Applied Soft Computing 32 (2015) 518–531

Contents lists available at ScienceDirect

Applied Soft Computing

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

Local search based metaheuristics for the robust vehicle routing problem with discrete scenarios

Elyn Solano-Charris[a,b], Christian Prins[a,*], Andréa Cynthia Santos[a]

[a] ICD-LOSI, UMR CNRS 6281, Université de Technologie de Troyes, 12, rue Marie Curie, CS 42060, 10004 Troyes Cedex, France
[b] Universidad de La Sabana, Escuela de Ciencias Económicas Y Administrativas, Chía (Cundinamarca), Colombia

ARTICLE INFO

Article history:
Received 11 April 2014
Received in revised form 10 March 2015
Accepted 22 March 2015
Available online 16 April 2015

Keywords:
Vehicle routing
Uncertain travel times
Robust optimization
Greedy randomized adaptive search procedure
Iterated local search
Tour splitting

ABSTRACT

The Capacitated Vehicle Routing Problem (CVRP) is extended here to handle uncertain arc costs without resorting to probability distributions, giving the Robust VRP (RVRP). The unique set of arc costs in the CVRP is replaced by a set of discrete scenarios. A scenario is for instance the travel time observed on each arc at a given traffic hour. The goal is to build a set of routes using the lexicographic min–max criterion: the worst cost over all scenarios is minimized but ties are broken using the other scenarios, from the worst to the best. This version of robust CVRP has never been studied before. A Mixed Integer Linear Program (MILP), two greedy heuristics, a local search and four metaheuristics are proposed: a Greedy Randomized Adaptive Search Procedure, an Iterated Local Search (ILS), a Multi-Start ILS (MS-ILS), and an MS-ILS based on Giant Tours (MS-ILS-GT) converted into feasible routes via a lexicographic splitting procedure. The greedy heuristics provide the other algorithms with good initial solutions. Tests on small instances (10–20 customers, 2–3 vehicles, 10–30 scenarios) show that the four metaheuristics retrieve all optima found by the MILP. On larger cases with 50–100 customers, 5–20 vehicles and 10–20 scenarios, MS-ILS-GT dominates the other approaches. As our algorithms share the same components (initial heuristic, local search), the positive contribution of using the giant tour approach is confirmed on the RVRP.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

The Capacitated Vehicle Routing Problem (CVRP) is a classical combinatorial optimization problem raised by the distribution of goods in logistic networks. Due to its important applications in industry and its theoretical interest, it has generated an abundant literature with two strong trends motivated by the lack of realism of academic problems.

The first trend is to study *rich vehicle routing problems* which combine many features from real applications. The other one is to try to build *robust solutions*, less affected by the various uncertainties encountered on the field. Several communities have their own interpretation of robustness and propose various tools to evaluate or achieve it, e.g., simulation methods, stochastic programming, robust optimization and fuzzy logic.

Robust optimization is used here for a Robust VRP (RVRP) with uncertain travel times modeled by a set of discrete scenarios. A mathematical model, two constructive heuristics, one local search and four metaheuristics are proposed to determine a set of routes minimizing the worst cost (total cost of the routes) over all scenarios. The paper is structured as follows. Section 2 gives a literature review. The problem is specified and modeled in Section 3. The lexicographic comparison of solutions is explained in Section 4. Sections 5 and 6 present the greedy heuristics and the local search shared by all our metaheuristics, described in Section 7. A computational evaluation is conducted in Section 8, before a conclusion in Section 9.

2. Literature review

As a vast literature is devoted to vehicle routing and robust optimization, this section in four parts focuses on essential references and recent works. The first part cites a few key-articles on the CVRP,

* Corresponding author.
E-mail addresses: erlyn.solano_charris@utt.fr, erlyn.solano@unisabana.edu.co (E. Solano-Charris), christian.prins@utt.fr (C. Prins), andrea.duhamel@utt.fr (A.C. Santos).

http://dx.doi.org/10.1016/j.asoc.2015.03.058
1568-4946/© 2015 Elsevier B.V. All rights reserved.

the second one recalls the principles of stochastic and robust optimization, the third part discusses the position of our work with respect to published works on uncertain travel time while the last part reviews the scant literature on robust VRPs.

2.1. The Capacitated Vehicle Routing Problem

The Vehicle Routing Problem (VRP) introduced by Dantzig and Ramser [18] has become a central problem in distribution management. The aim of this NP-hard combinatorial optimization problem is to determine a set of routes with minimum total cost, to serve a set of customers with known demands using a fleet of identical vehicles based at a depot node. Most authors address the version with capacitated vehicles, called Capacitated VRP (CVRP).

In addition to its industrial importance, the VRP has led to hundreds of variants with additional attributes such as heterogeneous fleets of vehicles, multiple depots, multi-period horizons, etc. It is also a laboratory-problem to test new ideas in optimization. As quoted by Laporte [32], "The study of the VRP has given rise to major developments in the fields of exact algorithms and heuristics. In particular, highly sophisticated mathematical programming approaches and powerful metaheuristics for the VRP have been put forward in recent years".

The VRP has inspired a rich literature. A taxonomic review published in 2009 by Eksioglu et al. [19] found 1494 papers from 1954 to 2006 (included) in bibliographical databases, using only "vehicle routing" as search expression. As the review of a so vast literature is beyond the scope of this article, we prefer to provide the reader with a few key-references on the CVRP, before discussing robustness issues. The best entry points, covering both exact algorithms, heuristics and applications, are the books by Toth and Vigo [53] and Golden et al. [23].

A large number of exact and heuristic algorithms are available for the CVRP. Only relatively small instances with one hundred customers can be consistently solved to optimality by the best exact methods like the branch-and-cut-and-price algorithm from Fukasawa et al. [21] and the more recent and general approach proposed by Baldacci and Mingozzi [4]. As many real instances are much larger, heuristics are required to determine good solutions in reasonable running times.

The simplest and fastest methods are constructive heuristics, such as the famous savings method from Clarke and Wright [15] A popular strategy consists in reducing the CVRP to the Traveling Salesman Problem (TSP). In *cluster-first route-second methods*, clusters of customers compatible with vehicle capacity are determined then a TSP is solved to build a route in each cluster. In the opposite and less common *route-first cluster-second* approach, a TSP tour is partitioned into CVRP routes. The interested reader will find many classical heuristics in two surveys by Laporte et al. [33] et Cordeau et al. [17]. A recent review by Prins et al. [48] covers route-first cluster-second algorithms.

Concerning metaheuristics, the tabu search algorithms which were the leaders in the 90s have been progressively superseded in the last decade by evolutionary algorithms [38,47] and adaptive large neighborhood search [46]. The most effective metaheuristic for the CVRP is currently a hybrid genetic algorithm designed by Vidal et al. [54] and recently generalized to 26 VRP variants [56]. Vidal et al. [55] survey the state of the art metaheuristics available nowadays for rich vehicle routing problems.

2.2. Robustness in optimization

The word "robust" has various meanings in optimization, automatic control and engineering. In practice, the data for an optimization problem can be uncertain, inexact, noised, or likely to change in future. An optimal solution computed using current parameters can be strongly affected by perturbations, becoming suboptimal or even infeasible. What most decision makers call *robust solution* is a solution resisting as much as possible such perturbations.

Robustness can be addressed via *stochastic optimization* when uncertainty has a probabilistic description. Two main approaches are used to solve stochastic optimization problems. In *chance-constrained Programming*, an optimal solution is calculated to satisfy the constraints with a given probability. In *stochastic programming with recourse*, some feasibility constraints are relaxed and included in the objective function, assuming that violations induced by random events after the implementation of first stage decisions can be repaired by recourse actions. The goal is then to minimize the expected cost for the two stages. As stochastic optimization is not applied here, we recommend only three good entry points: Birge and Louveaux [11] for an overview, Bianchi et al. [10] about metaheuristics in stochastic combinatorial optimization, and Gendreau et al. [22] on stochastic vehicle routing problems.

Stochastic models are powerful but have three drawbacks: the underlying probability distributions must be known, their convolutions must be computationally tractable, and the solutions can become infeasible for some realizations of random events, e.g., when recourse is impossible. *Robust Optimization* (RO) is a more recent framework to handle uncertainty while avoiding these drawbacks. This approach is no longer stochastic but rather deterministic and set-based. According to Bertsimas et al. [8], "instead of seeking to immunize the solution in some probabilistic sense to stochastic uncertainty, here the decision-maker constructs a solution that is optimal for any realization of the uncertainty in a given set".

One branch of RO studies mathematical programs for which tractable robust counterparts can be obtained and develops computational tools, see for instance Ben-Tal et al. [6] for a review. Other authors like Kouvelis and Yu [30] or Aissi et al. [2] focus on combinatorial optimization problems and investigate the algorithmic complexity of their robust versions. For instance, finding a shortest path problem in a graph is a polynomial problem, while the robust version with two scenarios for the arc costs is NP-hard, even in layered networks [58].

Choosing an uncertainty set and a robustness criterion is critical when dealing with a RO problem. The uncertainty set can be defined by a convex set, e.g. a polyhedron [6,8], or by an assignment of plausible values to each model parameter, called *scenario*. In *interval scenarios*, each parameter can take any value in a given interval while in *discrete scenarios* considered in our study its possible values are explicitly listed. In practice, most companies record archives or historical data which can be mined to provide realistic scenarios.

