The Static and Stochastic VRP with Time Windows and both random Customers and Reveal Times

Michael Saint-Guillain, Christine Solnon, Yves Deville

The Static and Stochastic VRP with Time Windows and both random Customers and Reveal Times

Michael Saint-Guillain[1,2], Christine Solnon[2], Yves Deville[1]

[1]ICTEAM, Université catholique de Louvain, Belgium
[2]LIRIS, Institut National des Sciences Appliquées de Lyon, France

Abstract. Static and stochastic vehicle routing problems (SS-VRP) aim at modeling and solving real life problems by considering uncertainty on the data. In particular, customer data may not be known with certainty. Before the beginning of the day, probability distributions on customer data are used to compute a first-stage solution that optimizes an expected cost. Customer data are revealed online, while the solution is executed, and a recourse strategy is applied on the first-stage solution to quickly adapt it. Existing SS-VRP variants usually make a strong assumption on the time at which a stochastic customer reveals its data (*e.g.*, when a vehicle arrives at the corresponding location). We introduce a new SS-VRP where customer reveal times are stochastic. We define first-stage solutions and a recourse strategy for this new problem. A key point is to introduce waiting locations that are used in the first stage-solution to wait for the realization of customer stochastic data. We show how to compute the expected cost of a first-stage solution in pseudo polynomial time, in the particular case where the vehicles are not constrained by a maximal capacity. We also introduce a local search-based approach for optimizing the first-stage solution, and introduce a *scale* parameter to tune the precision and cost of the expected cost computation. Experimental results on small to large instances demonstrate its efficiency and flexibility.

1 Introduction

The Vehicle Routing Problem (VRP) aims at modeling and solving a real life common operational problem, in which a set of customers must be visited using a fleet of vehicles. Each customer comes with a certain demand. In the VRP with Time Windows (VRPTW), each customer must be visited within a given time window. A feasible solution of the VRPTW is a set of vehicle routes, such that every customer is visited exactly once during its time window and that sum of the demands along each route does not exceed the corresponding vehicle's capacity. The objective is then to find an optimal feasible solution, where optimality is usually defined in terms of travel distances.

The classical deterministic VRP(TW) assumes that customer data are known with certainty before the computation of the solution. Contrary to standard academic formulations, real world applications usually have missing part of the problem data when computing a solution. For instance, only a subset of the customer demands may be known before online execution. Missing demands hence arrive in a dynamic fashion, while vehicles are on their route. In such a context, a solution should contain operational decisions that deal with current known demands, but should also anticipate potential unknown demands. Albeit uncertainty may be considered for various attributes of the VRP

(*e.g.*, travel times), we focus on situations where the customer data are unknown a priori, and we assume that we have some probabilistic knowledge on missing data (*e.g.*, probability distributions computed from historical data). This probabilistic knowledge is used to compute a first-stage solution which is adapted online when random variables are realized. Two different kinds of adaptations may be considered: *Dynamic and Stochastic VRPTW* (DS-VRPTW) and *Static and Stochastic VRPTW* (SS-VRPTW).

In the DS-VRPTW, the solution is re-optimized at each time-step, and this re-optimization involves solving an $\mathcal{NP}$-hard problem so that it is usually approximated with meta-heuristics as proposed, for example, in [1,2,3]. Note that the DS-VRPTW assumes a probabilistic knowledge on the potential requests. In contrary, in [4] for instance no prior knowledge is provided on the potential requests, which are then assumed to be uniformly distributed in the Euclidean plan.

In the SS-VRP(TW), no expensive reoptimization is allowed during online execution. When unknown information is revealed, the first stage solution is adapted online by applying a predefined recourse strategy whose time complexity is polynomial. In this case, the goal is to find a first stage solution that minimizes its total cost plus the *expected* extra cost caused by the recourse strategy. For example, in [5], the first stage solution is a set of vehicle tours which is computed offline with respect to probability distributions of customer demands. Real customer demands are revealed online, and two different recourse strategies are proposed: in the first one, each demand is assumed to be known when the vehicle arrives at the customer place, and if it is larger than or equal to the remaining capacity of the vehicle, then the first stage solution is adapted by adding a round trip to the depot to unload the vehicle; in the second recourse strategy, each demand is assumed to be known when leaving the previous customer and the recourse strategy is refined so that customers with null demands are skipped.

In this paper, we focus on the SS-VRPTW, and introduce a new variant where no strong assumption is made on the moment at which customer requests are revealed during the operations (contrary to most existing work that assume that customer requests are known either when arriving at the customer place, or when leaving the previous customer). In this new variant, called the *SS-VRPTW with random Customers and Reveal time* (SS-VRPTW-CR), the reveal times of customer requests are random variables. To handle uncertainty on reveal times, we introduce waiting locations when computing first-stage solutions: the routes computed offline visit waiting locations and a waiting time is associated with each waiting location. When a customer request is revealed, it is either accepted (if it is possible to serve it) or rejected. The recourse strategy then adapts routes so that all accepted requests are guaranteed to eventually be served. The goal is to compute the first-stage solution that minimizes the expected number of rejected requests.

Our motivating application is an on-demand health care service for elderly or disabled people. Health care services are provided directly at home by mobile medical units. Every person who's registered to the service can request a health care support at any moment of the day with the guarantee to be satisfied within a given time window. From historical data, we know, for each customer region and each time unit, the probability that a request appears. Given this stochastic knowledge, we compute a first-stage solution. When a request appears (online), the recourse strategy is used to decide

whether the request is accepted or rejected and to adapt medical unit routes. When a request is rejected, the system must rely on an external service provider in order to satisfy it. Therefore, the goal is to minimize the expected number of rejected requests.

Organization. In section 2, we review the existing studies on VRPs that imply stochastic customers. Section 3 formally defines the general SS-VRPTW-CR. Section 4 describes a recourse strategy for this problem. Section 5 shows how the expected number of rejected requests can be efficiently computed from a first stage solution and for a specific recourse strategy. Section 6 describes a local search-based approach for approximating an optimal first stage solution. Experimental results are analysed in section 7. Further research directions are finally discussed in section 8.

2 Related work

The most studied cases in SS-VRPs are stochastic customers (presence of customers are random variables), stochastic demands (quantities required by customers are random variables), and stochastic times (travel and/or service times are random variables). Since the SS-VRPTW-CR belongs to the first case, we focus this review on customers uncertainty.

