partial routes', where each
   reqEdge e is a route
6: AugmentRoutes ← applyAugmentProcedure(routes) %Apply Extended-Augment phase
7: for (edge e ∈ savings) do %Starts route-merging process
8:     i ← origin(e)
9:     j ← end(e)
10:    for each route aR in AugmentRoutes with aR traversing i do
11:        iR ← aR
12:        for each route bR in AugmentRoutes with bR traversing j do
13:            jR ← bR
14:            if isMergePossible(iR, jR, vCap) then
15:                mR ← mergeRoutes(iR, jR, i, j)
16:                routes ← delete(iR)
17:                routes ← delete(jR)
18:                routes ← add(mR)
19:            end if
20:        end for
21:    end for
22: end for
23: sol ← reconstruct(routes, graph)
24: return sol
```

through a path that does not contain any requesting edge. Thus, using exterior nodes during the merging process of routes $iR$ and $jR$ allows for the exclusion of entire sections from these routes without losing any edge-service already assigned to them. At this point, the algorithm checks if the new route $mR$ does not violate the time-based capacity of the vehicle, $vCap$ (line 15). In case that all the constraints are satisfied, the new route ($mR$) is added to the current solution in substitution of the two routes that were merged, which are deleted (lines 16 to 18).

In order to complete a merge of two routes, it might be necessary to change the orientation in which one of the routes is traversed –which, in turn, defines the orientation in which the connecting edge must be traversed. This process continues until either a successful merge is attained or no more routes satisfy the initial conditions. The final step (line 23) consists in reconstructing the final solution, by adding the missing connection paths among the merged routes. This procedure is carried out by computing the shortest path between the edges in the route, using all-pairs shortest paths generated at the beginning of the procedure. Once the algorithm is completed, the solution is returned (line 24) as the output of the method.

# 4 The Strategic Oscillation Simheuristic

Based on the deterministic heuristic introduced in the previous section, we propose a simheuristic approach to solve the stochastic TCARP. Simheuristic algorithms combine metaheuristics with simulation, and have been recently employed to solve stochastic optimization problems in areas as diverse as vehicle routing problems (Guimarans et al., 2018), inventory routing problems (Gruler et al., 2018, 2020), project management (Panadero et al., 2020), e-commerce network design (Pagès-Bernaus et al., 2019), or waste collection management (Gruler et al., 2017a,b). Specifically, we propose an oscillation pattern to create a balance between search intensification and diversification. *Strategic oscillation* (SO) is closely linked to the origins of Tabu Search (Glover and Laguna, 1998) and operates by orienting moves in relation to a critical level, as identified by a stage of construction. Constructive and destructive oscillation, where constructive steps add elements and destructive steps drop them, are very popular to enhance classic constructive processes. We implemented a simplified and effective SO, known as iterated greedy or IG (Ruiz and Stützle, 2007), which obtains high-quality solutions by iterating over a greedy constructive heuristic. The strategic oscillation uses two main phases: destruction and construction. The greedy heuristic in our case is the extended heuristic (tested with and without the augment procedure) described above. Algorithm 2 depicts the main characteristics of our approach, composed of three stages. First, a feasible initial solution is generated using a constructive heuristic. Then, the IG metaheuristic improves this initial solution by iteratively exploring the search space. This process conducts a small number of Monte Carlo simulation (MCS) runs to obtain rough estimates of the solution performance under stochastic conditions. This allows us to generate a pool of 'promising' elite solutions. In the third stage, a refinement procedure, using a larger number of MCS runs, is applied to these elite solutions. The latter step provides a more accurate estimation of the expected total cost (Rabe et al., 2020). We now describe the previous steps in more detail.

As shown in Algorithm 2, our method applies the extended savings heuristic in line 1 to obtain an initial solution, *InitSol*. It is computed using the expected demand of the random variables, $E(D_e)$. In other words, we first solve the deterministic version of our problem in which we simply have an expected (average) value $d_e$. Once the initial solution is obtained, we submit it to a standard evaluation process using stochastic data. In particular, we use MCS to simulate many scenarios (values of the random variables) and compute the average objective function value (total travel time) across all the scenarios. To keep the computational time low, we limit this simulation to 100 replications or scenarios. This process is called *FastSimulation* in line 2 of the algorithm.

One of the main advantages of simheuristics is that the objective function information is supplemented with risk analysis (Ferone et al., 2019). In our problem, random times in the demands may cause a failure in the routes, since vehicles have a maximum traveling time $Q$. Therefore, when we submit a solution to the MCS, in some of the scenarios a route may turn out to be infeasible because its total traveling time exceeds $Q$. In that case, a corrective action has to be taken, which implies an early termination of the route: the vehicle must return to the depot without visiting all the edges in order to satisfy the time constraint. Then, a new vehicle goes to the route to complete it from that point on. This extra cost of going to and returning from the depot has to be added to the total time. Hence, it becomes part of the average expected cost returned by the MCS. Additionally, we compute the fraction of times (percentage of scenarios) that the solution is feasible, and where no corrective action is needed. Hence, simulation allows us to estimate reliability of a complex process (Faulin et al., 2008). This reliability value provide a measure of the solution robustness across different scenarios.

The initial solution reliability is set as a threshold, *reliableThreshold*, to filter the solutions from subsequent iterations. Our method collects the best solutions identified in the search process, and saves them in a pool of solutions (*poolBestSols*). We do not admit solutions with a reliability value lower than the threshold into this pool. As the search progresses, we increment the reliability threshold by 0.1 every 50 iterations, and only very robust solutions will become part of this elite set. The main loop of the method (lines 6 to 20 in Algorithm 2) performs construction and destruction steps according to the strategic oscillation principle. In particular, we first remove some elements from the *baseSol*, which originally is the initial solution. An interesting point is

**Algorithm 2** Simheuristic approach

```
1: initSol ← BR-Heuristic(inputs,β) % Randomized heuristic
2: reliableThreshold ← fastSimulation(initSol)
3: baseSol ← initSol
4: bestSol ← initSol
5: t_elapsed ← 0
6: while (t_elapsed < t_max) do % SO Stage
7:     newSol ← oscillation(baseSol,inputs)
8:     newSol ← localSearch(newSol)
9:     δ ← getDetTotalTime(newSol) - getDetTotalTime(baseSol)
10:    if (δ < 0) then
11:        fastSimulation(newSol) % Monte Carlo simulation
12:        if (getStochTotalTime(newSol) - getStochTotalTime(bestSol) < 0) and
13:           (Reliability(newSol) < reliableThreshold) then
14:            bestSol ← newSol
15:            insert(poolBestSols, bestSol)
16:        end if
17:    end if
18:    baseSol ← AcceptanceCriteria(newSol,bestSol)
19:    t_elapsed ← update(t_elapsed)
20: end while
21: for (sol ∈ poolBestSol) do % Refined simulation
22:    deepSimulation(sol)
23:    if (getStochTotalTime(sol) < getStochTotalTime(bestSol)) then
24:        bestSol ← sol
25:    end if
26: end for
27: return bestSol
```

how to select the elements to be removed from the solution. We consider two different strategies here. The first one, called *random*, simply removes $k$ routes in the *bestSol* at random –where $k$ is a random number between 2 and the number of routes in the solution. The second strategy, called *selective* computes, for each route, the difference between the vehicle capacity (in terms of time) and its total traveling time. We then order the routes in decreasing order with respect to this difference and select one at random from the first $k$. The rationale behind this step is that large differences represent an opportunity to improve the route, while low differences imply compact routes with good allocation of edges. Note that, in this case, the parameter $k$ has a different role than in the previous strategy. Then, we also remove from the solution the two closest routes to the selected one. To do that, we compute the distance between two routes as the minimum between the pairs of nodes, each one in a different route. Removing the three routes from the solution may provide a significant change in the solution, which could permit us to create a different one.

In both strategies, *random* and *selective*, once the selected routes have been removed we obtain a partial solution. This partial solution needs to be completed with new routes to serve the required arcs. At this point, we apply the extended heuristic to reconstruct the partial solution. This creates new routes to visit the nodes that were in the removed routes. Hence, a new solution is generated. In this way, we complete an oscillation step. It is worth mentioning, that the random strategy has a diversification component –searching for new combination of routes–, while the selective strategy is based on intensifying the search –looking for improved outcomes. Both components are studied in our computational experiments. The acceptance-criterion of our SO (line 20 in Algorithm 2) establishes the rules by which the algorithm wanders over the regions of the search space in the quest for better solutions. The following criteria are analyzed: *(i) replace if better* –i.e., the new solution is accepted only if it attains a better objective value; and *(ii) random walk*, –i.e., it always selects the new solution, regardless of its objective function value, which prevents the method from being confined in the area of one local optimum.