Concerning robustness criteria, consider to fix ideas a minimization problem with a feasible set X and a set S of p discrete scenarios. Let $f(x, k)$ be the cost of solution x in scenario k, x_k^* an optimum for scenario k, and f_k^* its cost. The *min–max version* [57] seeks a solution minimizing the worst cost over all scenarios, i.e., $\min_{x\in X} \max_{k\in S}\{f(x, k)\}$. The *lexicographic min–max version* [44] uses the other scenarios, from the worst to the best, to break ties. The *lexicographic α-robustness* [27] is similar but includes a tolerance threshold α to avoid discriminating among solutions with similar values. The *min–max regret* version $\min_{x\in X} \max_{k\in S}\{f(x, k) - f_k^*\}$, discussed for instance in [3,12], aims at minimizing the maximum deviation (regret) to the optimal solution values f_k^*, $k = 1, 2, \ldots, p$. As robust solutions are often too conservative, Bertsimas and Sim [9] propose to limit the number of uncertain parameters allowed to deviate from their nominal values to a number Γ, called *budget of uncertainty*. Applied to the cost and constraint coefficients of a linear or mixed integer program, their approach leads to a robust version with a moderate increase in size. In particular, a 0–1 linear program with uncertainties limited to the n cost coefficients can be treated by solving at most $n + 1$ instances of the original problem.

NP-hard discrete RO problems can also be solved using classical exact approaches, such as branch-and-cut or Dantzig-Wolfe decomposition, see examples in [12] for the min–max regret criterion. A significant stream of research develops heuristics with performance guarantees [2,3,28]. Surprisingly, very few metaheuristics have been developed. For instance, Nikulin designed a simulated annealing algorithm for a robust spanning tree problem [41]. A few other heuristics are cited in the next subsection on robust vehicle routing.

2.3. Position of our work in the research on uncertain travel times

Our RVRP considers uncertain travel modeled by discrete scenarios, each scenario defining one travel time for each arc. The aim is to minimize the worst cost (total route duration) over all scenarios, using a lexicographic method to break ties. These scenarios do not correspond to realizations of random variables and are not associated with probabilities of occurrence.

In practice, uncertainties can also affect demands (e.g., in waste collection), service times (repairs), the presence of customers (parcel delivery), and departure times (aircraft waiting for good weather conditions). Apart from robust and stochastic optimization, approaches such as constraint relaxation, spare capacity in planned routes, simulation, and fuzzy logic can be employed. For instance, strict appointments for customer deliveries can be replaced by time windows to better resist traffic perturbations, giving a VRP with time windows [37].

A key-issue discussed by Gómez et al. [24] is the accurate modeling of travel times. Most authors use additive probability distributions, e.g., normal [29] or gamma laws [50]. As travel times often grow quickly from a minimum to a maximum and then slowly decrease with a long tail, a lognormal distribution looks pertinent [34]. According to Gómez et al. [24], an accurate distribution may not exist and a specific one must be chosen for each problem and even each arc. These authors propose to use phase-type distributions, which can approximate any positive and continuous distribution while computing exactly their convolutions.

Concerning the realism of our discrete scenarios, they can be provided by the traffic control centers implemented in most large cities, with two advantages: they correspond to real data and reflect possible correlations among arc costs. We think interesting solutions can be obtained if enough traffic records are considered. The price to pay is a running time proportional to $p \log p$ for p scenarios, as seen in Section 5. Anyway, in our opinion, the confirmation of the accuracy of all models handling uncertain travel times would require large-scale validations on real road networks, during a few weeks at least, to see the real cost achieved on the field. Obviously, such validations would be very costly.

Another problem is route feasibility. When travel and/or service times are uncertain, a solution may be infeasible under route-duration restrictions. For instance, the multi-start sampling heuristic used in [24] is based on chance-constrained programming: It minimizes the total expected duration of the routes under service level constraints which keep the probability of route failure below a small probability. Chen et al. [14] propose one chance-constrained model and a second one allowing recourse actions. Time windows can also lead to infeasibility. All published works use soft windows and penalize their violations (early and late arrivals at customers) in the objective function, e.g., [50]. We notice that another possibility, not yet studied, would be to consider hard windows and allow each vehicle to perform several trips.

In our RVRP, the planned routes remain feasible in spite of uncertainties. This situation is not common but we observed an example in our city (Troyes), where the municipality sends technicians to replace damaged bulbs of public lights. Routes are prepared using average travel times but, due to traffic conditions, the actual duration of the routes is often larger. The maximum daily working time is not exceeded, but the uncertainties delay the allocation of technicians to other tasks afterwards, like repairs in municipal buildings.

Finally, a strong point of our approach is mentioned in Section 4. The RVRP can be viewed as a multi-objective problem, each objective being the total cost for one scenario. An optimum for our lexicographic min–max objective is also Pareto-optimal for this multi-objective version. As no other solution dominates it, it is appealing for a decision maker.

2.4. Vehicle routing problems solved via robust optimization

The first robust capacitated problem, the CVRP with uncertain demands, was studied only in 2008 by Sungur et al. [49]. These authors consider demand vectors built as deviations around mean values and three uncertainty sets defined as linear combinations of these vectors (convex hull, ellipsoid, and box), following an approach from Ben-Tal and Nemirovski for robust linear programs [7]. The Miller-Tucker-Zemlin (MTZ) model for the CVRP, a mixed integer linear program (MILP) with one binary variable per arc, can be adapted to each set while confining the uncertainty to the right-hand side of subtour elimination constraints. Tests with a MILP solver for 15 to 100 customers and the convex hull version indicate that robust solutions offer a good protection against uncertain demands, with only a small extra cost over optimal CVRP routes. However, most instances beyond 50 customers are not solved and total demands exceeding 90% of fleet capacity lead in general to infeasibility.

In a tutorial paper, Ordóñez [43] shows that the MTZ model for the CVRP can be modified to handle uncertain demands, travel times, travel costs, or customers. His tests on small instances (12 clients) compare Sungur's approach, which uses the convex hull of demand vectors as uncertainty set, with chance-constrained and recourse methods. They show that the RO approach, which has often the reputation of being too conservative, achieves the same costs as stochastic programming for demands deviating from nominal values by 10 to 15%.

Gounaris et al. [25] study a more generic case with demands supported on a polyhedron. Robust versions are described for the MTZ, the Two-index Vehicle Flow (2VF), and the one and two-commodity flow formulations. A robust counterpart of the Rounded Capacity Inequalities (RCI) is also derived. On instances with 15–135 customers, using CPLEX, the 2VF model reinforced by RCI cuts can solve to optimality most instances up to 50 nodes.

Three other papers deal with uncertain demands. Moghaddam et al. [40] use Particle Swarm Optimization (PSO), with one term favoring a balanced loading of vehicles in the objective function. Compared with Sungur et al. exact method [49], all instances can be solved with an average saving of 7%, provided one or two unmet demands are tolerated.

Noorizadegan et al. [42] select the heterogeneous fixed-fleet VRP and develop a MILP based on three-index variables x_{ij}^k (equal to 1 if arc (i, j) is used by vehicle k) and MTZ subtour elimination constraints. There again, tractable robust formulations affecting only the right-hand side of these constraints can be developed for three uncertainty frameworks: Ben-Tal and Nemirovski [7], Bertsimas and Sim [9] and chance-constrained programming. The obtained models are compared on instances with 20 clients using CPLEX.

Erbao et al. [13] address the robust Open VRP, where vehicles do not return to the depot after the last customer. A three-index MILP with MTZ constraints is given, without being used in numerical evaluations. A differential evolution algorithm (a population metaheuristic) is made robust using four simple rules, including the load balancing system of Moghaddam et al. [40]. The resulting heuristics are compared using 16 classical OVRP instances with 50–199 customers, modified by augmenting the demands randomly between 0 and 10%.

A few research works have dealt with uncertain travel costs or times. Agra et al. [1] address an uncapacitated VRP with time windows raised by maritime transportation. Travel costs and times are asymmetric and depend on the vessel used. The goal is to find a least-cost set of routes respecting time windows and feasible for all travel times in a given uncertainty polytope. Two new formulations are proposed and solved by ad-hoc decomposition algorithms, made more efficient if the "budget of uncertainty" approach of Bertsimas and Sim [9] is used. Instances with 20–50 nodes can be solved exactly in 30 min on a 2.5 GHz PC.

An approach between robust optimization and stochastic programming is proposed by Han et al. [26] for the uncapacitated VRP with uncertain travel times. A set of time intervals is associated with each arc. A scenario is defined by selecting for each arc one interval according to a given probability, e.g., the travel on a road can last 10–15 min if traffic is normal (probability 30%) or 25–30 min in case of congestion (probability 70%). The idea is to find for each scenario a robust solution, minimizing the worst cost over the selected intervals, and then a solution with minimum expected cost. A branch and cut algorithm is developed and tested on instances with 10–25 customers and 2 intervals per arc. Toklu et al. [51] tackle the CVRP with uncertain travel costs defined by intervals. They adopt the approach from Bertsimas and Sim [9] to perturb a limited number Γ of arc costs. Their solution method is an ant colony system (an elitist form of ant colony optimization), tested on instances with 100 or 150 customers. The same authors improve their results in [52] using several ant systems to build a pool of robust and diversified solutions.