The Traveling Salesman Problem (TSP) is a special case of the VRP with only one uncapacitated vehicle. [6] introduced the TSP with stochastic Customers (SS-TSP-C), and provided mathematical formulations and a number of properties and bounds. In particular, he showed that an optimal solution for the deterministic problem can be arbitrarily bad in case of uncertainty. [7] developed the first exact solution method for the SS-TSP-C, using the integer L-shaped method [8] to solve instances up to 50 customers. Heuristics for the SS-TSP-C have then been proposed (e.g. [9,10,11]) as well as meta-heuristics such as simulated annealing [12] or ant colony optimization [13].

The first SS-VRP with stochastic Customers (SS-VRP-C) has been studied by [9] as a generalization of the SS-TSP-C. [14] compared different heuristics. [5] considered a VRP with stochastic Customers and Demands (SS-VRP-CD). A customer demand is assumed to be revealed either when the vehicle leaves the previous customer or when it arrives at the customer's own location. Two different recourse strategies are proposed. For both strategies, closed-form mathematical expressions are provided to compute the expected total distance, provided a first stage solution. [15] and [16] developed the first exact algorithm for solving the SS-VRP-CD for instances up to 70 customers, by means of an integer L-shaped method. [17] later proposed a tabu search to efficiently approximate the solution. Experimentations are reported on instances with up to 46 customers.[18] later developed an adaptive memory programming metaheuristic for the SS-VRP-C and assess it on benchmarks with up to 483 customers and 38 vehicles.

Particularly close to the SS-VRPTW-CR is the SS-TSP-C with Deadlines [19]. Unlike the SS-VRPTW-CR, the set of customers is revealed at the beginning of the operations. A recent literature review on SS-TSP-C may be found in [20].

[21] considered a variant of the SS-VRPTW-C, the Courier Delivery Problem with Uncertainty. Unlike the SS-VRPTW-CR, authors assume that customer presences are not revealed at some random moment during the operations, but all at once at the beginning of the day (that is, after computing the first stage solution).

3 Description of the SS-VRPTW-CR

Input data. We consider a complete directed graph $G = (V, A)$ and a discrete time horizon $H = [1, h]$, where interval $[a, b]$ denotes the set of all integer values i such that $a \leq i \leq b$. To every arc $(i, j) \in A$ is associated a travel time (or distance) $d_{i,j} \in \mathbb{N}$. The set of vertices $V = \{0\} \cup W \cup C$ is composed of a depot 0, a set of m waiting locations $W = [1, m]$ and a set of n customer regions $C = [m + 1, m + n]$. We note $W_0 = W \cup \{0\}$ and $C_0 = C \cup \{0\}$. The fleet is composed of K uncapacitated vehicles.

We consider the set $R = C \times H$ of potential customer requests such that an element $(c, \Gamma) \in R$ represents a potential request revealed at time $\Gamma \in H$ for customer region c. To each potential request $r = (c, \Gamma) \in R$ is associated a deterministic demand $q_r \in [1, Q]$, a deterministic service duration $s_r \in H$ and deterministic time window $[e_r, l_r]$ with $\Gamma \leq e_r \leq l_r \leq h$. We note p_r the probability that r appears on vertex c at time Γ, and assume independence between request probabilities. Although our formalism imposes $\Gamma \geq 1$ for all potential requests, in practice a request may be known with certainty that is, with probability 1.

To simplify notations, a request $r = (c, \Gamma)$ can be written in place of its own region c. For instance, the distance $d_{v,c}$ can also be intuitively written $d_{v,r}$. Furthermore, we use Γ_r to denote the reveal time of a request $r \in R$ and c_r for its customer region.

First stage solution. The first-stage solution is computed offline, before the beginning of the time horizon. It consists in a set of K vehicle routes visiting a subset of the m waiting vertices, together with time variables denoted τ indicating how long a vehicle should wait on each vertex. More specifically, we denote (x, τ) a first stage solution to the SS-VRPTW-CR, where:

- $x = \{x_1, ..., x_K\}$ defines a set of K sequences of waiting vertices of W, such that each sequence x_k starts and ends with 0 and each vertex of W occurs at most once in x. We note $W^x \subseteq W$ the set of waiting vertices visited in x.
- $\tau : W^x \to H$ associates a waiting time $\tau_w \geq 1$ to every waiting vertex $w \in W^x$;
- Each sequence $\langle 0, w_1, ..., w_{m'}, 0\rangle$ in x is such that the vehicle is back to the depot before the end of the day.

In other words, x defines a *Tour Orienteering Problem* (TOP, see [22]) to which each visited location is assigned a waiting time by τ. Given a first stage solution (x, τ), we define $on(w) = [\underline{on}(w), \overline{on}(w)]$ for each vertex $w \in W^x$ such that $\underline{on}(w)$ (resp. $\overline{on}(w)$) is the arrival (resp. departure) time on w. In a sequence $\langle 0, w_1, ..., w_{m'}, 0\rangle$ in x, we then have $\underline{on}(w_i) = \overline{on}(w_{i-1}) + d_{w_{i-1},w_i}$ and $\overline{on}(w_i) = \underline{on}(w_i) + \tau_{w_i}$ for $i \geq 1$ and assume both $\overline{on}(0) = 1$ and $\underline{on}(0) = \overline{on}(w_{m'}) + d_{w_{m'},0} \leq h$. Figure 1 (left) illustrates an example of first stage solution on a basic SS-VRPTW-CR instance.

Recourse strategy and second stage solution. A recourse strategy $\mathcal{R}$ states how a second stage solution is gradually constructed as requests are dynamically revealed. In this paragraph, we define the properties of a recourse strategy. An example of recourse strategy is given in Section 4.

Let $\xi \subseteq R$ be the set of requests that reveal to appear by the end of the horizon H. The set ξ is also called a *scenario*. We note $\xi^t \subseteq \xi$ the set of requests appearing at time

$t \in H$, *i.e.*, $\xi^t = \{r \in \xi : \Gamma_r = t\}$. We note $\xi^{1..t} = \xi^1 \cup ... \cup \xi^t$ the set of requests appeared up to time t.

A second stage solution is incrementally constructed at each time unit by following the skeleton provided by the first stage solution (x, τ). At a given time t of the horizon, we note (x^t, A^t) the current state of the second stage solution:

- $\diamond$ x^t defines a set of vertex sequences describing the route operations performed up to time t. Unlike x, we define x^t on a graph that also includes the customer regions. Operations described in x^t must satisfy the time window and vehicle capacity constraints imposed by the VRPTW.
- $\diamond$ $A^t \subseteq \xi^{1..t}$ is the set of *accepted* requests up to time t. Requests of $\xi^{1..t}$ that do not belong to A^t are said to be *rejected*.

We distinguish between requests that are accepted and those that are both accepted and satisfied. Up to a time t, an accepted request is said to be *satisfied* if it is visited in x^t by a vehicle. Accepted requests that are no yet satisfied must be *guaranteed to be eventually satisfied* according to their time window.