# 5 Computational Experiments

The proposed simheuristic has been implemented using Java SE 8.0 and tested on a workstation with a multi-core processor Intel Xeon E5-2650 v4 and using 32GB of RAM. To the best of our knowledge, there are no TCARP instances using random time-based capacities that we can use as benchmarks. Accordingly, we have extended a deterministic set of previously published TCARP and CARPDD data sets. The extension incorporates both random servicing times and traversing times. In particular, the following data sets have been used:

- The *tcarp* data set, introduced by Keenan (2005a), and available at Keenan (2005b), contains 10 very sparse networks with up to 44 nodes and 50 edges derived from real rural networks in Ireland. In the corresponding graphs, all edges are required and represent roads containing a number of customers that must be visited by a fleet of vehicles based at the depot. The traversing time $t_e$ of an edge $e$ is related to its length, and the service time $q_e$ is equal to $t_e$ plus the time for visiting all customers on edge $e$. Both traversing and service times, as well as the vehicle capacity $Q$, represent times expressed in minutes. For each network, three instances are derived by setting $Q$ equal to 40, 50, or 60 min, respectively. Results (lower and upper bounds) for this data set are available in Bartolini et al. (2013).

- The *tegl* data set, proposed by Bartolini et al. (2013), uses the networks of the *egl* data set (Li and Eglese, 1996). The input data for these instances was generated as follows: *(i)* for each edge $e \in E$, its length $l_e$ (in km) is defined as $l_e = 0.1 \cdot c_e$ –where $c_e$ represents the edge cost in the original *egl* data set; *(ii)* the traversing time $t_e$ of edge $e$ is set equal to $0.35 \cdot 60 \cdot l_e$ to simulate the time (in minutes) needed to traverse the edge by a vehicle with speed 35 km/h; *(iii)* for each required edge $e \in E_{req}$, a random number of customers $s_e \in [1, 5]$ is assigned to $e$; and *(iv)* for required edges, the service time $q_e$ is set equal to $t_e + 5 \cdot s_e$ –i.e., they assume a 5-min stop at each customer. Finally, for each network two instances are created by setting the time-based capacity $Q$ equal to 240 and 360 min, respectively. Updated results for this data set (lower and uppers bounds) are available in Bartolini et al. (2013).

- The *val* data set, introduced by Kirlik and Sipahioglu (2012), contains 34 instances defined on 10 different graphs. For each graph, different instances were created by changing the capacity of the vehicles and all edges are required in the *val* data set. These instances consist of 24 to 50 nodes and 34 to 97 edges. From this data set, we have selected the large-sized instances, which are expected to be the most challenging ones. Updated results for this data set (lower bounds) are available in Kirlik and Sipahioglu (2012).

- The *rural* data, available at Keenan (2017), contains a large-scale sparse instances based on a simplified version of real Irish rural road networks, extracted from a Geographic Information System (GIS). The data set represents rural postal delivery, where a postman in a van may service several hundred houses widely distributed over a very sparse road network. The main constraint is the length of a working day. As the postman must sort his / her post before driving, the typical day left for driving is 6 to 7h. A long road may have several edges, as sections of a road need to be identified for address purposes. Consequently, there are many nodes of degree two in the network. The average degree of these rural local networks is around 2.5, compared to around 3 for typical main road networks and 3.5 for cities built on a block pattern. These large networks are similar to the *tcarp* data set, which was previously made available Keenan (2005b) as a small example of this type of network. Three instances are then created by setting $Q$ equal to 360, 400, and 420 min, respectively. The best known solutions (BKS) for this data set are available in De Armas et al. (2019).

Regarding the algorithm design, after some tests we have decided to use the *random* choice for the destruction process, as well as the *replace-if-better* choice for the acceptance criterion.

Table 1 summarizes the main characteristics of these sets: number of instances, minimum number of nodes, maximum number of nodes, minimum number of edges, maximum number of edges, and time-based capacity. Notice that not all instances in a dataset use the same value for the time capacity $Q$. Hence, the table shows the different $Q$-values employed in each dataset.

Table 1: Summary of data sets.

| Dataset | Instances | Min. Nodes | Max. Nodes | Min. Edges | Max. Edges | Time capacities (Q) |
|---|---|---|---|---|---|---|
| *tcarp* | 10 | 17 | 44 | 24 | 50 | 40, 50, 60 |
| *tegl* | 2 | 77 | 140 | 98 | 190 | 240, 360 |
| *val* | 5 | 30 | 50 | 50 | 97 | 144 - 918 |
| *rural* | 2 | 221 | 599 | 245 | 872 | 360, 400, 420 |

To extend the aforementioned instances, we have assumed that the time delay, $D_e$, follows a Log-Normal probability distribution. The Log-Normal distribution is a natural choice for describing non-negative random variables, such as times. In the real-world, historical data could be used to determine the most appropriate distribution for the time requested to traverse or service each edge. The Log-Normal distribution has two parameters: the location parameter, $\mu$, and the scale parameter, $\sigma$, which relate to the expected value $E[D_e]$ and the variance $Var[D_e]$ as follows:

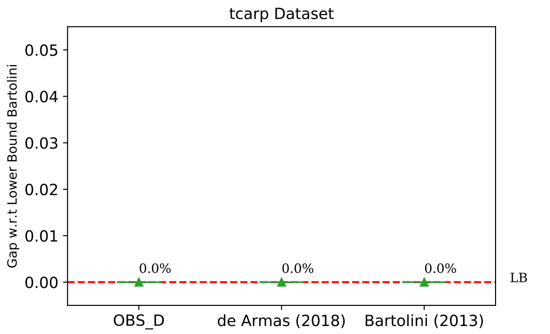
$$\mu_e = \ln(E[D_e]) - \frac{1}{2}\ln\left(1 + \frac{Var[D_e]}{E[D_e]^2}\right) \tag{1}$$

$$\sigma_e = \left|\sqrt{\ln\left(1 + \frac{Var[D_e]}{E[D_e]^2}\right)}\right| \tag{2}$$

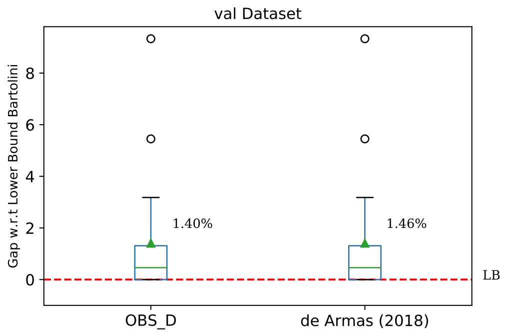
We have set the variability in the delay of travel / servicing times with reference to the deterministic equivalents such that $Var[D_e] = c \cdot t_e$ and $c \geq 0$. The parameter $c$ is a design parameter that allows us to experiment with different levels of uncertainty. It is expected that, as $c$ converges to zero, the results from the stochastic version converge to those obtained in the deterministic scenario. We consider three different levels of uncertainty: low ($j = 0.1$), medium ($j = 1$), and large ($j = 10$). Tables 2 to 5 present the results for the different sets of instances in a scenario with a large level of uncertainty ($c = 10$). The first column of these tables identifies the instance, while the second column depicts the time-based capacity, $Q$. We have divided the rest of columns into two different parts. In the first part of each table, we validate our approach on the original TCARP instances –i.e., without considering stochasticity. Thus, the OBS-D column shows our best deterministic solution, and the OBS-D$_t$ column displays the computational time –in seconds– to reach it. The next columns correspond to previously published deterministic results, which include: *(i)* the results provided by De Armas et al. (2019) for all the tested data sets; *(ii)* the results provided by Bartolini et al. (2013) for the *tcarp*, *tegl*, and *val* data sets; and *(iii)* the results provided by Keenan (2005a) for the *rural* data set. Subsequently, the next columns show the corresponding gap between our approach (OBS-D) and other state-of-the-art methods. The second part of the tables reports the results obtained for the TCARP stochastic instances considering random delays and overtime penalties. The OBD-D-S column shows the expected cost obtained when the best deterministic solution (OBS-D) is evaluated under a stochastic scenario, with the corresponding level of uncertainty. To compute this column, we have executed just the algorithm disabling the simulation part (fast simulation process), and we have applied the 'intensive' simulation process to the best deterministic solution. The next two columns ($F1rel[6]$ and $F2rel[7]$) show the reliability of OBD-D-S. We have considered two different ways to measure the reliability. $F1rel$ shows the reliability measured in terms of the percentage of routes that are completed without violating the time capacity $Q$. The $F2rel$ displays the reliability measured in the same way than $F1rel$, but considering that if the demand capacity constraint is violated in the last edge –which connects with the depot– the route does not incur into a failure. This reliability could be seen as a relaxation of the capacity constraint. Similarly, column OBD-S-S shows the expected cost obtained for the stochastic TCARP using the solution provided by our simheuristic approach. The next two columns ($F1rel$[9] and $F2rel$[10]) show their associated reliabilities. Finally, the last column reports the computational time to reach the OBD-S-S solution. Regarding *tgl* instances, despite they were not solved in De Armas et al. (2019), we have applied their algorithm and code to obtain and report their results.