Finally, uncertainties on both travel times and demands are considered by Lee et al. [35] for a VRP with customer deadlines. A budget of uncertainty is defined by limiting the sum of deviations of travel times and demands to their nominal values. A set-partitioning formulation is proposed and solved via column generation. The uncertainties are limited to the column generation subproblem, solved using a robust version of a shortest path algorithm with resource constraints. Instances with 20–40 customers are solved to optimality.

3. Motivation, problem definition and mathematical model

Summarizing the literature review, most works on robust vehicle routing develop min–max models which are directly solved via MIP solvers. As the deterministic versions are already NP-hard, this approach is limited to 50 customers. Only four metaheuristics, all based on interval scenarios, were published for larger instances. Our contribution is (a) to study a new kind of RVRP with discrete scenarios and uncertain arc costs, (b) to optimize a lexicographic min–max criterion to break ties when two scenarios give the same worst cost, (c) to propose a mathematical model, and (d) to develop constructive heuristics and metaheuristics.

The mathematical model is mainly provided as a compact and unambiguous specification for the problem. It is derived from a model proposed by Kulkarni and Bhave [31] for the classical capacitated VRP (without uncertainties), but it is extended here to handle multiple scenarios and optimize a different objective function. The model has really been implemented and tested, in Section 8. Only small instances can be solved to optimality, but the model is useful to show that our metaheuristics can retrieve most optimal solutions on such instances.

Concerning proposed algorithms, we first designed constructive heuristics, to provide the other methods with good initial solutions. They are also useful for tackling large instances in reasonable running times. Then we elaborated a local search procedure, which can be used either to improve the solutions obtained by the simple heuristics or to serve for intensification in our metaheuristics, which are based on the GRASP and ILS frameworks.

The Robust Vehicle Routing Problem (RVRP) addressed here can be defined as follows. As the two directions of a road can be differently affected when traffic is perturbed, we consider a complete but loopless directed graph $G = (V, A)$, while the CVRP is defined on an undirected network. V denotes the node-set, with one depot (node 0) and n customers with positive demands q_i, while $A = \{(i, j) \mid i, j \in V, i \neq j\}$ stands for the arc-set. A fleet of m identical vehicles with capacity Q is based at the depot.

Instead of being random variables, the uncertain arc costs are modelled by a set of p discrete scenarios $S = \{1, 2, \ldots p\}$, where each scenario k defines one non-negative cost c_{ij}^k for each arc $(i, j) \in A$. Let F define the set of feasible solutions. Like in the CVRP, a solution $\omega \in F$ is a set of routes: each route is done by one vehicle which leaves the depot, serves a subset of customers whose total demand does not exceed Q, and returns to the depot.

Let $cost(\omega, k)$ be the cost of solution ω for scenario k, i.e., the total cost of the routes with the arc costs of this scenario. Define $cost(\omega) = (cost(\omega, 1), cost(\omega, 2), \ldots, cost(\omega, p))$ as the vector of costs for all scenarios, and $worst(\omega) = \max\{cost(\omega, k) \mid k \in S\}$ the worst cost. The goal of our mathematical model is to determine a solution ω^* minimizing this worst cost over all scenarios, i.e., such that $\omega^* = \arg\min\{worst(\omega) \mid \omega \in F\}$. We show in the next section how to solve iteratively the *lexicographic min–max version*. As the CVRP, known to be NP-hard, is the RVRP with one scenario ($p = 1$), the RVRP is also NP-hard.

The RVRP can be specified by the following compact MILP, in which no vehicle index is required. The routes are defined by binary variables x_{ij}, equal to 1 if and only if arc (i, j) is used in the solution. The resulting trips can be visualized by plotting the nodes and drawing one segment from node i to node j if $x_{ij} = 1$. Constraints (3) and (4) mean that one arc is used to reach and leave each client. The maximum fleet size is guaranteed via constraints (5). Inequalities (6) and (7) generalize the MTZ subtour-elimination constraints for the Traveling Salesman Problem (TSP) [39]. Assuming that q_i is a collected quantity, t_i is the vehicle load when leaving i. If arc (i, j) is not used, the constraint (7) for this arc becomes $t_i - t_j \leq Q - q_j$ and is trivially satisfied: as $t_i \leq Q$ and $t_j \geq q_j$ via (6), the left-hand side cannot exceed $Q - q_j$.

$$\min \quad z = \delta \tag{1}$$

$$\sum_{(i,j)\in A} c_{ij}^k x_{ij} \leq \delta \quad \forall \quad k \in S \tag{2}$$

$$\sum_{j\in V} x_{ji} = 1 \quad \forall \quad i \in V\backslash\{0\} \tag{3}$$

$$\sum_{j \in V} x_{ij} = 1 \quad \forall \quad i \in V \backslash \{0\} \tag{4}$$

$$\sum_{i \in V \backslash \{0\}} x_{0i} \le m \tag{5}$$

$$q_i \le t_i \le Q \quad \forall \quad i \in V \backslash \{0\} \tag{6}$$

$$t_i - t_j + Q\, x_{ij} \le Q - q_j \quad \forall \quad (i, j) \in A, i \ne 0, j \ne 0 \tag{7}$$

$$x_{ij} \in \{0, 1\} \quad \forall \quad (i, j) \in A \tag{8}$$

$$\delta \ge 0 \tag{9}$$

Consider now a route $(r_1, r_2, \ldots, r_u)$, which means $x_{r_i, r_{i+1}} = 1$ for $i = 1, 2, \ldots, u - 1$: constraints (6) and (7) hold by setting for instance $t_{r_1} = 0$ and $t_{r_i} = t_{r_{i-1}} + q_{r_i}$ for $i = 2, 3, \ldots, u$. Vehicle capacity is also respected since $t_{r_u} \le Q$ from (6). The binary variables are declared in (8). The other equations lead to a min–max version. In (2), the total cost of the arcs used for each scenario is bounded above by the variable δ, defined in constraint (9). The goal (1) is to minimize δ.

4. Lexicographic min–max robustness criterion

Preliminary tests with heuristics minimizing the worst cost gave mitigated results. Indeed, several moves can induce the same decrease in the worst cost and cannot be distinguished. However, a move with no effect on the worst cost can improve the second worst cost, the third one, etc., which suggests a lexicographic approach to determine which move is the best.

The *lexicographic min–max criterion*, see Ogryczak [44], has several interests. First, it gives the same worst cost as the classical min–max approach. Second, the solution offers a better protection against the second worst scenario, the third one, etc. Third, an optimal solution for this criterion is also Pareto-optimal for the multi-objective version of the problem, each objective being the total cost of the routes for one scenario [44]. Fourth, we observed a positive effect on the worst cost in local search procedures: while a search based on the min–max criterion is blocked as soon as no move improves the worst cost, the decrease of the worst cost can often restart after a few moves which reduce the other costs.

Our MILP of Section 3 can be solved lexicographically in p iterations. The first iteration solves the min–max version to get the worst scenario π and its cost z^*_π. The worst scenario is the one for which constraint (2) is tight. The second iteration solves the min–max version with the set of scenarios $S \setminus \{\pi\}$ and the additional constraint $\delta \le z^*_\pi$, etc.

In heuristics, lexicographic comparisons must be done explicitly. Using the notation of Section 3, consider one solution ω with its cost vector $cost(\omega)$ and build a vector $V(\omega)$ by sorting the components of $cost(\omega)$ in non-increasing order of costs. If $V(\omega, k)$ is the k-th component of $V(\omega)$, we have $worst(\omega) = V(\omega, 1)$. Solution ω is strictly better than solution ω' if and only if $V(\omega)$ is lexicographically smaller than $V(\omega')$, denoted as $V(\omega) < V(\omega')$. The comparison, done like words in a dictionary, is specified in Algorithm 1.

Algorithm 1. Lexicographic comparison of two solutions for p scenarios.

```
1:  function better (ω, ω′) : boolean
2:    $k \leftarrow 1$
3:    while ($k \le p$) and ($V(\omega, k) = V(\omega', k)$) do $k \leftarrow k + 1$ end while
4:    better ← ($k \le p$) and ($V(\omega, k) < V(\omega', k)$)
5:  end function
```

All our algorithms use lexicographic comparisons to select the best move among a set of candidates. For example, consider a classical operation, cheapest insertion, and one solution ω with $cost(\omega) = (10, 20, 30)$. Its lexicographic vector is $V(\omega) = (30, 20, 10)$, with a worst cost $worst(\omega) = V(\omega, 1) = 30$ reached in scenario 3. If customer i is inserted between nodes u and v in one trip, the cost variation for scenario k is $\Delta_k = c^k_{ui} + c^k_{iv} - c^k_{uv}$. Let Δ denote the cost variation vector for the p scenarios and compare one insertion $\omega \rightarrow \omega'$ with $\Delta' = (15, 16, 4)$, and one $\omega \rightarrow \omega''$ with $\Delta'' = (16, 12, 6)$. The resulting solutions are such that $cost(\omega') = (25, 36, 34)$ and $cost(\omega'') = (26, 32, 36)$. They have the same worst cost (36), even if it is now reached by scenario 2 for ω'. The lexicographic vectors are $V(\omega') = (36, 34, 25)$ and $V(\omega'') = (36, 32, 26)$. As $V(\omega'') < V(\omega')$, the second insertion will be preferred.