Figure 1(right) illustrates an example of second stage solution being partially constructed at some moment of the time horizon.

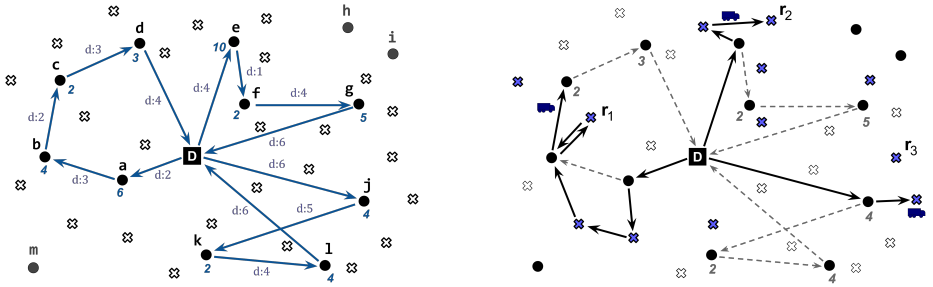


Fig. 1. On the left: first stage solution with $K = 3$ vehicles. The depot, waiting vertices and customer regions are represented by a square, circles and crosses respectively. Arrows represent vehicle routes and integers indicate waiting times at waiting locations. Values preceded by 'd' indicate travel times. Waiting vertices h, i and m are not part of the first stage solution. Here $\overline{on}(D) = 1$, $\underline{on}(a) = 3$, $\overline{on}(a) = 9$, $\underline{on}(b) = 12$, $\overline{on}(b) = 16$, etc.
On the right: partial second stage solution (plain arrows). Filled crosses are accepted requests. Some accepted requests, such as r_1, have been satisfied (or the vehicle is currently traveling towards the location, *e.g.*, r_2), while some others are not yet satisfied (*e.g.*, r_3).

Before starting the operations (time 0), x^0 is a set of K sequences that only contain vertex 0, and $A^0 = \emptyset$. At each time unit $t \in H$, given a first stage solution (x, τ), a previous state (x^{t-1}, A^{t-1}) of the second stage solution and a set ξ^t of requests appearing at time t, the new state (x^t, A^t) is obtained by applying a specific recourse strategy $\mathcal{R}$:

$$(x^t, A^t) = \mathcal{R}\big((x, \tau), (x^{t-1}, A^{t-1}), \xi^t\big). \tag{1}$$

A necessary property of a recourse strategy is to avoid reoptimization. We consider that $\mathcal{R}$ avoids reoptimization if the computation of (x^t, A^t) is achieved in polynomial time.

We note cost$(\mathcal{R}, x, \tau, \xi) = |\xi \setminus A^h|$ the final cost of a second stage solution with respect to a scenario ξ, given a first stage solution (x, τ) and under a recourse strategy $\mathcal{R}$. This cost is the number of requests that are rejected at the end h of the time horizon.

Optimal first stage solution. An optimal first stage solution (x, τ) to the SS-VRPTW-CR minimizes the expected cost of the second stage solution under a given strategy $\mathcal{R}$, satisfying statements (2)-(3):

$$\text{(SS-VRPTW-CR)} \quad \underset{x,\tau}{\text{Minimize}} \;\; \mathcal{Q}^{\mathcal{R}}(x, \tau) \tag{2}$$

$$\text{s.t.} \quad (x, \tau) \text{ is a first stage solution.} \tag{3}$$

The objective function $\mathcal{Q}^{\mathcal{R}}(x, \tau)$, which is nonlinear in general, determines the *expected number of rejected requests*, i.e. requests that fail to be visited under recourse strategy $\mathcal{R}$ and first stage solution (x, τ):

$$\mathcal{Q}^{\mathcal{R}}(x, \tau) = \sum_{\xi \subseteq R} \Pr(\xi) \text{ cost}(\mathcal{R}, x, \tau, \xi) \tag{4}$$

where $\Pr(\xi)$ defines the probability of scenario ξ. Since we assume independence between requests, we have $\Pr(\xi) = \prod_{r \in \xi} p_r \cdot \prod_{r \in R \setminus \xi} (1 - p_r)$.

4 Description of a recourse strategy

In order to avoid reoptimization, the set R of potential requests is ordered. Furthermore, given a first stage solution (x, τ) that visits the set W^x of waiting locations, each potential request $r = (c, \Gamma) \in R$ is assigned to exactly one waiting vertex (and hence, a vehicle) in W^x.

Informally, the recourse strategy accepts a new request if it is possible for the vehicle associated to its corresponding waiting vertex location to adapt its first stage tour to visit the customer. The vehicle will then travel from the waiting location to the customer and return to the waiting location. Time window constraints should be respected, and the already accepted requests should not be perturbed. In the recourse strategy we propose here, we assume the vehicles not to be constrained by a maximal capacity.

Request ordering. Before computing first-stage solutions, we order R by increasing reveal time Γ_r first, end of time window l_r second and request number r to break ties. Let $<_R$ denote this total strict order on R. Whereas the remaining of the paper is based on the assumption of total order on Γ_r, the ordering criteria may be modified without loss of generality (*e.g.*, replacing l_r by e_r), as long as the total order remains strict and primarily based on Γ_r, i.e. $\forall r_1, r_2 \in R, \Gamma_{r_1} < \Gamma_{r_2} \Rightarrow r_1 <_R r_2$.

Request assignment according to a first stage solution. Given a first-stage solution (x, τ), we assign each request of R to a waiting vertex visited in (x, τ). This assignment is computed for each first stage solution (x, τ) before the application of the recourse strategy. As an optimally fair distribution of the potential requests might be excessively expensive to compute, we propose the following heuristic.

Let $t^{\min}_{r,w} = \max\{\underline{on}(w),\ \Gamma_r,\ e_r - d_{w,r}\}$ be the minimum time for leaving waiting location w to satisfy request r. Indeed, a vehicle cannot handle r before (1) the vehicle is on w, (2) r is revealed, and (3) the beginning e_r of the time window minus the time $d_{w,r}$ needed to go from w to r.

Let $t^{\max}_{r,w} = \min\{l_r - d_{w,r},\ \overline{on}(w) - d_{w,r} - s_r - d_{r,w}\}$ be the latest time at which a vehicle can handle r (which also involves a service time s_r) from waiting location w and still leave it in time $t \leq \overline{on}(w)$.