# 6 Analysis of Results

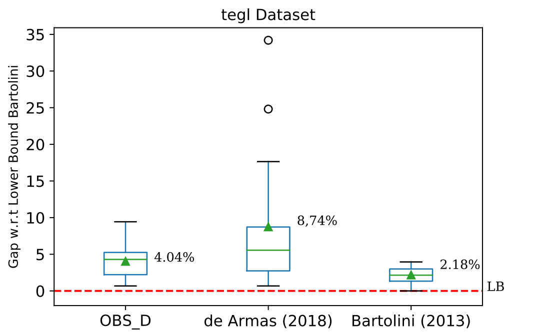
The results show that our approach provides high-quality solutions for the deterministic TCARP. Figure 2 depicts an overview of tables 2 to 5 about the performance of our algorithm in a deterministic scenario. In these box-plots, the vertical axis represents the gap obtained with respect to the lower bounds (LB) reported in the literature. Notice that our algorithm reaches the BKS for the *tcarp* data set, and outperform the solutions provided by De Armas et al. (2019) for the other three data sets. In average, we obtain a gap about 1.82%, with a maximum average gap of 4.04% for the *tegl* data set. These results highlight the capabilities of our algorithm.



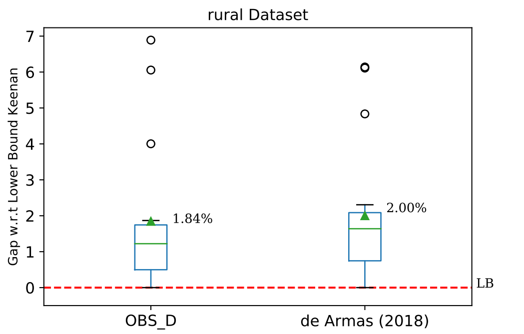

(a) A comparison of *tcarp* instances with respect to lower bound (LB) of Bartolini et al. (2013)




(b) A comparison of *val* instances with respect to lower bound (LB) of Bartolini et al. (2013)




(c) A comparison of *tegl* instances with respect to lower bound (LB) of Bartolini et al. (2013)




(d) A comparison of *rural* instances with respect to lower bound (LB) of Keenan (2005a)

Figure 2: Comparison between our approach and other state-of-the-art approaches for the deterministic TCARP.

Regarding the stochastic scenario –which represents the main contribution of this paper–, results show that the solutions provided by our approach for the stochastic TCARP (OBD-S-S) clearly outperform the solutions for the deterministic TCARP when these are simulated (OBD-S-D). In other words, near-optimal solutions for the deterministic version of the problem might be

Table 2: Computational results for the *tcarp* data set.

| | | Deterministic TCARP | | | | | Gaps (%) | | | Stochastic TCARP | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Q | OBS-D [1] | OBS-$D_t$ (s) | De Armas (2018) [2] | Bartolini (2013) LB [3] | UB [4] | [1-2] | [1-3] | [1-4] | OBS-D-S [5] | F1 Rel. [6] | F2 Rel. [7] | OBS-S-S [8] | F1 Rel. [9] | F2 Rel. [10] | OBS-S-$S_t$ (s) |
| S1 | 60 | 104 | 1 | 104 | 104 | 104 | 0.00% | 0.00% | 0.00% | 356 | 0.52 | 0.60 | 271 | 0.69 | 0.74 | 1 |
| S1 | 50 | 108 | 1 | 108 | 108 | 108 | 0.00% | 0.00% | 0.00% | 290 | 0.78 | 0.79 | 275 | 0.80 | 0.82 | 2 |
| S1 | 40 | 112 | 1 | 112 | 112 | 112 | 0.00% | 0.00% | 0.00% | 362 | 0.69 | 0.71 | 353 | 0.77 | 0.79 | 1 |
| S2 | 60 | 156 | 4 | 156 | 156 | 156 | 0.00% | 0.00% | 0.00% | 365 | 0.77 | 0.78 | 317 | 0.80 | 0.81 | 20 |
| S2 | 50 | 158 | 4 | 158 | 158 | 158 | 0.00% | 0.00% | 0.00% | 424 | 0.75 | 0.77 | 365 | 0.81 | 0.84 | 7 |
| S2 | 40 | 164 | 6 | 164 | 164 | 164 | 0.00% | 0.00% | 0.00% | 505 | 0.71 | 0.75 | 472 | 0.80 | 0.82 | 2 |
| S3 | 60 | 215 | 1 | 215 | 215 | 215 | 0.00% | 0.00% | 0.00% | 546 | 0.72 | 0.73 | 483 | 0.73 | 0.76 | 15 |
| S3 | 50 | 229 | 1 | 229 | 229 | 229 | 0.00% | 0.00% | 0.00% | 625 | 0.69 | 0.72 | 585 | 0.72 | 0.74 | 1 |
| S3 | 40 | 245 | 4 | 245 | 245 | 245 | 0.00% | 0.00% | 0.00% | 793 | 0.70 | 0.74 | 684 | 0.75 | 0.77 | 7 |
| S4 | 60 | 146 | 1 | 146 | 146 | 146 | 0.00% | 0.00% | 0.00% | 306 | 0.76 | 0.75 | 283 | 0.77 | 0.81 | 4 |
| S4 | 50 | 162 | 1 | 162 | 162 | 162 | 0.00% | 0.00% | 0.00% | 320 | 0.80 | 0.84 | 298 | 0.82 | 0.85 | 3 |
| S4 | 40 | 174 | 1 | 174 | 174 | 174 | 0.00% | 0.00% | 0.00% | 434 | 0.75 | 0.82 | 384 | 0.79 | 0.84 | 7 |
| S5 | 60 | 140 | 1 | 140 | 140 | 140 | 0.00% | 0.00% | 0.00% | 326 | 0.75 | 0.78 | 287 | 0.78 | 0.79 | 2 |
| S5 | 50 | 149 | 1 | 149 | 149 | 149 | 0.00% | 0.00% | 0.00% | 414 | 0.63 | 0.66 | 376 | 0.77 | 0.81 | 1 |
| S5 | 40 | 165 | 1 | 165 | 165 | 165 | 0.00% | 0.00% | 0.00% | 465 | 0.73 | 0.78 | 445 | 0.74 | 0.79 | 4 |
| S6 | 60 | 104 | 1 | 104 | 104 | 104 | 0.00% | 0.00% | 0.00% | 395 | 0.76 | 0.82 | 348 | 0.83 | 0.86 | 6 |
| S6 | 50 | 107 | 10 | 107 | 107 | 107 | 0.00% | 0.00% | 0.00% | 238 | 0.77 | 0.79 | 215 | 0.88 | 0.89 | 1 |
| S6 | 40 | 113 | 1 | 113 | 113 | 113 | 0.00% | 0.00% | 0.00% | 297 | 0.71 | 0.73 | 145 | 0.82 | 0.85 | 2 |
| S7 | 60 | 68 | 1 | 68 | 68 | 68 | 0.00% | 0.00% | 0.00% | 127 | 0.93 | 0.94 | 117 | 0.92 | 0.93 | 3 |
| S7 | 50 | 68 | 1 | 68 | 68 | 68 | 0.00% | 0.00% | 0.00% | 122 | 0.91 | 0.93 | 119 | 0.93 | 0.93 | 8 |
| S7 | 40 | 68 | 1 | 68 | 68 | 68 | 0.00% | 0.00% | 0.00% | 170 | 0.83 | 0.83 | 147 | 0.83 | 0.91 | 4 |
| S8 | 60 | 83 | 1 | 83 | 83 | 83 | 0.00% | 0.00% | 0.00% | 172 | 0.84 | 0.86 | 154 | 0.88 | 0.92 | 5 |
| S8 | 50 | 83 | 1 | 83 | 83 | 83 | 0.00% | 0.00% | 0.00% | 182 | 0.75 | 0.83 | 174 | 0.79 | 0.82 | 1 |
| S8 | 40 | 87 | 1 | 87 | 87 | 87 | 0.00% | 0.00% | 0.00% | 195 | 0.78 | 0.84 | 169 | 0.87 | 0.88 | 5 |
| S9 | 60 | 177 | 1 | 177 | 177 | 177 | 0.00% | 0.00% | 0.00% | 337 | 0.66 | 0.70 | 324 | 0.80 | 0.82 | 1 |
| S9 | 50 | 193 | 3 | 193 | 193 | 193 | 0.00% | 0.00% | 0.00% | 426 | 0.70 | 0.73 | 410 | 0.74 | 0.79 | 11 |
| S9 | 40 | 221 | 1 | 221 | 221 | 221 | 0.00% | 0.00% | 0.00% | 484 | 0.74 | 0.77 | 439 | 0.75 | 0.78 | 7 |
| S10 | 60 | 171 | 5 | 171 | 171 | 171 | 0.00% | 0.00% | 0.00% | 408 | 0.67 | 0.73 | 383 | 0.82 | 0.85 | 6 |
| S10 | 50 | 180 | 31 | 180 | 180 | 180 | 0.00% | 0.00% | 0.00% | 449 | 0.74 | 0.77 | 416 | 0.87 | 0.88 | 45 |
| S10 | 40 | 192 | 1 | 192 | 192 | 192 | 0.00% | 0.00% | 0.00% | 515 | 0.69 | 0.74 | 506 | 0.88 | 0.89 | 1 |
| *Average:* | | *145* | *3* | *145* | *145* | *145* | *0.00%* | *0.00%* | *0.00%* | *368* | *0.74* | *0.77* | *331* | *0.81* | *0.83* | *6* |

Table 3: Computational results for the *tegl* data set.