In a local search for a minimization problem, an improving move must reduce the cost of the incumbent solution. Here, a move without effect on the worst cost is accepted if it gives a lexicographically better solution. For instance, consider again ω with $cost(\omega) = (10, 20, 30)$ and $V(\omega) = (30, 20, 10)$. A move $\omega \rightarrow \omega'$ with $\Delta' = (2, -3, 2)$ is rejected because its degrades the worst cost. If $\Delta' = (2, -3, -1)$, it is accepted like in a classical local search since the objective function (the worst cost) is improved. Finally, if $\Delta = (9, -1, 0)$, the worst cost is unchanged but the move is accepted since $V(\omega') = (30, 19, 19) < V(\omega) = (30, 20, 10)$.

In the CVRP, most moves can be evaluated in $O(1)$ on the basis of cost variations, e.g., $c^k_{ui} + c^k_{iv} - c^k_{uv}$ in our previous example for node insertion. In the RVRP, the complexity becomes $O(p \log p)$ because of the sorting algorithm required to get the lexicographic vector.

5. Greedy heuristics

We tried first insertion heuristics, but these methods which work well on the CVRP collapse for multiple scenarios. We finally adapted the CVRP heuristic from Clarke and Wright (CW) [15], and derived also a randomized version RCW for our multi-start metaheuristics. From now on, it is assumed that a RVRP solution ω is encoded as an array of trips, while each trip, identified by its index, is stored as a list of customers between two copies of the depot.

5.1. Clarke and Wright heuristic for the RVRP

The original version of CW for the CVRP begins with a trivial solution composed of one dedicated trip for each customer (the "daisy") and then performs mergers (concatenations) of two trips in order to minimize the cost variation at each iteration. For one route with last customer i and another with first customer j, this cost variation $c_{ij} - c_{i0} - c_{0j}$ only depends on i, j and the depot. In parallel, each concatenation saves one vehicle.

An efficient implementation for the CVRP consists in sorting the $n(n-1)/2$ edges of the undirected graph in increasing order of cost variation, giving a list Λ. This step is possible in $O(n^2 \log n)$. Then each edge (i, j) of Λ is tested: if i and j are still at the extremities of two distinct routes whose total load does not exceed Q, then these two routes are merged. These tests cost $O(1)$ and the merger itself $O(n)$. As at most $n - 1$ mergers are executed during the algorithm, the total complexity is dominated by the initial sort in $O(n^2 \log n)$.

For the RVRP, CW is more involved due to the directed graph and the scenarios, see Algorithm 2. There are eight ways to merge two trips T and U: If $\bar{T}$ is the reversal of trip T, we have to test four cases (T, U), $(T, \bar{U})$, $(\bar{T}, U)$ and $(\bar{T}, \bar{U})$, plus four others by exchanging T and U. Four cases are enough in the CVRP, since for instance (T, U) and $(\bar{U}, \bar{T})$ have the same cost. Moreover, each merger must be evaluated for each scenario and the best merger is the one giving a solution with a minimum lexicographic vector. Finally, sorting the arcs at the beginning brings nothing because in the RVRP the route costs intervene in cost variations.

Algorithm 2. Clarke and Wright heuristic for the RVRP – $CW(\omega)$.

```
1:  compute the initial solution $\omega$ with one route per customer and set
      $nbtrips(\omega)$ to $n$
2:  prepare all pre-computed data
3:  **repeat**
4:    compute the lexicographic vector $V(\omega)$
5:    set $W_{best}$ to $(\infty, \infty, \ldots, \infty)$ and $found$ to false
6:    **for** each trip index $T$ of $\omega$ **do**
7:      $i, j \leftarrow$ first and last customers of $T$
8:      **for** each trip index $U$ of $\omega$ such that $U \neq T$ and $load(T) + load(U) \leq Q$ **do**
9:        $found \leftarrow$ true
10:       $u, v \leftarrow$ first and last customers of $U$
11:       $ctu \leftarrow cost(T) + cost(U)$ //Total cost of $T$ and $U$, to shorten formulas
12:       // evaluate concatenation $(T, U)$
13:       **for** $k \leftarrow 1$ **to** $p$ **do** $W_k \leftarrow cost(k) - ctu + b(T, k, j) + c^k_{ju} + a(U, k, u)$
          **end for**
14:       sort $W$ in decreasing cost order
15:       **if** $W < W_{best}$ **then** $W_{best} \leftarrow W$; $T_{best} \leftarrow T$; $U_{best} \leftarrow U$ **end if**
16:       // evaluate concatenation $(T, \bar{U})$
17:       **for** $k \leftarrow 1$ **to** $p$ **do** $W_k \leftarrow cost(k) - ctu + b(T, k, j) + c^k_{jv} + a(\bar{U}, k, v)$
          **end for**
18:       sort $W$ in decreasing cost order
19:       **if** $W < W_{best}$ **then** $W_{best} \leftarrow W$; $T_{best} \leftarrow T$; $U_{best} \leftarrow -U$ **end if**
20:       // evaluate concatenation $(\bar{T}, U)$
21:       **for** $k \leftarrow 1$ **to** $p$ **do** $W_k \leftarrow cost(k) - ctu + b(\bar{T}, k, i) + c^k_{iu} + a(U, k, u)$
          **end for**
22:       sort $W$ in decreasing cost order
23:       **if** $W < W_{best}$ **then** $W_{best} \leftarrow W$; $T_{best} \leftarrow -T$; $U_{best} \leftarrow U$ **end if**
24:       // evaluate concatenation $(\bar{T}, \bar{U})$
25:       **for** $k \leftarrow 1$ **to** $p$ **do** $W_k \leftarrow cost(k) - ctu + b(\bar{T}, k, i) + c^k_{iv} + a(\bar{U}, k, v)$
          **end for**
26:       sort $W$ in decreasing cost order
27:       **if** $W < W_{best}$ **then** $W_{best} \leftarrow W$; $T_{best} \leftarrow -T$; $U_{best} \leftarrow -U$ **end if**
28:     **end for**
29:   **end for**
30:   **if** $found$ and ($W_{best} < V(\omega)$ or $nbtrips(\omega) > m$) **then**
31:     **if** $T_{best} < 0$ **then** $T_{best} \leftarrow -T_{best}$; invert trip $T_{best}$ **endif**
32:     **if** $U_{best} < 0$ **then** $U_{best} \leftarrow -U_{best}$; invert trip $U_{best}$ **endif**
33:     concatenate the clients of $U_{best}$ at the end of $T_{best}$ then delete trip $U_{best}$
34:     update pre-computed data for trips $T_{best}$ and $\bar{T}_{best}$
35:   **end if**
36: **until** (not $found$)
```

In lines 1–2, CW builds the initial solution ω (daisy) and pre-computes four types of data: the number of trips, $nbtrips(\omega)$, the load of route T, $load(T)$, the cost of trip T before customer i in scenario k, $b(T, k, i)$, and the cost after i, $a(T, k, i)$. The parameter ω is implicit in the three last symbols, to make the notation lighter. These data are also prepared for the inverted trip $\bar{T}$. Fig. 1 shows on two cases of mergers how these data can be used.

Then, each iteration of the *repeat* loop browses all pairs of distinct trips (T, U) and searches for the best pair (lexicographically), even if this leads to a degraded solution. A boolean *found* is set to true if a feasible merger is detected.

For a given pair (T, U), only the first four cases of mergers are evaluated since the pair (U, T) is inspected by another iteration. For each case, CW determines the cost vector W of the resulting solution and sorts it to get the lexicographic vector. When improved, the lexicographic vector of the best merger (W_{best}) is updated and the corresponding pair of route indexes (T_{best}, U_{best}) is recorded, multiplied by -1 to remember that a route must be inverted. A detail is not mentioned in the algorithm to reduce its number of lines: the *for* loops which compute W lines 13, 17, 21, and 25 are exit if $W_k > W_{best}(1)$. This simple condition is sufficient to say that the merger tested is not better than the best one found so far.

In lines 30–35, provided feasible mergers have been detected ($found$ = true), the best merger is executed if it improves the incumbent solution or if fleet size is exceeded. In other words, we accept degrading mergers, but only to avoid vehicles in excess. The customers of trip U_{best} are moved at the end of T_{best}, U_{best} is deleted and the pre-computations updated for T_{best} only. The main loop stops when it finds no feasible merger.

Note that more than m vehicles can be used at the end, but we have never seen this case on the instances tested which, like all CVRP benchmarks, satisfy the necessary condition $m \geq \lceil \sum_{i=1}^{n} q_i/Q \rceil$. Should this occur, the local search in the next section can save vehicles by moving their customers to different trips. Anyway, determining if there exists a solution with at most m vehicles is a bin-packing problem, which is already NP-hard.