Given a first stage solution (x, τ), we assign each request $r \in R$ either to a waiting vertex of W^x or to $\perp$ (to denote that r is not assigned). We note $w(r)$ this assignment which is computed as follows:

- ◇ Let $W^x_r = \{w \in W^x : t^{\min}_{r,w} \leq t^{\max}_{r,w}\}$ be the set of feasible waiting locations for r
- ◇ If $W^x_r = \emptyset$ then set $w(r)$ to $\perp$ (r is always rejected)
- ◇ Else set $w(r)$ to the feasible vertex of W^x_r that has the least number of requests already assigned to it (break further ties w.r.t. vertex number)

Once finished, the request assignment ends up with a partition $\{\pi_\perp, \pi_1, ..., \pi_K\}$ of R, where π_k is the set of requests assigned to the waiting vertices visited by vehicle k and $\pi_\perp$ is the set of unassigned requests (such that $\mathrm{w}(r) = \perp$). We note $\pi^w_k \subseteq \pi_k$ the set of requests assigned to $w \in W^x$ in route k. We note $\mathrm{fst}(\pi^w_k)$ the first request of π^w_k according to order $<_R$, and for each request $r \in \pi^w_k$ such that $r \neq \mathrm{fst}(\pi^w_k)$ we note $\mathrm{prv}(r)$ the request of π^w_k that immediately precedes r according to order $<_R$.

Using the recourse strategy to adapt a first stage solution at a current time t. At each time step t, the recourse strategy is applied to compute the second stage solution (x^t, A^t), given the first stage solution (x, τ), the second stage solution (x^{t-1}, A^{t-1}) at the end of time $t-1$, and the incoming requests ξ^t.

A^t is the set of accepted requests. It is initialized with A^{t-1}. Then, each incoming request of ξ^t is considered (taken by increasing order of $<_R$) and either accepted (added to A^t) or rejected (not added to A^t) by applying the following decision rule:

- ◇ Let k be the vehicle associated with r (i.e., $r \in \pi^k$)
- ◇ Let $y : R \to H$ be the function returning the time at which k finishes to satisfy all accepted requests that precede r (according to $<_R$) and reaches waiting vertex $\mathrm{w}(r)$. Namely, $y(r)$ is the time at which k is available for r and is defined by:
  - ⋆ If $r = \mathrm{fst}(\pi^w_k)$, then $y(r) = \underline{on}(w)$
  - ⋆ else if $\mathrm{prv}(r) \notin A^t$ then $y(r) = y(\mathrm{prv}(r))$
  - ⋆ else $y(r) = \max(y(\mathrm{prv}(r)) + d_{w,\mathrm{prv}(r)}, e_{\mathrm{prv}(r)}) + s_{\mathrm{prv}(r)} + d_{\mathrm{prv}(r),w}$

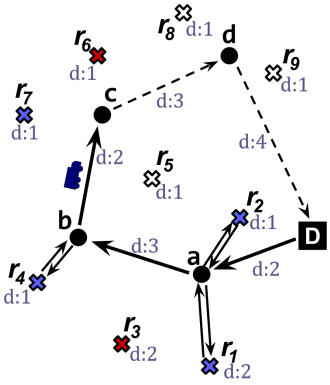
  If $y(r)$ allows k to reach r during its time window and to arrive in time to its next waiting location (*i.e.*, $y(r) \leq t^{\max}_{r,\mathrm{w}(r)}$) then r is accepted and added to A^t; otherwise it is rejected.

Once A^t has been computed, vehicle operations for time unit t must be decided. Vehicles operate independently of each other. If vehicle k is traveling between a waiting location and a customer region, or if it is serving a request, then its operation remains unchanged; Otherwise, let w be the current waiting location (or the depot) of vehicle k:

- ◇ If $t = \overline{on}(w)$, the operation for k is "travel from w to the next waiting vertex (or the depot), as defined in the first stage solution"

◇ Otherwise, let $P = \{r \in \pi_k^w | c_r \notin x^t \wedge (r \in A^t \vee t < \Gamma_r)\}$ be the set of requests of π_k^w that are not yet satisfied and that are either accepted or with unknown revelation
  ⋆ If $P = \emptyset$, then the operation for k is "travel back to the depot"
  ⋆ Otherwise, let r^{next} be the smallest element of P according to $<_R$
    → If $t < t^{\min}_{r^{\text{next}},w}$, then the operation for k is "wait until $t+1$"
    → Otherwise, the operation is "travel to r^{next}, serve it and come back to w"

Figure 2 shows an example of second stage solution at a current time $t = 17$, from an operational point of view.



r	w(r)	Γ_r	$[e_r, l_r]$	$t^{min}_{r,a}$	$t^{max}_{r,a}$	status	y(r)	sat(r)
a	Arrival time : 3			Departure time : 9				
1	a	2	2 , 6	3	4	satisf.	3	5
2	a	2	2 , 8	3	7	satisf.	7	8
3	a	5	5 , 15	5	5	rejected	9	n.a.
b	Arrival time : 12			Departure time : 16				
4	b	10	13, 16	13	13	satisf.	12	14
5	b	13	13 , 17	13	14	absent !	15	n.a.
c	Arrival time : 18			Departure time : 20				
6	c	16	16 , 20	18	18	accepted	18	19
7	c	17	17 , 19	18	18	rejected	20	n.a.
d	Arrival time : 23			Departure time : 26				
8	d	18	18 , 25	23	24	unknown	23	?
9	d	20	20 , 30	23	24	unknown	?	?

current time: t = 17

Fig. 2. Example of second stage solution at time $t = 17$, under strategy $\mathcal{R}1$. A filled cross represents a request that appeared, an empty one a request that is either still unknown (*e.g.*, r_8) or revealed as being absent (that is didn't appear, *e.g.*, r_5). Here $\pi_k = \langle r_a, r_1, \ldots, r_9 \rangle$ is the sequence of requests assigned to the vehicle, according to (x, τ). We assume $q_r = s_r = 0, \forall r \in R$. $sat(r)$ represents, for a request r, the time at which r gets satisfied.

5 Expected cost of second stage solutions

Provided a recourse strategy $\mathcal{R}$ and a first stage solution (x, τ) to the SS-VRPTW-CR, a naive approach for computing $\mathcal{Q}^{\mathcal{R}}(x, \tau)$ would be to literally follow equation (4), therefore using the strategy described by $\mathcal{R}$ in order to confront (x, τ) to each and every possible scenario $\xi \subseteq R$. Because there is an exponential number of scenarios with respect to $|R|$, this naive approach is not affordable in practice. In this section, we show how the expected number of rejected requests $\mathcal{Q}^{\mathcal{R}}(x, \tau)$ under the recourse strategy described in Section 4 may be computed in $\mathcal{O}(nh^2)$ using closed form expressions, in the special case where vehicles are of infinite capacity.