| | | Deterministic TCARP | | | | | Gaps (%) | | | Stochastic TCARP | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Inst. | Q | OBS-D [1] | OBS-$D_t$ (s) | De Armas (2018) [2] | Bartolini (2013) LB [3] | Bartolini (2013) UB [4] | [1-2] | [1-3] | [1-4] | OBS-D-S [5] | F1 Rel. [6] | F2 Rel. [7] | OBS-S-S [8] | F1 Rel. [9] | F2 Rel. [10] | OBS-S-$S_t$ (s) |
| tegl-e1-A | 360 | 1184 | 29 | 1367 | 1162 | 1178 | -13.39% | 1.89% | 0.51% | 1695 | 0.77 | 0.85 | 1225 | 0.99 | 0.99 | 40 |
| tegl-e1-C | 240 | 1589 | 42 | 1655 | 1452 | 1461 | -3.99% | 9.44% | 8.76% | 2296 | 0.71 | 0.76 | 1797 | 0.93 | 0.95 | 50 |
| tegl-e2-A | 360 | 2023 | 27 | 2023 | 1902 | 1921 | 0.00% | 6.36% | 5.31% | 2796 | 0.69 | 0.73 | 2651 | 0.71 | 0.75 | 39 |
| tegl-e2-C | 240 | 2438 | 57 | 2481 | 2323 | 2388 | -1.73% | 4.95% | 2.09% | 3839 | 0.63 | 0.69 | 3627 | 0.68 | 0.74 | 78 |
| tegl-e3-A | 360 | 2425 | 36 | 2439 | 2373 | 2373 | -0.57% | 2.19% | 2.19% | 3509 | 0.63 | 0.67 | 3434 | 0.82 | 0.91 | 41 |
| tegl-e3-C | 240 | 3011 | 13 | 3019 | 2925 | 3000 | -0.26% | 2.94% | 0.37% | 4752 | 0.64 | 0.70 | 4564 | 0.65 | 0.71 | 29 |
| tegl-e4-A | 360 | 2686 | 24 | 2686 | 2658 | 2690 | 0.00% | 1.05% | -0.15% | 3876 | 0.65 | 0.69 | 3686 | 0.71 | 0.75 | 52 |
| tegl-e4-C | 240 | 3270 | 10 | 3270 | 3248 | 3351 | 0.00% | 0.68% | -2.42% | 5215 | 0.60 | 0.67 | 5066 | 0.62 | 0.69 | 26 |
| tegl-s1-A | 360 | 1760 | 43 | 2225 | 1658 | 1721 | -20.90% | 6.15% | 2.27% | 2387 | 0.61 | 0.64 | 2197 | 0.78 | 0.80 | 63 |
| tegl-s1-C | 240 | 2578 | 22 | 3094 | 2479 | 2577 | -16.68% | 3.99% | 0.04% | 3506 | 0.63 | 0.69 | 2984 | 0.87 | 0.95 | 56 |
| tegl-s2-A | 360 | 4023 | 47 | 4114 | 3846 | 3926 | -2.21% | 4.60% | 2.47% | 5166 | 0.69 | 0.72 | 4358 | 0.83 | 0.92 | 82 |
| tegl-s2-C | 240 | 5587 | 67 | 5605 | 5309 | 5413 | -0.32% | 5.24% | 3.21% | 7957 | 0.63 | 0.68 | 7900 | 0.64 | 0.69 | 92 |
| tegl-s3-A | 360 | 4436 | 23 | 4436 | 4213 | 4303 | 0.00% | 5.29% | 3.09% | 6052 | 0.65 | 0.69 | 6067 | 0.65 | 0.68 | 35 |
| tegl-s3-C | 240 | 6105 | 50 | 6122 | 5801 | 5973 | -0.28% | 5.24% | 2.21% | 8879 | 0.61 | 0.66 | 8644 | 0.63 | 0.80 | 58 |
| tegl-s4-A | 360 | 5202 | 15 | 5202 | 5080 | 5190 | 0.00% | 2.40% | 0.23% | 7435 | 0.62 | 0.66 | 7349 | 0.63 | 0.79 | 58 |
| tegl-S4-C | 240 | 7256 | 58 | 7282 | 7098 | 7317 | -0.36% | 2.23% | -0.83% | 10860 | 0.60 | 0.65 | 10767 | 0.60 | 0.82 | 19 |
| *Average:* | | *3473* | *35* | *3564* | *3345* | *3424* | *-3.79%* | *4.04%* | *1.83%* | *5014* | *0.65* | *0.70* | *4770* | *0.73* | *0.81* | *51* |

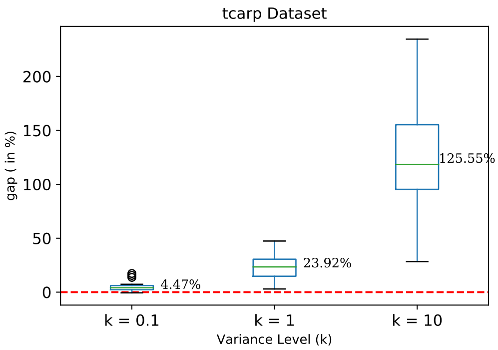
Table 4: Computational results for the *val* data set (CARPDD).