The initial daisy and all pre-computations can be done in $O(np)$. There are at most $n-1$ main iterations, when all routes are merged into one TSP tour. At each iteration, $O(n^2)$ pairs (T, U) are inspected and each pair is evaluated in $O(p \log p)$ due to the sorting algorithm. The complexity to execute the best merger is negligible: the two routes are inverted (if necessary) and concatenated in $O(n)$ while pre-computations are updated in $O(p)$ only for trips T_{best} and $\bar{T}_{best}$. Hence, CW runs in $O(n^3 p \log p)$ for the RVRP, instead of $O(n^2 \log n)$ for the CVRP.

5.2. Randomized version

As two of our metaheuristics perform multiple restarts, we also designed a randomized version RCW (Randomized CW) to provide them with initial solutions. Randomness is introduced by perturbing the solution costs, evaluated for each possible merger in Algorithm 2. We fix a perturbation level θ and compute before each merger evaluation a random number ρ ranging from 0 to θ percent of current solution cost. Finally, ρ is added to W_k, $k = 1, 2, \ldots, p$. The random generator is initialized with a fixed seed, to get reproducible results, and the algorithm is nested in a main loop doing a given number of iterations *ncalls*, without resetting the generator. The resulting procedure is invoked by a $RCW(\theta, ncalls, \omega)$ statement.

6. Local search procedure

Our metaheuristics call the local search LS of Algorithm 3. Each main iteration (lines 2–17) seeks the best improving move and stops when no such move exists. The inner loops (lines 4–12) browse each trip T, each node i in T, each trip U and each node j in U. The nodes can be customers or the depot at the beginning of the trips. If $T = U$, the procedure *MovesOnOneTrip* evaluates the moves designed for a single trip, otherwise *MovesOnTwoTrips* tests the moves affecting two distinct trips. The best improving move (if any) is applied to the incumbent solution in line 14.

The pre-computations in line 1 are used to speed up move evaluations. They prepare the same data as in the CW heuristic for each route T, each scenario k and each customer i in the route: the load of the route, $load(T)$, the cost to reach customer i, $b(T, k, i)$, and the cost to return to the depot after i, $a(T, k, i)$. Additionally, we pre-compute the cumulated quantity served up to customer i included, $load(T, i)$. The total complexity of these pre-computations is $O(np)$. After a move, they need to be updated only for the modified trips (line 15).

MoveOnOneTrip evaluates the following moves on a given trip T and two nodes i, j in T (note that vehicle capacity is always respected):

- relocate node i after node j, if i is a customer,
- relocate i and the next node u (if both are customers) after node j,
- relocate with inversion: same move but i and u are swapped in the reinsertion,
- interchanges: swap two strings with 1 or 2 customers, starting respectively with i and j,
- 2-opt move: reverse the string of customers from i to j included (j must be after i),

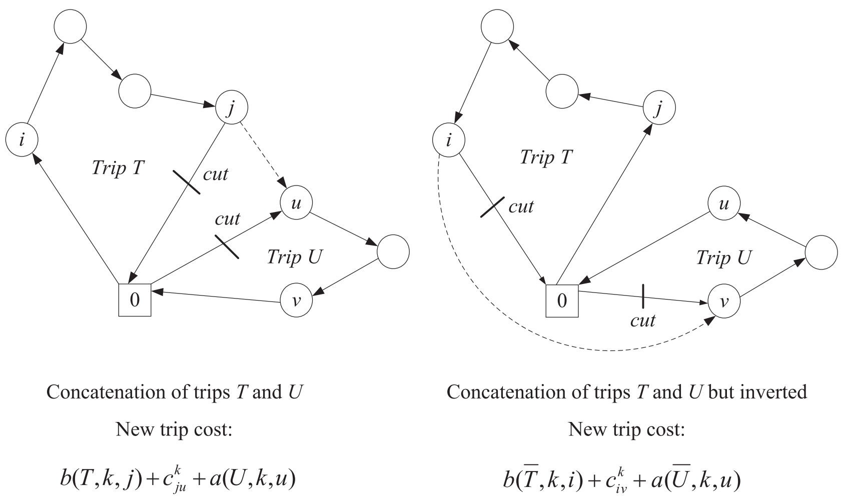

Trip T
cut
cut
Trip U
i
j
u
v
0
Concatenation of trips T and U
New trip cost:
$b(T,k,j)+c^k_{ju}+a(U,k,u)$
Concatenation of trips T and U but inverted
New trip cost:
$b(\overline{T},k,i)+c^k_{iv}+a(\overline{U},k,u)$

Fig. 1. Examples of mergers for a scenario k: (T, U) and $(\overline{T}, \overline{U})$.

- inverted 2-opt: the strings of customers before i and after j are inverted (see Fig. 2).

Algorithm 3. Local search procedure – $LS(\omega)$

```
1:  do the pre-computations for all trips
2:  repeat
3:    Initialize the lexicographic vector of the best move: $V_{best} \leftarrow V(\omega)$
4:    for each trip $T$ of $\omega$ and each node $i$ in $T$ (except end-depot) do
5:      for each trip $U$ of $\omega$ and each node $j$ in $U$ (except end-depot) do
6:        if $U = T$ then
7:          MovesOnOneTrip ($T, i, j, V_{best}, best\_move$)
8:        else
9:          MovesOnTwoTrips ($T, i, U, j, V_{best}, best\_move$)
10:       end if
11:     end for
12:   end for
13:   if $V_{best} < V(\omega)$ (improving move found) then
14:     apply $best\_move$ to the incumbent solution $\omega$
15:     update pre-computations for the modified trips (one or two)
16:   end if
17: until $V_{best} \geq V(\omega)$ (no improving move is found)
```

The strings in interchanges may have different lengths. Cost variation formulas for relocations and interchanges look like the ones for the CVRP. For instance, if customer i between nodes a and b is relocated between nodes u and v ($u \neq a$), the cost variation for scenario k is computed in $O(1)$ as $c^k_{ab} - c^k_{ai} - c^k_{ib} + c^k_{ui} + c^k_{iv} - c^k_{uv}$. 2-opt moves are more involved because the graph is directed and the cost of a node string changes when reversed. However, the pre-computations allow evaluations in constant time, as can be seen in Fig. 2. Note that the inverted 2-opt move used here is irrelevant for the CVRP, where arc costs are symmetric.

The moves scanned by *MovesOnTwoTrips* consider two given trips T and U, one node i in T, and one node j in U. They must check vehicle capacities before computing the cost variations. For instance, $load(T) - q_i + q_j \leq Q$ and $load(U) + q_i - q_j$ must hold if i and j in routes T and U are exchanged. The moves on two trips are similar to the ones on a single trip, except the 2-opt moves depicted in Fig. 3. No reversal is required for the case on the left, also called 2-opt*. The case on the right involves reversals of two strings of nodes.

The two procedures *MovesOnOneTrip* and *MovesOnTwoTrips* evaluate each move like in Algorithm 4. They compute for each scenario k the cost $newcost(k)$ of the solution generated by the move. Then the vector $newcost$ is sorted to give a lexicographic vector V_{new}.

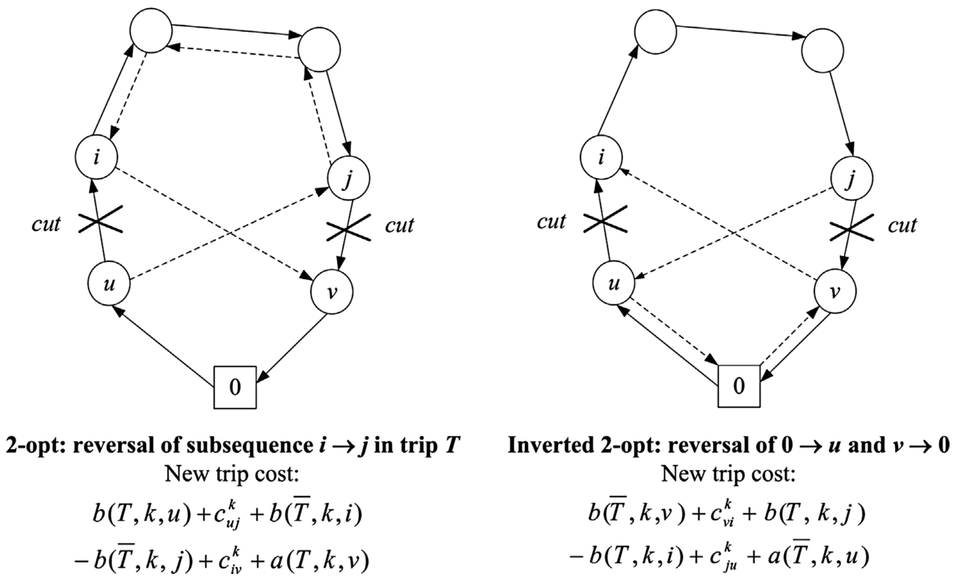
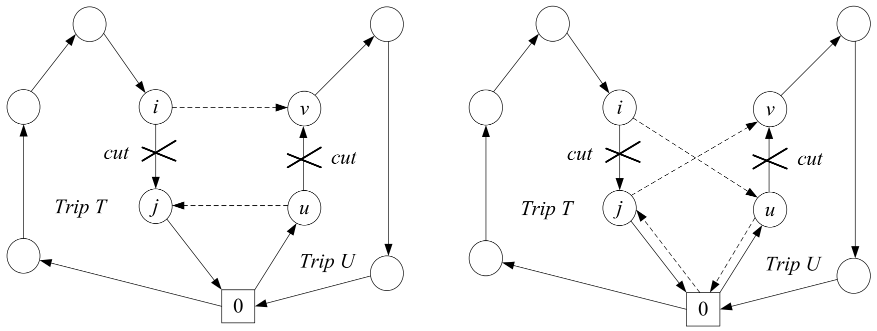

i
j
u
v
0
cut
cut
2-opt: reversal of subsequence $i \to j$ in trip T
New trip cost:
$b(T,k,u)+c^k_{uj}+b(\overline{T},k,i)$
$-b(\overline{T},k,j)+c^k_{iv}+a(T,k,v)$
Inverted 2-opt: reversal of $0 \to u$ and $v \to 0$
New trip cost:
$b(\overline{T},k,v)+c^k_{vi}+b(T,k,j)$
$-b(T,k,i)+c^k_{ju}+a(\overline{T},k,u)$

Fig. 2. 2-opt moves for one trip T and scenario k.