We assume that the potential request probabilities are independent from each other such that, for any couple of requests $r, r' \in R$, the probability $p_{r \cap r'}$ that both requests appear is given by $p_{r \cap r'} = p_r \cdot p_{r'}$.

Expected cost. $\mathcal{Q}^{\mathcal{R}}(x, \tau)$ is equal to the expected number of rejected requests, which in turn is equal to the expected number of requests that reveal to appear minus the

expected number of accepted requests. The expected number of revealed requests is given by the sum of all request probabilities, whereas the expected number of accepted requests is equal to the sum, for every request r, of the probability that it belongs to A^h:

$$\mathcal{Q}^{\mathcal{R}}(x,\tau) = \sum_{r\in R} p_r - \sum_{r\in R} \Pr\{r \in A^h\} = \sum_{r\in R} \left(p_r - \Pr\{r \in A^h\}\right) \tag{5}$$

where the right-hand side of the equation comes from the independence hypothesis.

If we consider a request $r \in \pi_k$, the probability $\Pr\{r \in A^h\}$ only depends on the time at which vehicle k is available for r, which itself depends on previous operations. Recall the $y : R \to H$ function described in section 4: $y(r)$ is that time. Whereas $y(r)$ is deterministic for a specific scenario, it is not anymore in the context of the computation of $\Pr\{r \in A^h\}$ and we are thus interested in its probability distribution. More specifically, we compute the probability that, at a time $t \in H$, a request r already appeared and the vehicle leaves $\mathrm{w}(r)$ to satisfy it. Let's call this probability $g_1(r,t)$:

$$g_1(r,t) \equiv \Pr\{\text{request } r \text{ appeared at time } t' \leq t \text{ and } \mathrm{departureTime}(r) = t\}$$

where $\mathrm{departureTime}(r)$ is the time at which the vehicle leaves vertex $\mathrm{w}(r)$ in order to serve r, if r has been accepted. According to the recourse strategy, for a specific scenario we see that $\mathrm{departureTime}(r) = \max(y(r), t^{\min}_{r,w(r)})$.

The probability $\Pr\{r \in A^h\}$ that a request r gets satisfied is the probability that both r appears and that $\mathrm{departureTime}(r) \leq t^{\max}_{r,w}$, that is:

$$\Pr\{r \in A^h\} = \sum_{t=1}^{t^{\max}_{r,w}} g_1(r,t) = \sum_{t=t^{\min}_{r,w}}^{t^{\max}_{r,w}} g_1(r,t). \tag{6}$$

The calculus of $g_1(r,t)$ is less obvious. Since $\mathrm{departureTime}(r)$ depends on previous operations on the same waiting location $w = \mathrm{w}(r)$, we calculate $g_1(r,t)$ recursively starting from the first request $r_1 = \mathrm{fst}(\pi^w_k)$ assigned to the waiting location, up to the current request r. The second stage solution strictly respects the first stage schedule when visiting the waiting vertices, that is, these are guaranteed to be visited according to their arrival ($\underline{on}$) and departure ($\overline{on}$) times. The base case is then:

$$g_1(r_1,t) = \begin{cases} p_{r_1}, & \text{if } \ t = \max(\underline{on}(w), t^{\min}_{r_1,w}) \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Indeed, if r_1 appeared then the vehicle leaves w at time $t^{\min}_{r_1,w}$, unless it has not yet reached w at that time. The general case of a request $r >_R r_1$, $r \in \pi^w_k$, depends on the time at which the vehicle gets rid of the preceding request $\mathrm{prv}(r)$. Let $f(r,t)$ be the probability that, at time t, the vehicle either reaches back w after having served r, or discard r because it is not satisfiable or because it has revealed not to appear. It represents the time at which the vehicle becomes available for the next request after r in π^w_k, if any (computation of f is detailed below). We define $g_1(r,t)$ based on f-

probabilities of the previous request $\mathrm{prv}(r)$:

$$g_1(r,t) = \begin{cases} p_r \cdot f(\mathrm{prv}(r), t) & \text{if } \ t > t^{\min}_{r,w} \\ p_r \cdot \sum_{t'=\underline{on}(w)}^{t^{\min}_{r,w}} f(\mathrm{prv}(r), t') & \text{if } \ t = t^{\min}_{r,w} \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Indeed, if $t > t^{\min}_{r,w}$ the vehicle leaves w to serve r as soon as it gets rid of the previous one $\mathrm{prv}(r)$. In such case, $g_1(r,t)$ is the probability that both r has already appeared and the vehicle is available for it at time t, that is, finished with request $\mathrm{prv}(r)$ at time t. At any time below $t^{\min}_{r,w}$, the probability that the vehicle leaves w must obviously be zero, since $t^{\min}_{r,w}$ is the minimum time for serving r from location $w = \mathrm{w}(r)$. At time $t = t^{\min}_{r,w}$, we must consider the possibility that the vehicle was waiting for being able to serve r, but from an earlier time $t' < t^{\min}_{r}$. The overall probability that the vehicle leaves w for request r at time $t = t^{\min}_{r,w}$ is then p_r times all the f-probabilities that the vehicle was actually available from a time $\underline{on}(w) \leq t' \leq t^{\min}_{r,w}$.

The f-probabilities of a request r depend on what exactly happened to r. Namely, from a time t there are two cases: either r consumed operational time, or it didn't at all:

$$f(r,t) = \ g_1(r, t - S_r) \cdot \delta^w(r, t - S_r) + \ g_1(r,t) \cdot \big(1 - \delta^w(r,t)\big) \ + \ g_2(r,t). \tag{9}$$

where $S_r = d_{w,r} + s_r + d_{r,w}$ and the function $\delta^w(r,t)$ returns 1 iff request r is satisfiable from time t and vertex w, *i.e.*, $\delta^w(r,t) = 1$ if $t \leq t^{\max}_{r,w}$, and $\delta^w(r,t) = 0$ otherwise.

The first term in the summation of the right hand side of equation (9) gives the probability that request r actually appeared and got satisfied. In such a case, $\mathrm{departureTime}(r)$ must be the current time t, minus delay S_r needed for serving r.