| | Deterministic TCARP | | | | | Gaps (%) | | Stochastic TCARP | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Inst. | Q | OBS-D [1] | OBS-$D_t$ (s) | De Armas (2018) [2] | Bartolini (2013) LB [3] | [1-2] | [1-3] | OBS-D-S (3) [4] | F1 Rel. [5] | F2 Rel. [6] | OBS-S-S [7] | F1 Rel. [8] | F2 Rel. [9] | OBS-S-$S_t$ (s) |
| Val6 | 918 | 221 | 1 | 221 | 221 | 0.00% | 0.00% | 221 | 1.00 | 1.00 | 221 | 1.00 | 1.00 | 1 |
| Val6 | 648 | 223 | 1 | 223 | 221 | 0.00% | 0.90% | 248 | 0.97 | 0.97 | 230 | 0.99 | 0.99 | 1 |
| Val6 | 270 | 239 | 17 | 239 | 239 | 0.00% | 0.00% | 820 | 0.66 | 0.70 | 396 | 0.92 | 0.93 | 21 |
| Val7 | 465 | 279 | 1 | 279 | 279 | 0.00% | 0.00% | 349 | 0.96 | 0.96 | 347 | 0.99 | 0.99 | 2 |
| Val7 | 349 | 283 | 2 | 283 | 283 | 0.00% | 0.00% | 686 | 0.78 | 0.86 | 367 | 0.98 | 0.99 | 16 |
| Val7 | 151 | 352 | 3 | 352 | 347 | 0.00% | 1.44% | 1081 | 0.80 | 0.85 | 678 | 0.99 | 0.99 | 1 |
| Val8 | 579 | 386 | 11 | 386 | 386 | 0.00% | 0.00% | 628 | 0.81 | 0.86 | 409 | 0.99 | 0.99 | 9 |
| Val8 | 434 | 386 | 29 | 386 | 386 | 0.00% | 0.00% | 786 | 0.74 | 0.82 | 601 | 0.94 | 1.00 | 16 |
| Val8 | 188 | 519 | 15 | 519 | 503 | 0.00% | 3.18% | 1525 | 0.68 | 0.79 | 1072 | 0.99 | 0.99 | 16 |
| Val9 | 511 | 324 | 27 | 324 | 323 | 0.00% | 0.31% | 492 | 0.82 | 0.82 | 545 | 0.99 | 0.99 | 49 |
| Val9 | 380 | 327 | 9 | 326 | 326 | 0.31% | 0.31% | 438 | 0.78 | 0.82 | 390 | 0.94 | 0.96 | 23 |
| Val9 | 304 | 334 | 17 | 338 | 332 | -1.18% | 0.60% | 841 | 0.75 | 0.81 | 529 | 0.91 | 0.93 | 26 |
| Val9 | 152 | 445 | 25 | 445 | 422 | 0.00% | 5.45% | 1440 | 0.74 | 0.82 | 1044 | 0.99 | 0.99 | 21 |
| Val10 | 480 | 438 | 19 | 438 | 436 | 0.00% | 0.46% | 709 | 0.84 | 0.87 | 603 | 0.90 | 0.91 | 42 |
| Val10 | 365 | 449 | 25 | 449 | 446 | 0.00% | 0.67% | 836 | 0.76 | 0.80 | 738 | 0.86 | 0.88 | 37 |
| Val10 | 288 | 465 | 15 | 465 | 459 | 0.00% | 1.31% | 680 | 0.72 | 0.79 | 590 | 0.86 | 0.91 | 36 |
| Val10 | 144 | 844 | 30 | 844 | 772 | 0.00% | 9.33% | 1692 | 0.67 | 0.79 | 2191 | 0.73 | 0.83 | 36 |
| *Average:* | | *383* | *15* | *383* | *375* | *-0.05%* | *1.41%* | *828* | *0.79* | *0.84* | *644* | *0.94* | *0.96* | *21* |

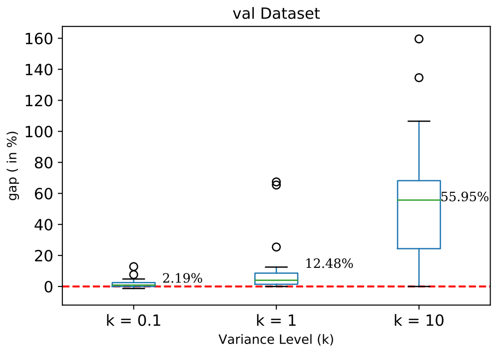
Table 5: Computational results for the *rural* data set.

| | | Deterministic TCARP | | | | Gaps (%) | | Stochastic TCARP | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Inst. | Q | OBS-D [1] | OBS-D$_t$ (s) | De Armas (2018) [2] | Keenan,(2005) LB [3] | [1-2] | [1-3] | OBS-D-S [4] | F1 Rel. [5] | F2 Rel. [6] | OBS-S-S [7] | F1 Rel. [8] | F2 Rel. [9] | OBS-S-S$_t$ (s) |
| Rural1 | 420 | 1195 | 51 | 1195 | 1195 | 0.00% | 0.00% | 1453 | 0.71 | 0.71 | 1353 | 0.81 | 0.81 | 44 |
| Rural1 | 400 | 1195 | 31 | 1195 | 1195 | 0.00% | 0.00% | 1330 | 0.89 | 0.89 | 1278 | 0.94 | 0.94 | 132 |
| Rural1 | 360 | 1195 | 142 | 1193 | 1193 | 0.17% | 0.17% | 1524 | 0.72 | 0.72 | 1307 | 0.86 | 0.86 | 165 |
| Rural2 | 420 | 1831 | 128 | 1844 | 1825 | -0.70% | 0.33% | 2218 | 0.71 | 0.72 | 2214 | 0.80 | 0.81 | 245 |
| Rural2 | 400 | 1838 | 188 | 1840 | 1821 | -0.11% | 0.93% | 2231 | 0.74 | 0.74 | 2219 | 0.80 | 0.81 | 380 |
| Rural2 | 360 | 1845 | 412 | 1836 | 1821 | 0.49% | 1.32% | 2273 | 0.75 | 0.75 | 2211 | 0.76 | 0.76 | 593 |
| Rural3 | 420 | 2198 | 321 | 2218 | 2168 | -0.90% | 1.38% | 2562 | 0.74 | 0.74 | 2532 | 0.75 | 0.75 | 353 |
| Rural3 | 400 | 2191 | 1334 | 2193 | 2156 | -0.09% | 1.62% | 2621 | 0.69 | 0.69 | 2568 | 0.71 | 0.71 | 1293 |
| Rural3 | 360 | 2182 | 653 | 2182 | 2142 | 0.00% | 3.45% | 2755 | 0.69 | 0.70 | 2629 | 0.74 | 0.74 | 728 |
| Rural4 | 420 | 2468 | 1708 | 2518 | 2373 | -1.99% | 4.00% | 2821 | 0.66 | 0.67 | 2629 | 0.74 | 0.74 | 1995 |
| Rural4 | 400 | 2487 | 1954 | 2489 | 2345 | -0.08% | 6.06% | 2805 | 0.72 | 0.73 | 2792 | 0.69 | 0.71 | 1838 |
| Rural4 | 360 | 2498 | 2738 | 2450 | 2337 | 1.96% | 6.89% | 2928 | 0.70 | 0.72 | 2903 | 0.65 | 0.66 | 1912 |
| Rural5 | 420 | 1658 | 404 | 1674 | 1647 | -0.96% | 0.67% | 1860 | 0.85 | 0.85 | 1726 | 0.95 | 0.95 | 596 |
| Rural5 | 400 | 1664 | 806 | 1655 | 1644 | 0.54% | 1.22% | 1814 | 0.84 | 0.84 | 1757 | 0.94 | 0.94 | 936 |
| Rural5 | 360 | 1656 | 968 | 1666 | 1636 | -0.60% | 1.22% | 1853 | 0.80 | 0.80 | 1892 | 0.94 | 0.94 | 1281 |
| *Average:* | | *1876* | *789* | *1877* | *1833* | *-0.15%* | *1.95%* | *2203* | *0.75* | *0.75* | *2125* | *0.81* | *0.81* | *833* |

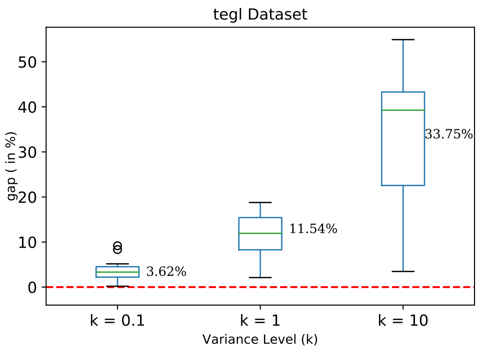
sub-optimal solutions for the stochastic version. Notice that the OBD-S-S also outperforms the OBD-S-D in terms of reliability. Hence the importance of integrating simulation methods during the searching process when dealing with stochastic optimization problems. Notice also that the OBS-D value can be seen as a reference lower bound value in a scenario with perfect information –i.e., without uncertainty– for the expected cost under stochastic conditions. Similarly, while (OBD-S-D) can be seen as an upper bound for the expected cost. For the different data sets, Figure 3 shows the percentage gap of the OBS-S-S solutions, with respect the best deterministic solution (OBS-D), using different variance levels ($c = 0.10, 1, 10$).