2-opt move on two trips – Case A

Feasibility:

$load\ (T,i) + load\ (U) - load\ (U,u) \le Q$

$load\ (U,u) + load\ (T) - load\ (T,i) \le Q$

New cost for T and U:

$$b(T,k,i) + c^k_{iv} + b(\overline{U},k,v)$$
$$+ b(U,k,u) + c^k_{uj} + a(T,k,j)$$

2-opt move on two trips – Case B

Feasibility:

$load(T,i) + load(U,u) \le Q$

$load(T) - load(T,i) + load(U) - load(U,u) \le Q$

New cost for T and U:

$$b(T,k,i) + c^k_{iu} + b(U,k,u)$$
$$+ b(U,k,u) + c^k_{uj} + a(T,k,j)$$

Fig. 3. 2-opt moves for two trips T and U and scenario k.

If this vector improves upon the best cost vector V_{best} found so far, we update V_{best} and the associated move, *best₋move*, to execute it once all moves have been evaluated.

Each local search iteration browses $O(n^2)$ moves. Thanks to the precomputations, each move can be evaluated in $O(1)$ for each scenario. However, deciding whether a move is improving or not requires a sort of the p scenario costs and a lexicographic comparison, like in the Clarke and Wright heuristic. Hence, a local search doing μ moves runs in $O(\mu n^2 p \log p)$.

In line 4 of Algorithm 4, if for one scenario the new cost exceeds the worst cost of current solution, the move is non-improving: we can drop its evaluation and proceed with the next kind of move. This avoids calling uselessly the sorting algorithm (line 6) and the lexicographic comparison (line 7). During our tests with 10–30 scenarios, this simple trick has divided the running times by 15 on average.

Algorithm 4. Evaluation of a move in *MovesOneTrip* and *MovesTwoTrips*

```
1:  if the move satisfies capacity constraints then
2:    for $k \leftarrow 1$ to $p$ do
3:      compute newcost($k$) the solution cost for scenario $k$ if the move
        were done
4:      if newcost($k$) > worst($\omega$) then exit
5:    end for
6:    sort newcost in decreasing cost order giving $V_{new}$
7:    if $V_{new} < V_{best}$ then
8:      update $V_{best}$ and the kind of move, best₋move
9:    end if
10: end if
```

7. Four local search based metaheuristics

This section describes four metaheuristics combining one of our two greedy heuristics (CW or RCW) and the local search LS: one Greedy Randomized Adaptive Search Procedure (GRASP), one Iterated Local Search (ILS), one Multi-Start ILS (MS-ILS) and another MS-ILS alternating between two search spaces, TSP tours and RVRP solutions. These metaheuristic frameworks were selected for their simple general structure and their reduced number of components and parameters. Moreover, assembling the same components into different metaheuristic structures makes easier an appraisal of the respective contributions of each structure.

7.1. Greedy randomized adaptive search procedure

With Simulated Annealing, GRASP is probably the simplest metaheuristic [20]. It consists in sampling the search space using a greedy randomized heuristic and applying local improvement to the obtained solutions. Our GRASP for the RVRP performs a fixed number of calls to the local search *LS*, *ncalls*. At the beginning, the CW heuristic described in Section 5 and the local search LS presented in Section 6 are executed to get a provisional best solution ω. This first call to a deterministic heuristic is not a standard feature of GRASP but is justified here by the good results of CW. The other iterations call the randomized Clarke and Wright heuristic RCW, then LS, and update ω if the resulting solution ω' is better. Recall that θ is a percentage defining the degree of randomness in RCW, see Section 5.

7.2. Two ILS metaheuristics

While GRASP can be viewed as a random sampling of local optima, ILS metaheuristics generate a sequence of local optima with decreasing costs, by applying a perturbation operator and a local search to a copy of the incumbent solution, see [36] for a tutorial.

Algorithm 5. Greedy Randomized Adaptive Search Procedure – *GRASP* (θ, *ncalls*, ω)

```
1: initialize the random number generator with a fixed seed for reproducibility
2: CW ($\omega$)
3: LS ($\omega$)
4: for calls $\leftarrow$ 2 to ncalls do
5:   RCW ($\theta$, $\omega'$)
6:   LS ($\omega'$)
7:   if $V(\omega') < V(\omega)$ then $\omega \leftarrow \omega'$ endif
8: endfor
```

Algorithm 6 describes a multi-start version (MS-ILS). Its main loop (lines 3–19) launches *nils* successive ILS which return their best solution ω' to update a global best solution ω (line 18). The first ILS starts from the solution computed by the Clarke and Wright heuristic CW, while the next ones are initialized using the randomized version RCW (line 4). The local search *LS* is applied to get the first local optimum (line 5). Then, each ILS performs *ncalls* iterations (lines 7–17). Each iteration takes a copy ω'' of the incumbent solution ω' and applies the perturbation procedure (*Perturb*) and the local search. The search moves to the resulting solution only in case of improvement (lines 11–12).

Algorithm 6. Multi-Start ILS – *MS-ILS* (θ, *nils*, *ncalls*, *minswaps*, *maxswaps*, ω)

```
1:  initialize the random number generator with a fixed seed for
    reproducibility
2:  initialize the components of $V(\omega)$ to $\infty$
3:  for $ils \leftarrow 1$ to $nils$ do
4:    if $ils = 1$ then $CW(\omega')$ else $RCW(\theta, \omega')$ endif
5:    $LS(\omega')$
6:    $swaps \leftarrow minswaps$
7:    for $calls \leftarrow 2$ to $ncalls$ do
8:      $\omega'' \leftarrow \omega'$
9:      $Perturb(\omega'', swaps)$
10:     $LS(\omega'')$
11:     if $V(\omega'') < V(\omega')$ then
12:       $\omega' \leftarrow \omega''$
13:       $swaps \leftarrow minswaps$
14:     else
15:       $swaps \leftarrow \min(maxswaps, swaps + 1)$
16:     end if
17:   end for
18:   if $V(\omega') < V(\omega)$ then $\omega \leftarrow \omega'$ endif
19: end for
```

The perturbation procedure *Perturb* performs a given number (*swaps*) of random exchanges of customers, among the ones respecting vehicle capacities. The perturbation level is adaptive and varies between two bounds *minswaps* and *maxswaps*. At the beginning of each ILS, the variable *swaps* is initialized to *minswaps* (line 6). At each iteration, if the solution generated via perturbation and local search does not outperform the current solution, *swaps* is incremented, but without exceeding *maxswaps*, otherwise it is reset to *minswaps*.

In Section 8 we test this MS-ILS and the more classical ILS structure obtained by setting *nils* to 1. The rationale behind MS-ILS is to restart periodically an ILS from diversified solutions instead of losing time in unproductive iterations. MS-ILS is also called GRASP×ILS since it can be viewed as a GRASP where the local search is replaced by an ILS.

7.3. Multi-start ILS with giant tours

Beasley [5] explained how to partition optimally (subject to the sequence) one TSP tour $\tau = (\tau_1, \tau_2, \ldots, \tau_n)$, called *giant tour*, into CVRP routes. The partitioning procedure called *Split* relies on an auxiliary graph $H = (Z, B)$, directed and acyclic. The node-set Z contains one dummy node 0 and one node i per customer. The arc-set B contains one arc $(i-1, j)$ if the subsequence from τ_i to τ_j (included) can make a feasible trip, i.e., if its total load fits vehicle capacity. This arc is weighted by the trip cost $c_{0,\tau_i} + \sum_{u=1}^{j-1} c_{\tau_u,\tau_{u+1}} + c_{\tau_j,0}$. An optimal splitting of τ corresponds to a minimum-cost path from node 0 to node n in graph H.

Neglected for a long time, this idea has been applied in the last decade to powerful metaheuristics for various vehicle routing problems, as shown in a recent survey [48]. We propose for the RVRP the MS-ILS with giant tours (MS-ILS-GT) of Algorithm 7, which alternates between two search spaces, giant tours and RVRP solutions.