The second and third terms of equation (9) add the probability that the vehicle was available time t, but that request r did not consume any operational time. There are only two possible reasons for that: either r actually appeared but was not satisfiable (second term), or r did not appear at all (third term), where $g_2(r,t)$ is the probability that r did not appear and is discarded at time t, and is computed as follows. For the base case of first potential request $r_1 = \mathrm{fst}(\pi^w_k)$, we have:

$$g_2(r_1,t) = \begin{cases} 1 - p_{r_1} & \text{if } \ t = \max(\underline{on}(w), \Gamma_{r_1}) \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

The general case for $r \geq r_1$, $r \in \pi^w_k$, is quite similar to the one of function g_1. We just consider the probability $1 - p_r$ that r doesn't reveal and replace $t^{\min}_{r,w}$ by Γ_r:

$$g_2(r,t) = \begin{cases} (1-p_r) \cdot f(\mathrm{prv}(r), t) & \text{if } \ t > \max(\underline{on}(w), \Gamma_r) \\ (1-p_r) \cdot \sum_{t'=\underline{on}(w)}^{\max(\underline{on}(w),\Gamma_r)} f(\mathrm{prv}(r), t') & \text{if } \ t = \max(\underline{on}(w), \Gamma_r) \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

A note on implementation. Since we are interested in computing $\Pr\{r \in A^h\}$ for each request r separately, by following the definition of g_1, we only require the f-probability associated to $\mathrm{prv}(r)$ to be already computed. This suggests a dynamic programming

Algorithm 1: Local search to compute a first stage solution of SS-VRPTW-CR

1 Let (x, τ) be an initial feasible first stage solution.
2 Initialize the neighborhood operator *op* to 1
3 **while** *some stopping criterion is not met* **do**
4 Select a solution (x', τ') at random in $\mathcal{N}_{op}(x, \tau)$
5 **if** *some acceptance criterion is met on* (x', τ') **then** set (x, τ) to (x', τ') and *op* to 1 ;
6 **else** change the neighborhood operator *op* to $op \% n_{op} + 1$;
7 **return** the best first stage solution computed during the search

approach. Computing all the f-probabilities can then be incrementally achieved while filling up a 2-dimensional matrix containing all the f-probabilities.

Computational complexity. Complexity of computing the expected cost is equivalent to the one of filling up a $|\pi_k^w| \times h$ matrix for each visited waiting location $w \in W^x$, in order to store all the $f(r, t)$ probabilities. By processing incrementally on each waiting location separately, each matrix cell can be computed in constant time using equation (9). In particular, once the probabilities in cells $(\mathrm{prv}(r), 1 \cdots t)$ are known, the cell (r, t) such that $r \neq \mathrm{fst}(\pi_k^w)$ can be computed in $\mathcal{O}(1)$ according to equations (8) - (11). Given n customer regions and a time horizon of length h, we have at most $|R| = nh \geq \sum_{w \in W^x} |\pi_k^w|$ potential requests. It then requires at most $\mathcal{O}(|R|h) = \mathcal{O}(nh^2)$ constant time operations to compute $\mathcal{Q}^{\mathcal{R}}(x, \tau)$.

6 Local Search for the SS-VRPTW-CR

Algorithm 1 describes a Simulated Annealing [23] local search approach for approximating the optimal first stage solution (x, τ), minimizing $\mathcal{Q}^{\mathcal{R}}(x, \tau)$. The computation of $\mathcal{Q}^{\mathcal{R}}(x, \tau)$ is performed according to equations of section 5 and is considered from now as a black box. Starting from an initial feasible first stage solution (x, τ), Algorithm 1 iteratively modifies it by using a set of $n_{op} = 9$ neighborhood operators. At each iteration, it randomly chooses a solution (x', τ') in the current neighborhood (line 4), and either accepts it and resets the neighborhood operator *op* to the first one (line 5), or rejects it and changes the neighborhood operator *op* to the next one (line 6). At the end, the algorithm simply returns the best solution (x^*, τ^*) encountered so far.

Initial solution and stopping criterion. The initial first stage solution is constructed by randomly adding each waiting vertex in a route $k \in [1, K]$. All waiting vertices are thus initially part of the solution. The stopping criterion depends on the computational time dedicated to the algorithm.

Neighborhood operators. We consider 4 wellknown operators for the VRP: relocate, swap, inverted 2-opt, and cross-exchange (see [24,25] for detailed description). In addition, 5 new operators are dedicated to waiting vertices: 2 for either inserting or removing from W^x a waiting vertex w picked at random, 2 for increasing or decreasing the waiting time τ_w at random vertex $\mathrm{w} \in W^x$, and 1 that transfers a random amount of waiting time units from one waiting vertex to another.

Acceptance criterion. We use a Simulated Annealing acceptance criterion. Improving solutions are always accepted, while degrading solutions are accepted with a probability that depends on the degradation and on a temperature parameter, *i.e.*, the probability of accepting (x', τ') is $e^{-\frac{1-\mathcal{Q}^{\mathcal{R}}(x,\tau)/\mathcal{Q}^{\mathcal{R}}(x',\tau')}{T}}$. The temperature T is updated by a *cooling factor* $0 < \alpha < 1$ at each iteration of Algorithm 1: $T \leftarrow \alpha \cdot T$. During the search process, T gradually evolves from an initial temperature T_{init} to nearly zero. A restart strategy is implemented by resetting the temperature to $T \leftarrow T_{\text{init}}$ each time T decreases below a fixed limit T_{min}.

7 Experimentations

Test instances. We have randomly generated instances for the SS-VRPTW-CR. Each test instance is drawn in a square of $[100, 100]$ distance units, and is characterized by:

- ◇ The number $|C| \in \{30, 50, 80\}$ of *customer regions*, randomly distributed in the square. Each customer $c \in C$ region hosts n_{TS} potential requests.
- ◇ The number $|W| \in \{20, 30, 50\}$ of *waiting vertices*, randomly distributed in the square.
- ◇ The size $h = 480$ of the *horizon* (corresponding to the number of minutes in an 8 hour day).
- ◇ The number $n_{\text{TS}} = 24$ of *time slots*. Time slots are introduced because it is not realistic to detail request probabilities for each time unit of the horizon (*i.e.*, every minute). We set n_{TS} to 24 so that the probability that a request appears at a customer region is specified for 20 minute time slots.
- ◇ The number $K \in \{1, 3, 5, 10\}$ of available vehicles.

Travel times between vertices correspond to Euclidean distances, divided by a velocity parameter specified in the instance. Figure 3 shows an example of test instance. As a convention, the first time slot is associated to time unit 1 whereas time slot i is associated to time unit $1 + (i-1) \cdot \lfloor h/n_{\text{TS}} \rfloor$. In our instance *a.1*, a potential request r associated to time slot 2 has a reveal time $\Gamma_r = 21$ and no potential request is associated to time units $[2, 20]$, $[22, 40]$, etc. All the test instances are available at `http://becool.info.ucl.ac.be/resources/benchmarks-ss-vrptw-cr`.