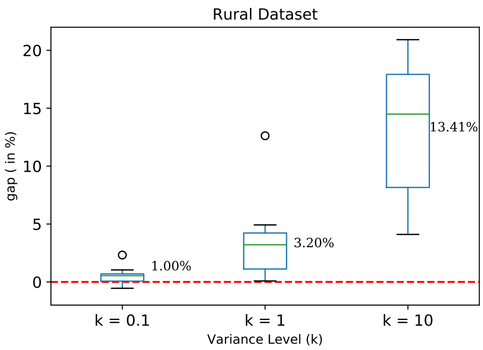

(a) *tcarp* dataset




(b) *val* dataset




(c) *tegl* dataset




(d) *rural* dataset

Figure 3: Gaps (in %) comparing the OBS-S-S solution with different level of variance w.r.t the "ideal" (uncertainty free) OBS-D solution.

All in all, the results confirm that not accounting for stochasticity during the search process may have a significant impact on the quality of the final solution. For instance, the direct application of deterministic solutions provide a slightly lower cost in ideal situations without uncertainty, but will cause a much higher cost in a scenario under uncertainty –which will be increased as uncertainty grows.

# 7 Adding Loading-based Capacity Constraints

Despite the main goal of this paper is to analyze the stochastic TCARP, our simheuristic approach can be also generalized to the case with both random time-based ($Q$) constraints and vehicle-loading constraints. The latter should not be exceeded by the sum of random demands, $L_e$, associated with each edge $e$ in a route $r$. Notice that the extension of the algorithm to cover this general case is quite straightforward: we just consider additional random variables and impose a new rule during the route-merging process, so that routes can only be merged if no time or capacity constraints are violated. We provide an example using one of the *tcarp* instances. As with the time-based capacity, we have assumed that the random demand associated with an edge $e$ follows a Log-Normal probability distribution, with $Var[L_e] = c \cdot l_e$ –as before, $c \geq 0$ is a design parameter.

Table 6 shows the obtained results for the S2 *tcarp* instance in a scenario with a medium level of uncertainty ($c = 1$). As shown in the first and second columns of the table, we have assumed a maximum time capacity of $Q = 60$, whilst the loading capacity per vehicle, $L$, varies between 40 and 10. The first row of the table shows the results obtained when only the time capacity is considered. The remaining rows offer the best-found solutions for a specific value of loading capacity. As can be observed, for a loading capacity between 40 and 30, the deterministic cost of the solutions and the number of routes do not vary with respect to the solution in which no loading capacity was considered. This is due to the fact that loading capacities are not violated for any route in the deterministic scenario. Regarding the expected cost, columns OBS-D-S and OBS-S-S show that, for $L = 30$, it increases with respect to the previous executions. This is due to the existence of route failures associated with the loading capacity. As the value of $L$ decreases, the number of routes increases (due to the smaller loading capacity of each vehicle). As a consequence, both the deterministic and the expected costs increase as well.

Figures 4 and 5 depict, respectively, the best-found solutions when the loading capacity is not considered and when it is established to $L = 20$. While in the former case, we employ just 3 routes, in the latter we employ a total of 5 routes –i.e., more routes, although smaller than the previous ones, are necessary to cover all demands.

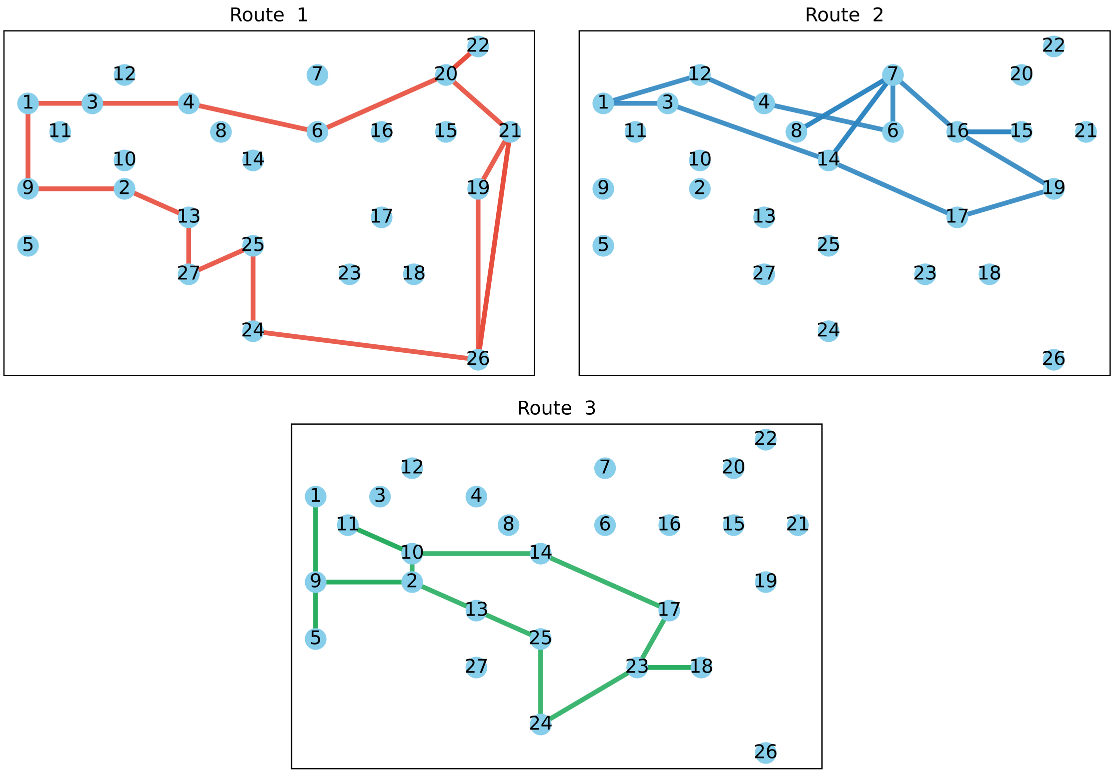



Figure 4: Best-found solution for $Q = 60$.

Table 6: Computational results for the S2 *tcarp* instance considering both time and volume capacities.