Algorithm 7. MS-ILS with giant-tours – *MS-ILS-GT* (same header as *MS-ILS*)

```
1:  initialize the random number generator with a fixed seed for
    reproducibility
2:  initialize the components of $V(\omega)$ to $\infty$
3:  for $ils \leftarrow 1$ to $nils$ do
4:    if $ils = 1$ then $CW(\omega')$ else $RCW(\theta, \omega')$ endif
5:    $LS(\omega')$
6:    $Concat(\omega', \tau')$
7:    $swaps \leftarrow minswaps$
8:    for $calls \leftarrow 2$ to $ncalls$ do
9:      $\tau'' \leftarrow \tau'$
10:     $Perturb(\tau'', swaps)$
11:     $Split(\tau'', \omega'')$
12:     $LS(\omega'')$
13:     if $V(\omega'') < V(\omega')$ then
14:       $\omega' \leftarrow \omega''$
15:       $Concat(\omega', \tau')$
16:       $swaps \leftarrow minswaps$
17:     else
18:       $swaps \leftarrow \min(maxswaps, swaps + 1)$
19:     end if
20:   end for
21:   if $V(\omega') < V(\omega)$ then $\omega \leftarrow \omega'$ endif
22: end for
```

Each iteration of the current ILS works on a pair (ω', τ'), where ω' is an RVRP solution and τ' the giant tour obtained by concatenating its routes (without depot copies) via the *Concat* procedure. A copy τ'' of τ' is perturbed then split to give an RVRP solution ω'', improved by the local search. If ω improves ω', the latter is updated and a new giant tour τ' is deduced using *Concat*, giving a pair (ω', τ') for the next iteration. The perturbation consists in random exchanges of customers, like in ILS and MS-ILS, but applied to giant tours: no capacity check is required.

Algorithm 8 describes a *Split* procedure for the RVRP, based on the same auxiliary graph as in the CVRP but computing a shortest path in the lexicographic sense. The auxiliary graph H is not built explicitly. Each node i has a label (X^i, V^i) composed of two p-vectors. X^i_k is the cost for scenario k of the best path from node 0 to node i. The components of X^0 and V^0 are set to zero (line 1) while those of X^i and V^i, $i \neq 1$, are initialized to a large value (line 2). The two nested loops beginning lines 3 and 5 inspect each feasible route $(0, \tau_i, \tau_{i+1}, \ldots, \tau_j, 0)$ compatible with vehicle capacity and compute its total demand W and its cost L_k for each scenario k (lines 6–12). Recall that the subsequence corresponds to arc $(i-1, j)$ in H. A new label $(\bar{X}, \bar{V})$ is built for the path obtained by appending this arc to the best path for node $i-1$. To do this, we compute $\bar{X} \leftarrow X^{i-1} + L$ (line 14) and sort $\bar{X}$ in decreasing order of costs to get $\bar{V}$ (line 15). If $\bar{V} < V^j$, we have found a better path to reach node j and the label of j can be updated (lines 17–18).

Algorithm 8. Splitting procedure for one giant tour τ – *Split* (τ, ω)

```
1:  $X^0, V^0 \leftarrow (0, 0, \ldots, 0)$
2:  for $i \leftarrow 1$ to $n$ do $X^i, V^i \leftarrow (\infty, \infty, \ldots, \infty)$ end for
3:  for $i \leftarrow 1$ to $n$ do
4:    $j \leftarrow i$
5:    repeat
6:      if $i = j$ then
7:        $W \leftarrow q(\tau_i)$
8:        for $k \leftarrow 1$ to $p$ do $L_k \leftarrow c^k_{0,\tau_i} + c^k_{\tau_i,0}$ end for
9:      else
10:       $W \leftarrow W + q(\tau_j)$
11:       for $k \leftarrow 1$ to $p$ do $L_k \leftarrow L_k - c^k_{\tau_{j-1},0} + c^k_{\tau_{j-1},\tau_j} + c^k_{\tau_j,0}$ end for
12:     end if
13:     if $W < Q$ then
14:       $\bar{X} \leftarrow X^{i-1} + L$
15:       sort $\bar{X}$ in non-increasing cost order to get $\bar{V}$
16:       if $\bar{V} < V^j$ then
17:         $V^j \leftarrow \bar{V}$
18:         $X^j \leftarrow \bar{X}$
19:       end if
20:     end if
21:   until $(j > n)$ or $(W > Q)$
22: end for
```

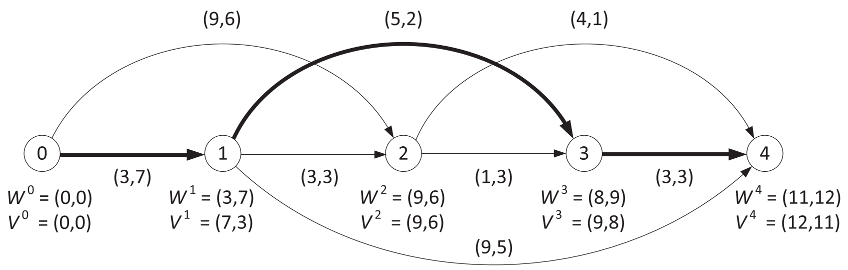


Fig. 4. Application of the splitting procedure to a giant tour with four customers.

At the end the lexicographic vector of the resulting RVRP solution ω is V^n. The solution itself can be deduced by backtracking on the shortest path, using a simple procedure described for instance in [47].

Fig. 4 gives an example for a giant tour of 4 customers with demands 3, 6, 2, 1, vehicles with capacity 10, and 2 scenarios. The arcs model feasible trips, e.g., (0, 3) does not exist because a trip with customers (1, 2, 3) would violate vehicle capacity. The values on each arc are the costs for the two scenarios. The computed labels are written under each node. The optimal lexicographic vector (12, 11) corresponds to the shortest path (0, 1, 3, 4) with bold arcs. Hence, the optimal splitting consists in three trips (0, 1, 0), (0, 2, 3, 0) and (0, 4, 0).

As *Split* inspects each feasible subsequence $(\tau_i, \tau_{i+1}, \ldots, \tau_j)$, $O(n^2)$ subsequences are tested in the worst case. As we have to sort $\bar{X}$ for each arc, *Split* runs in $O(n^2 p \log p)$. In practice, the algorithm is faster because many subsequences violate vehicle capacity.

8. Computational evaluation

8.1. Implementation and instances

Our algorithms are implemented in the Pascal-like language Delphi and executed on a Dell Precision M6600 portable PC with a 2.2 GHz Intel Core i7, 16 GB of RAM and Windows Professional. The MILP of Section 3 is translated in the modeling language GNU MathProg (a subset of AMPL) and solved using GLPK version 4.47 (www.gnu.org/software/glpk). Two sets of instances were randomly generated. The first set, to compare our heuristics with the MILP, contains 18 small instances with $n \in \{10, 15, 20\}$ customers, $m \in \{2, 3\}$ vehicles and $p \in \{10, 20, 30\}$ scenarios. The file names have an n–m–p format. All demands q_i and arc costs c_{ij}^k are integers randomly drawn in [1, 50]. Vehicle capacity Q is manually selected to load the fleet between 80 and 90% of its capacity. Note that the number of customers (10 to 20) is a typical limit to solve directly a MILP for the CVRP. The second set gathers 24 larger files with $n \in \{50, 100\}$ and $p \in \{10, 20\}$. The fleet size is $m \in \{5, 10\}$ for $n = 50$ and $m \in \{10, 20\}$ for $n = 100$. The two values of m for each problem size correspond to 5 and 10 stops per route on average. The vehicle capacity is $Q = 1,000$ while the demands are random integers drawn to use nearly 90% of fleet capacity, i.e., $q_{tot} \approx 0.9 \times mQ$, q_{tot} denoting the total demand. Each node i has its coordinates x_i and y_i randomly selected in [0, 1000]. The arc costs are travel times proportional to the Euclidean distance e_{ij}. For each arc (i, j), e_{ij} is first computed, then the integer cost c_{ij}^k for each scenario k is drawn at random in $[e_{ij}, (1 + d/100) \times e_{ij}]$, where $d \in \{10, 50, 100\}$ is a maximum deviation in percent to the baseline travel time, corresponding to three growing levels of traffic perturbation. For instance, $d = 100$ means that travel times are doubled in the worst case. The file name format is the same as in the small instances, except that the value of d is added at the end, e.g., 100–20–20–100 for $n = 100$, $m = p = 20$ and $d = 100\%$.

All randomized algorithms are executed 10 times on each instance. Although they really optimize the lexicographic min–max criterion, our experiments in the two following subsections report only the worst cost, to avoid huge tables of results with detailed cost vectors.

8.2. Results on small instances

The MILP is given a time limit of 4 h (14,400 seconds). The randomized Clarke and Wright heuristic RCW performs *ncalls* = 50 iterations with a perturbation factor $\theta = 8\%$ to randomize the savings. The four metaheuristics have the same computing budget of 5000 local optima, but 10 initial solutions are used in the multi-start versions MS-ILS and MS-ILS-GT. Hence, *ncalls* = 5000 for GRASP and ILS, while *nils* = 10 and *ncalls* = 500 for MS-ILS and MS-ILS-GT. In the two multi-start metaheuristics, RCW is also called with $\theta = 8\%$ to provide initial solutions. The perturbation level (number of customer exchanges) in the three ILS-based methods varies between *minswaps* = 1 and *maxswaps* = 3.