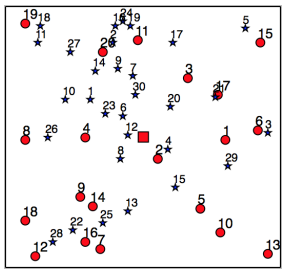


Fig. 3. Map representation of instance $a.1$. The depot (square) is located at the center. The instance counts 30 customer regions (stars) and 20 waiting vertices (circles).Although it is not visible here, the instance has a time horizon of 480 units and counts 24 time slots. If the operational day lasts 8 hours, a time unit represents a 1 minute in real time and each time slot lasts 20 minutes.

Potential requests. Each potential request r, associated with a customer region and a time slot, comes with a deterministic service time $s_r = 10$. The time window $[e_r, l_r]$ is such that e_r is chosen uniformly in $[\Gamma_r, \min(\Gamma_r + \frac{h}{n_{\text{TS}}}], h)$, and l_r is chosen uniformly in $[\max(e_l, d_{0,r}), \max(e_l + 10, d_{0,r})]$.

Scale parameter. A *scale* parameter is introduced in order to optimize expectations on coarser data, and therefore to speed-up computations. When equal to 1, expectations are computed while considering the original horizon. When *scale* > 1, expectations are computed from a coarse version of the initial horizon, scaled down by the factor $scale$. If *scale* =10 for instance, then the horizon is scaled to $h' = 48$. All the time data, such as travel and service times, but also time windows and reveal times, are then scaled as well (rounding up to nearest integer). When working on a scaled horizon (i.e. scale > 1), Algorithm 1 deals with an approximate but easier objective function $\mathcal{Q}^{\mathcal{R}}(x, \tau)$, in $\mathcal{O}(n(\frac{h}{\text{scale}})^2)$, and a reduced search space due to a coarse time horizon.

Algorithm 1 is then modified by simply adapting line 7 to return the best solution encountered so far, *according to the initial horizon*. Each time a new best solution is found during the search, its true expected cost is computed after scaling it up back to the original horizon, multiplying arrival, departure and waiting times by a factor *scale*.

Experimental plan. All experiments are done under a cluster composed of 32 64-bits AMD Opteron 1.4GHz cores. The code is developed in *C++11* and compiled with LLDB using *-O3* optimization flag. The Simulated Annealing parameters are set to $T_{\text{init}} = 5$, $T_{\text{min}} = 10^{-6}$ and $\alpha = 0.995$.

	30 seconds				3 minutes				30 minutes			
scale:	**1**	**2**	**5**	**10**	**1**	**2**	**5**	**10**	**1**	**2**	**5**	**10**
a.1	19.7	18.2	**15.0**	16.2	16.7	16.4	**14.9**	16.2	16.1	15.3	**14.9**	16.5
a.2	22.2	20.1	**16.4**	17.5	17.7	16.8	**16.2**	17.4	16.7	16.3	**16.2**	17.5
a.3	20.3	20.0	**14.4**	16.1	16.1	15.9	**14.0**	16.0	15.6	15.2	**14.0**	16.2
b.1	21.1	16.9	11.1	**11.0**	8.2	**6.3**	7.1	9.9	**5.6**	5.7	6.9	9.6
b.2	22.1	17.5	**9.6**	11.4	5.6	7.7	**6.8**	9.9	**5.1**	7.4	6.4	9.3
b.3	22.2	17.0	**10.8**	11.9	8.7	**7.2**	7.8	11.1	**6.2**	8.3	7.2	10.8
c.1	42.1	38.7	**23.6**	25.9	15.3	13.9	**12.2**	19.3	**8.3**	9.3	11.0	16.7
c.2	43.9	37.2	**25.8**	27.8	14.0	14.9	**13.5**	19.8	**9.9**	10.4	11.6	17.9
c.3	42.4	39.2	**24.3**	24.8	17.5	14.9	**11.7**	17.5	15.2	**8.8**	10.3	15.8
d.1	71.6	67.9	54.8	**54.3**	46.5	30.4	**26.1**	39.6	**11.8**	13.0	19.2	32.7
d.2	72.2	67.9	**52.8**	56.2	40.9	34.7	**25.7**	40.1	**12.6**	19.0	20.3	31.4
d.3	73.0	67.4	53.8	**51.5**	40.8	37.2	**23.0**	35.9	17.4	**11.9**	18.4	28.4

Table 1. Experimental results while varying horizon scale and computational time.

Results. Table 1 shows average experimental results over 10 runs on 12 instances: Instances $a.x$ (resp. $b.x$, $c.x$ and $d.x$) are such that $|C| = 30$ (resp. 30, 50 and 80), $|W| = 5$ (resp. 20, 30 and 50), and $K = 1$ (resp. 3, 5 and 10). Results are reported with *scale* $\in \{1, 2, 5, 10\}$ and with a CPU time limit $\in \{30, 180, 1800\}$ seconds.

Provided a limited computational time of 30 seconds, using a scaled horizon leads to better results. This is easily explained by the limited number of local search iterations performed when *scale*=1. As the computational time increases to 3 minutes, working on the original horizon size tends to provide better results. This trend is confirmed by moving to 30 minutes. As the available computational time increases, the accuracy in the objective function eventually overtakes the computational efficiency provided by scaled horizons, especially for large instances such as $c.x$ and $d.x$.

With their unique vehicle and because requests are uniformly distributed, instances $a.x$ may suffer from an evaluation function being roughly uniform. Sending the vehicle at some location to wait there is, most of the time, more or less equivalent to another location. Consequently, local optima are numerous and little diversified, the more promising ones being hard to detect when using scale 1 for only 30 minutes. In the contrary, using more vehicles (e.g. instances $b.x$) leads to a less uniform evaluation function. For example, concentrating all the vehicles in the same region would surely leads to poor results. On instances $a.x$, the diversification brought by scaled horizons then still prevails after 30 minutes. Given larger computation times (5 hours), results on scales 5 and 10 do not show a significant improvement:

	a.1				a.2				a.3			
scale $\in \{1, 2, 5, 10\}$:	15.3	15.1	**14.9**	16.5	16.2	**15.8**	16.1	17.3	14.8	14.6	**14.0**	16.1

Scales 1 and 2 in contrary tend to take promising benefits of a larger computation time.

Figure 4 shows how, for instance *c.1*, the real objective function (i.e. according to the original horizon) evolves in average (over 10 runs) during an execution of Algorithm 1. By reducing both the granularity of the search space and the complexity of the objective function, the parameter *scale* can therefore be used as a tradeoff between responsiveness and good quality solutions on the long term. Figure 4 also shows that the parameter *scale* can dynamically be reduced during the search.