| | | | Deterministic TCARP | | | | | | | Stochastic TCARP | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Inst. | Q | L | OBS-D [1] | Num. Routes [2] | Avg. Route Cost [3] | OBS-D$_t$ (s) | OBS-D-S [4] | F1 Rel. [5] | F2 Rel. [6] | OBS-S-S [7] | Avg. Route Cost [8] | F1 Rel. [9] | F2 Rel. [10] | OBS-S$_t$ (s) |
| S1 | 60 | - | 156 | 3 | 52.00 | 4 | 195.00 | 0.84 | 0.86 | 169.00 | 56.33 | 0.94 | 0.95 | 5 |
| S1 | 60 | 40 | 156 | 3 | 52.00 | 4 | 195.00 | 0.84 | 0.86 | 169.00 | 56.33 | 0.94 | 0.95 | 5 |
| S1 | 60 | 30 | 156 | 3 | 52.00 | 4 | 257.68 | 0.57 | 0.57 | 239.01 | 79.67 | 0.75 | 0.74 | 8 |
| S1 | 60 | 20 | 166 | 5 | 33.20 | 12 | 329.06 | 0.64 | 0.64 | 268.29 | 53.66 | 0.72 | 0.71 | 32 |
| S1 | 60 | 15 | 178 | 6 | 29.67 | 11 | 386.09 | 0.57 | 0.56 | 335.69 | 55.78 | 0.73 | 0.73 | 14 |
| S1 | 60 | 10 | 205 | 9 | 22.78 | 9 | 505.41 | 0.58 | 0.57 | 456.88 | 75.98 | 0.68 | 0.68 | 18 |

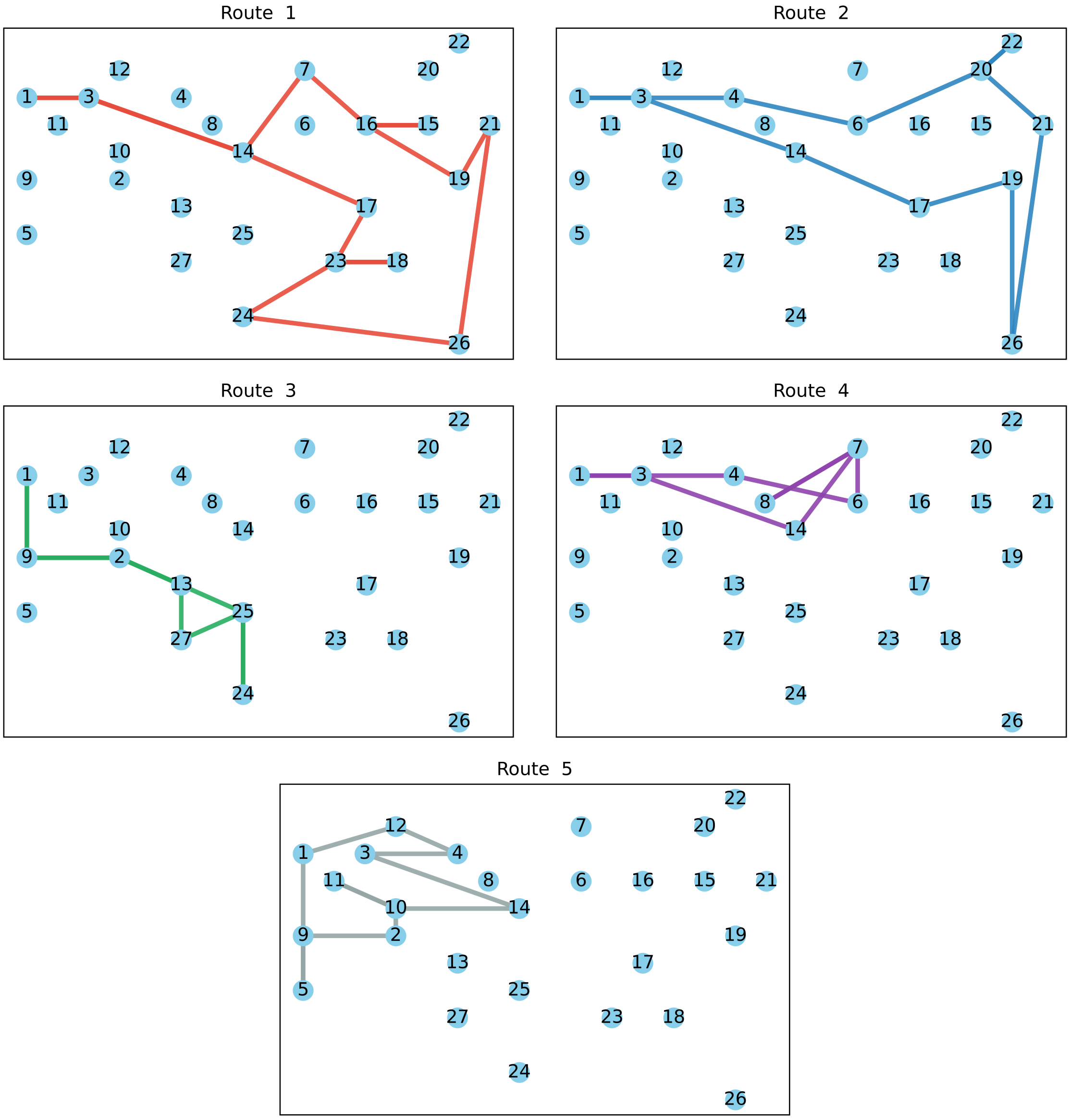



Figure 5: Best-found solution for $Q = 60$ and $L = 20$.

# 8 Conclusions and Future Work

In the traditional Capacitated Arc Routing Problem, the main constraint is usually given by the loading capacity of the vehicles. However, there are many real-life applications in which the main constraint is not the loading capacity of the vehicle but the maximum time available to complete each route. This is the challenge considered in the Time Capacitated Arc Routing Problem (TCARP). Some examples of these applications can be found in the use of unmanned aerial vehicles or drones (Luo et al., 2019), as well as in the utilization of road electric vehicles in smart cities (Fernández et al., 2019). In effect, both types of vehicles make use of electric batteries, which in practice limit their driving range capabilities. A similar situation occurs when considering the maximum number of hours that a driver can operate per day, which is limited by both legal and physical thresholds. Despite all these applications, the TCARP has been rarely studied in the scientific literature. Even more, the few articles that analyze the TCARP consider just its deterministic version. This paper goes one step beyond the state of the art by considering the stochastic version of the problem –i.e., a more realistic scenario in which travel and servicing times are modeled as random variables following a given probability distribution. The analysis of such stochastic scenarios is as challenging as critical, since aerial drones or road electric vehicles running out of battery might constitute a serious issue in terms of traffic safety and security.

In order to deal with the aforementioned challenge, this paper introduces a simheuristic algorithm, which combines a metaheuristic component with a simulation one. The metaheuristic component relies on a constructive heuristic, which is an adapted and extended version of a well-tested heuristic for the traditional CARP. The metaheuristic is also based in a well-tested strategic oscillation framework. The simulation component allows not only to test the promising solutions provided by the metaheuristic module, but it also provides feedback that guides the search for more 'reliable' routing plans –i.e., plans that can be executed, without failures, in a scenario under uncertainty.

A complete set of experiments is provided. These show that our approach is highly efficient in solving the deterministic version of the TCARP and, which is even more important, is able to effectively deal with the stochastic version. Our experiments also show an important conclusion, i.e.: that optimal (or near-optimal) routing plans for deterministic scenarios will easily become sub-optimal routing plans when executed in more realistic stochastic scenarios. This enhances the relevance of simulation-optimization approaches as the one presented here. Finally, we also show that our proposed simheuristic could be easily extended to include both time-based as well as volume-based capacity constraints.

As future work, the following lines are being considered: *(i)* to extend our approach so it considers the existence of correlations among travel and servicing times –e.g., when it rains, all travel times tend to be larger than expected under better weather conditions; *(ii)* to consider dynamic scenarios in which inputs, such as travel and servicing times or even required edges, might change over time, meaning that we have to face an optimization problem with non-static inputs (Arnau et al., 2018); and *(iii)* to consider TCARP versions, related to aerial drones and self-driving vehicles, which might need to be solved in real time and re-optimized every few minutes, thus requiring an 'agile' optimization approach (Martins et al., 2020).

## Acknowledgments

This work has been partially supported by the Spanish Ministry of Science (PID2019-111100RB-C21/AEI/ 10.13039/501100011033, RED2018-102642-T, PGC2018-0953322-B-C21/ MCIU/ AEI/ FEDER-UE). The authors are also grateful to the Michael Smurfit Graduate Business School at University College Dublin for supporting research stays that contributed to the development of this work.

## References

Arnau, Q., Juan, A.A., Serra, I., 2018. On the use of learnheuristics in vehicle routing optimization problems with dynamic inputs. Algorithms 11, 208.

Bartolini, E., Cordeau, J.F., Laporte, G., 2013. An exact algorithm for the capacitated arc routing problem with deadheading demand. Operations Research 61, 315–327.

Batista, G.V., Scarpin, C.T., Pécora, J.E., Ruiz, A., 2019. A new ant colony optimization algorithm to solve the periodic capacitated arc routing problem with continuous moves. Mathematical Problems in Engineering 2019.

Bayliss, C., Juan, A.A., Currie, C.S., Panadero, J., 2020. A learnheuristic approach for the team orienteering problem with aerial drone motion constraints. Applied Soft Computing , 106280.

Belenguer, J.M., Benavent, E., 2003. A cutting plane algorithm for the capacitated arc routing problem. Computers & Operations Research 30, 705–728.

Bertsimas, D., Howell, L.H., 1993. Further results on the probabilistic traveling salesman problem. European Journal of Operational Research 65, 68–95.

Chen, S., Golden, B., Wong, R., Zhong, H., 2009. Arc-routing models for small-package local routing. Transportation Science 43, 43–55.

Christiansen, C.H., Lysgaard, J., 2007. A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands. Operations Research Letters 35, 773–781.

Christiansen, C.H., Lysgaard, J., WøHlk, S., 2009. A branch-and-price algorithm for the capacitated arc routing problem with stochastic demands. Operations Research Letters 37, 392–398.

Corberán, A., Laporte, G., 2014. Arc routing: problems, methods, and applications. Society for Industrial and Applied Mathematics.