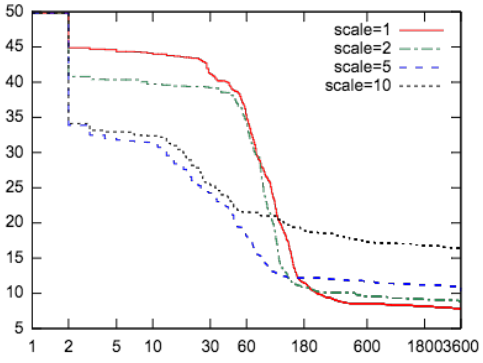


Fig. 4. Average evolution of the best real objective value in Algorithm 1, during 3600 seconds on instance $c.1$. During the first second, objective values rapidly decrease when optimizing on scaled horizon. Thereafter, depending on the available computation time, some scale factors reveal to be more efficient than others. For less than 1 minute, *scale*=5 leads to better results. With at least 10 minutes, using the original horizon is definitely better.

8 Conclusions and research directions

We introduced a new stochastic VRP, the SS-VRPTW-CR. Unlike existing SS-VRPs with random customers, we don't make any assumption on the moment at which a customer reveals its presence or absence. Instead, this is treated as a random variable as well. We proposed a recourse strategy for a special case of the SS-VRPTW-CR, when there is no maximal vehicle capacities. We showed how the exact expected cost can be computed in pseudo-polynomial time under this recourse strategy, and how to integrate in an efficient meta-heuristic method. Experiments are driven on generated test instances of various sizes. The average results show how a scale parameter, controlling the granularity of the time horizon, can be used to tune the optimization process in the case of limited computational times.

Maximal vehicle capacity constraints. The recourse strategy and equations we give can be extended to take care of vehicle capacities. We are currently working on a generalized version of these equations.

Contribution to online optimization. Another potential application of the SS-VRPTW-CR goes to online optimization problems such as the DS-VRPTW. Because of the huge complexity of reoptimization, heuristic methods are often preferred, including the so called Sample Average Approximation (SAA, see [26]). SAA relies on Monte Carlo sampling, making decisions based on a subset of the scenarios. Thanks to recourse strategies, the SS-VRPTW-CR provides an upper bound on the expected cost of a first stage solution under optimal reoptimization. The SS-VRPTW-CR could therefore be used as a subroutine in order to heuristically solve the DS-VRTPW, whilst considering the whole set of scenarios instead of only a subset of sampled ones. In such a context, the scale parameter we introduce in the experiments can be of great contribution.

Acknowledgments Christine Solnon is supported by the LABEX IMU (ANR-10-LABX-0088) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR).

References

1. Bent, R.W., Van Hentenryck, P.: Waiting and Relocation Strategies in Online Stochastic Vehicle Routing. IJCAI (2007) 1816–1821
2. Ichoua, S., Gendreau, M., Potvin, J.Y.: Exploiting knowledge about future demands for real-time vehicle dispatching. Transportation Science **40**(2) (may 2006) 211–225
3. Saint-Guillain, M., Deville, Y., Solnon, C.: A Multistage Stochastic Programming Approach to the Dynamic and Stochastic VRPTW. In: International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems. (2015) 357–374
4. Branke, J., Middendorf, M., Noeth, G., Dessouky, M.: Waiting Strategies for Dynamic Vehicle Routing. Transportation Science **39**(3) (aug 2005) 298–312
5. Bertsimas, D.J.: A vehicle routing problem with stochastic demand. Operations Research (1992)

6. Jaillet, P.: Probabilistic traveling salesman problems. PhD thesis, Massachusetts Institute of Technology (1985)
7. Laporte, G., Louveaux, F.V., Mercure, H.: A priori optimization of the probabilistic traveling salesman problem. Operations Research **42**(3) (1994) 543–549
8. Laporte, G., Louveaux, F.V.: The integer L-shaped method for stochastic integer programs with complete recourse. Oper. Res. Lett. **13**(3) (1993) 133–142
9. Jezequel, A.: Probabilistic vehicle routing problems. PhD thesis, Massachusetts Institute of Technology (1985)
10. Bertsimas, D.J., Chervi, P., Peterson, M.: Computational approaches to stochastic vehicle routing problems. Transportation science (1995) 1–34
11. Bianchi, L., Campbell, A.M.: Extension of the 2-p-opt and 1-shift algorithms to the heterogeneous probabilistic traveling salesman problem. European Journal of Operational Research **176**(1) (jan 2007) 131–144
12. Bowler, N.E., Fink, T.M., Ball, R.C.: Characterisation of the probabilistic travelling salesman problem. Physical Review E **68**(3) (nov 2003)
13. Bianchi, L., Gambardella, L.M., Dorigo, M.: An ant colony optimization approach to the probabilistic traveling salesman problem. Parallel Problem Solving from Nature (2002) 883–892
14. Waters, C.D.J.: Vehicle-scheduling problems with uncertainty and omitted customers. Journal of the Operational Research Society (1989) 1099–1108
15. Gendreau, M., Laporte, G., Séguin, R.: An exact algorithm for the vehicle routing problem with stochastic demands and customers. Transportation Science **29**(2) (1995) 143–155
16. Séguin, R.: Problemes stochastiques de tournees de vehicules. Centre de Recherche sur les Transports Publication **979** (1994)
17. Gendreau, M., Laporte, G., Séguin, R.: A Tabu Search Heuristic for the Vehicle Routing Problem with Stochastic Demands and Customers. Operations Research **44**(3) (1996) 469–477
18. Gounaris, C.E., Repoussis, P.P., Tarantilis, C.D., Wiesemann, W., Floudas, C.A.: An adaptive memory programming framework for the robust capacitated vehicle routing problem. Transportation Science (2014)
19. Campbell, A.M., Thomas, B.W.: Probabilistic traveling salesman problem with deadlines. Transportation Science **42**(1) (2008) 1–27
20. Henchiri, A., Bellalouna, M., Khaznaji, W.: A probabilistic traveling salesman problem: a survey. In: FedCSIS Position Papers. Volume 3. (sep 2014) 55–60
21. Sungur, I., Ren, Y.: A model and algorithm for the courier delivery problem with uncertainty. Transportation science **44**(2) (2010) 193–205
22. Chao, I.M., Golden, B.L., Wasil, E.A.: The team orienteering problem. European Journal of Operational Research **88**(3) (1996) 464–474
23. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598) (1983) 671–680
24. Kindervater, G.A.P., Savelsbergh, M.W.P.: Vehicle routing: handling edge exchanges. Local search in combinatorial optimization (1997) 337–360
25. Taillard, É., Badeau, P., Gendreau, M., Guertin, F., Potvin, J.Y.: A tabu search heuristic for the vehicle routing problem with soft time windows. Transportation science **31**(2) (1997) 170–186
26. Ahmed, S., Shapiro, A.: The sample average approximation method for stochastic programs with integer recourse. Submitted for publication (2002)