Corberán, A., Plana, I., Reula, M., Sanchis, J.M., 2019. A matheuristic for the distance-constrained close-enough arc routing problem. Top 27, 312–326.

Cortinhal, M.J., Mourão, M.C., Nunes, A.C., 2016. Local search heuristics for sectoring routing in a household waste collection context. European Journal of Operational Research 255, 68–79.

De Armas, J., Keenan, P., Juan, A.A., McGarraghy, S., 2019. Solving large-scale time capacitated arc routing problems: from real-time heuristics to metaheuristics. Annals of Operations Research 273, 135–162.

Dror, M., 2012. Arc routing: theory, solutions and applications. Springer Science & Business Media.

Faulin, J., Gilibert, M., Juan, A.A., Vilajosana, X., Ruiz, R., 2008. SR-1: A simulation-based algorithm for the capacitated vehicle routing problem, in: Proceedings of the Winter Simulation Conference, IEEE. pp. 2708–2716.

Fernández, E., Leitner, M., Ljubic, I., Ruthmair, M., 2019. Arc routing with electric vehicles.

Ferone, D., Gruler, A., Festa, P., Juan, A.A., 2019. Enhancing and extending the classical GRASP framework with biased randomisation and simulation. Journal of the Operational Research Society 70, 1362–1375.

Fleury, G., Lacomme, P., Prins, C., Ramdane-Cherif, W., 2005. Improving robustness of solutions to arc routing problems. Journal of the Operational Research Society 56, 526–538.

Fleury, G., Prins, C., Lacomme, P., Chérif, W.R., 2002. Robustness evaluation of solutions for the capacitated arc routing problem.

Glover, F., Laguna, M., 1998. Tabu search, in: Handbook of combinatorial optimization. Springer, pp. 2093–2229.

Gonzalez-Martin, S., Juan, A.A., Riera, D., Castellà, Q., Muñoz, R., Pérez, A., 2012. Development and assessment of the SHARP and RandSHARP algorithms for the arc routing problem. Artificial Intelligence Communications 25, 173–189.

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

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

Gruler, A., Panadero, J., de Armas, J., Moreno, J.A., Juan, A.A., 2020. A variable neighborhood search simheuristic for the multiperiod inventory routing problem with stochastic demands. International Transactions in Operational Research 27, 314–335.

Gruler, A., Panadero, J., de Armas, J., Pérez, J.A.M., Juan, A.A., 2018. Combining variable neighborhood search with simulation for the inventory routing problem with stochastic demands and stock-outs. Computers & Industrial Engineering 123, 278–288.

Gruler, A., Quintero-Araújo, C.L., Calvet, L., Juan, A.A., 2017b. Waste collection under uncertainty: a simheuristic based on variable neighbourhood search. European Journal of Industrial Engineering 11, 228–255.

Guimarans, D., Dominguez, O., Panadero, J., Juan, A.A., 2018. A simheuristic approach for the two-dimensional vehicle routing problem with stochastic travel times. Simulation Modelling Practice and Theory 89, 1–14.

Irnich, S., 2008. Solution of real-world postman problems. European Journal of Operational Research 190, 52 – 67.

Ismail, Z., Ramli, M.F., 2011. Implementation weather-type models of capacitated arc routing problem via heuristics. American Journal of Applied Sciences 8, 382.

Jin, X., Qin, H., Zhang, Z., Zhou, M., Wang, J., 2020. Planning of garbage collection service: An arc-routing problem with time-dependent penalty cost. IEEE Transactions on Intelligent Transportation Systems , 1–14.

Juan, A.A., Faulin, J., Ruiz, R., Barrios, B., Gilibert, M., Vilajosana, X., 2009. Using oriented random search to provide a set of alternative solutions to the capacitated vehicle routing problem, in: Operations Research and Cyber-Infrastructure. Springer, pp. 331–345.

Juan, A.A., Kelton, W.D., Currie, C.S., Faulin, J., 2018. Simheuristics applications: dealing with uncertainty in logistics, transportation, and other supply chain areas, in: Proceedings of the Winter Simulation Conference, IEEE. pp. 3048–3059.

Keenan, P., Naughton, M., 1996. Arc routing for rural Irish networks, in: Doležal, J., Fidler, J. (Eds.), System Modelling and Optimization: Proceedings of the Seventeenth IFIP TC7 Conference on System Modelling and Optimization, 1995. Springer US, Boston, MA, pp. 599–606.

Keenan, P.B., 2001. Spatial decision support systems for large arc routing problems. Ph.D. thesis. Faculty of Commerce. University College Dublin, Dublin, Ireland.

Keenan, P.B., 2005a. Lower bounds for the time capacitated arc routing problem. Technical Report. UCD Business School, University College Dublin. URL: http://mis.ucd.ie/Members/pkeenan/Working%20papers/TCARPLB.pdf.

Keenan, P.B., 2005b. TCARP datasets 1-10. URL: http://doi.org/10.13140/RG.2.1.2066.6485.

Keenan, P.B., 2017. TCARP large rural datasets. URL: http://doi.org/10.13140/RG.2.2.23723.75043.

Kirlik, G., Sipahioglu, A., 2012. Capacitated arc routing problem with deadheading demands. Computers & Operations Research 39, 2380–2394.

Lacomme, P., Prins, C., Ramdane-Chérif, W., 2001. A genetic algorithm for the capacitated arc routing problem and its extensions, in: Workshops on Applications of Evolutionary Computation, Springer. pp. 473–483.

Lacomme, P., Prins, C., Ramdane-Cherif, W., 2004. Competitive memetic algorithms for arc routing problems. Annals of Operations Research 131, 159–185.

Laporte, G., Musmanno, R., Vocaturo, F., 2010. An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. Transportation Science 44, 125–135.

Li, L.Y.O., Eglese, R.W., 1996. An interactive algorithm for vehicle routeing for winter - gritting. The Journal of the Operational Research Society 47, 217–228.

Luo, H., Zhang, P., Wang, J., Wang, G., Meng, F., 2019. Traffic patrolling routing problem with drones in an urban road system. Sensors 19, 5164.

Martins, L., Hirsch, P., Juan, A.A., 2020. Agile optimization of a two-echelon vehicle routing problem with pick-up and delivery. International Transactions in Operational Research .

Marzolf, F., Trépanier, M., Langevin, A., 2006. Road network monitoring: algorithms and a case study. Computers & Operations Research 33, 3494–3507.

Muyldermans, L., Pang, G., 2014. Variants of the capacitated arc routing problem, in: Corberán, A., Laporte, G. (Eds.), Arc routing: problems, methods, and applications. Society for Industrial and Applied Mathematics, pp. 223–253.

Pagès-Bernaus, A., Ramalhinho, H., Juan, A.A., Calvet, L., 2019. Designing e-commerce supply chains: a stochastic facility–location approach. International Transactions in Operational Research 26, 507–528.

Pallottino, S., 1984. Shortest-path methods: Complexity, interrelations and new propositions. Networks 14, 257–267.

Panadero, J., Doering, J., Kizys, R., Juan, A.A., Fito, A., 2020. A variable neighborhood search simheuristic for project portfolio selection under uncertainty. Journal of Heuristics 26, 353–375.

Rabe, M., Deininger, M., Juan, A.A., 2020. Speeding up computational times in simheuristics combining genetic algorithms with discrete-event simulation. Simulation Modelling Practice and Theory , 102089.

Ruiz, R., Stützle, T., 2007. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. European Journal of Operational Research 177, 2033–2049.

Tirkolaee, E.B., Mahdavi, I., Mehdi Seyyed Esfahani, M., 2018. A robust periodic capacitated arc routing problem for urban waste collection considering drivers and crew's working time. Waste Management 76, 138–146.

Willemse, E.J., Joubert, J.W., 2019. Efficient local search strategies for the mixed capacitated arc routing problems under time restrictions with intermediate facilities. Computers & Operations Research 105, 203–225.

Wøhlk, S., 2008. A decade of capacitated arc routing, in: Golden, B., Raghavan, S., Wasil, E. (Eds.), The Vehicle Routing Problem: Latest Advances and New Challenges. Springer US, pp. 29–48.

Wøhlk, S., Laporte, G., 2018. A fast heuristic for large-scale capacitated arc routing problems. Journal of the Operational Research Society 69, 1877–1887.