The detailed results are listed for each instance in Table A.1 (see Appendix A). For the MIP are given the best lower bound *LB*, the best solution cost z, the percentage gap $(z/LB - 1) \times 100$ and the running time in seconds t. Asterisks highlight proven optima while dashes mean the time limit is reached. 10 out of 18 instances are solved to optimality but no instance with $n = 20$ customers and/or $p = 30$ scenarios, except 10–3–30. For the Clarke and Wright heuristic CW, only the solution value z is given because running times are negligible (less than 0.005 s). As the other heuristics are randomized algorithms, we indicate for 10 runs the best cost found z_{min}, the average cost z_{avg} and the average time per run in seconds, t_{avg}.

These detailed results are summarized in Table 1, using five performance indicators averaged on the 18 instances: the minimum and average percentage gaps to GLPK lower bound over the 10 runs (these gaps are equal for the MILP and CW, which are tested using a single run), the standard deviation of the 10 gaps (*Std deviation*), the number of GLPK lower bounds retrieved by each heuristic (*LB hits*) and the average running time per run in seconds.

GLPK finds 10 proven optima and achieves an average solution gap close to 1% at the expense of a large running time. CW is very fast but its results are surprisingly weak, with a gap around 17%: according to our experience, this gap is rather close to 10% for the CVRP. RCW$_1$ corresponds to the solutions listed in Table A.1 for 50 iterations: the average deviation to *LB* drops to 7%. In fact, 3 iterations are enough to outperform CW on average. We added the RCW$_2$ column to show the indicators when 5000 iterations are granted: even in these conditions, RCW which has no local search cannot compete with the four metaheuristics.

The metaheuristics are very close to GLPK in terms of average gap, while being much faster. All the 10 proven optima are retrieved and three upper bounds are even improved (see instances 20–2–10, 20–2–30 and 20–3–30, with costs in boldface in Table A.1), which explains that the minimum gap is slightly improved for MS-ILS and MS-ILS-GT. Compared to ILS and MS-ILS, MS-ILS-GT lasts 50% more

Table 1
Indicators for the 18 small instances – 10 runs with 5000 calls to the local search.

Indicator	MILP	CW	RCW$_1$	RCW$_2$	GRASP	ILS	MS-ILS	MS-ILS-GT
Min gap *LB* %	1.05	16.56	7.26	3.76	1.12	1.04	0.93	0.93
Avg gap *LB* %	1.05	16.56	9.73	5.27	1.62	1.42	1.22	1.29
Std deviation %	–	–	1.47	0.81	0.30	0.30	0.20	0.20
LB hits	10	0	1	5	10	10	10	10
Time (s)	9350.40	<0.005	0.06	5.49	7.71	2.33	2.33	3.25

Table 2
Indicators for the 24 large instances – 10 runs with 5000 calls to the local search.

Indicator	CW	RCW	GRASP	ILS	MS-ILS	MS-ILS-GT
Min gap BS %	7.99	5.02	0.58	0.52	0.45	0.03
Avg gap BS %	7.99	6.22	0.88	1.28	0.93	0.20
Std deviation %	–	0.63	0.19	0.51	0.32	0.13
BS hits	0	0	6	9	9	18
Time (s)	0.10	5.66	593.20	18.63	20.51	51.28

Table 3
Indicators for the 24 large instances – Same duration as MS-ILS-GT.

Indicator	GRASP	ILS	MS-ILS	MS-ILS-GT
Min gap BS %	0.96	0.21	0.41	0.03
Avg gap BS %	1.43	0.93	0.75	0.20
Std deviation %	0.27	0.49	0.21	0.13
BS hits	3	11	10	18
Time (s)	51.3	51.3	51.3	51.3

Table 4
Friedman test for the 24 large instances – 5000 calls to the local search.

Algorithm	GRASP	ILS	MS-ILS	MS-ILS-GT
Average rank	2.79	3.52	2.65	1.04
Sum of ranks S_j	67	84.50	63.50	25.00
Sum of squared ranks	203.50	309.25	177.25	27.00
Indicator	A_2	B_2	T_2	$F_{0.99,3,69}$
Value	717.00	678.60	47.09	4.07

because of its calls to the splitting procedure. GRASP is the slowest metaheuristic, each of its 5000 calls to RCW needing $O(n^3 p \log p)$ running time.

Summarizing, the tests on small instances show that our metaheuristics compete with a direct resolution of the MILP in terms of solution quality and that ILS-based versions slightly outperform GRASP while being 2.5–3.5 times faster, using the same number of calls to the local search. However, the results of ILS-based versions are very similar and, compared with MS-ILS, the splitting procedure of MS-ILS-GT brings nothing here. The reason is that the auxiliary graph contains too few feasible paths and only 2 or 3 arcs per path.

8.3. Results for large instances

The large instances are out of reach for the MILP: even in four hours, GLPK is unable to improve the upper bound obtained by the Clarke and Wright heuristic. This is why the MILP is excluded from the tests on large instances. The parameters used are still *ncalls* = 50 for RCW, *ncalls* = 5000 for ILS, *nils* = 10 and *ncalls* = 500 for MS-ILS and MS-ILS-GT. However, we got better results using *minswaps* = 2 and *maxswaps* = 4 (instead of 1 and 3) in the perturbations applied by the three metaheuristics. The perturbation factor θ randomizing the savings in RCW was 8% for the small instances because too many iterations return identical solutions using smaller values. A 1% factor is enough on large instances.

The results are detailed in Table A.2 (see Appendix A) and summarized in Table 2. The two tables have the same format as the ones for small instances, except that the reference for the gaps is now the best solution (BS) found for each benchmark problem during our tests.

CW is still interesting for its small running time. The improvement brought by randomization in RCW is less marked than on small instances. The four metaheuristics can now be distinguished. MS-ILS-GT is the best in terms of gaps and number of best solutions found: apparently, using giant tours leads to better solutions on large instances. GRASP, MS-ILS and ILS follow in this order in terms of average gap. The statistical tests (Friedman test and pairwise tests) in the next subsection confirm with a confidence level of 1% that the best and worst methods are MS-ILS-GT and ILS (respectively) but find no statistical difference between GRASP and MS-ILS. The remarks about running times on small instances are still valid: As the number of customers is now larger, the GRASP becomes much slower than the other metaheuristics.

The metaheuristics have been evaluated up to now for 5000 local optima. As GRASP takes too much time, a natural question is to wonder whether we come to the same ranking if the same running time is allocated to each algorithm. Hence, we took the running times achieved by MS-ILS-GT in Table A.2 and allocated the same duration to GRASP, ILS and MS-ILS. The results are summarized in Table 3. Using again the average gaps, we have the ranking

Table 5
Pairwise test for the 24 large instances – 5000 calls to the local search.

$\lvert S_i - S_j\rvert$	ILS	MS-ILS	MS-ILS-GT
GRASP	<u>17.5</u>	3.5	<u>42</u>
ILS		<u>21</u>	<u>59.5</u>
MS-ILS			<u>38.5</u>

Table 6
Indicators for the 7 CVRP instances – 10 runs with 5000 calls to the local search.

Indicator	CW	RCW	GRASP	ILS	MS-ILS	MS-ILS-GT
Min gap BS %	7.30	4.61	1.24	0.76	0.33	0.10
Avg gap BS %	7.30	5.69	2.45	0.97	1.52	0.35
Std deviation %	–	0.61	0.73	0.13	0.69	0.18
BS hits	0	0	2	3	4	6
Time (s)	0.04	2.29	412.25	30.15	35.17	65.18

MS-ILS-GT < MS-ILS < ILS *s* GRASP, so this time GRASP becomes the least efficient metaheuristic. This strict ordering is also confirmed by the statistical tests, with a confidence level of 1%. Since the four algorithms are assembled from the same basic bricks (randomized heuristic RCW, perturbation procedure, local search), a first conclusion is that ILS-based structures offer a better search efficiency than GRASP for the same execution time.

We can also see that ILS, which is the less stable method, can be easily reinforced by doing multiple restarts (MS-ILS). Indeed, restarting periodically the search from a new initial solution is more profitable than prolongating uselessly a single search. Finally, adding the cyclic alternation between giant tours and complete RVRP solutions clearly brings an additional improvement. To the best of our knowledge, this is the first time that the contribution of a tour-splitting approach to a metaheuristic structure (here MS-ILS) is quantified.

8.4. Statistical tests

The optimality gap of an efficient metaheuristic follows rarely a normal distribution: The algorithm cannot find less than 0% (and this gap can be frequently achieved) then the probability increases rapidly to a maximum and then slowly decreases with a long tail. Conover [16] recommends non-parametric tests like the Friedman test, which require no assumption about the underlying probability distributions.

We detail here this test to compare our $k = 4$ randomized metaheuristics (GRASP, ILS, MS-ILS, MS-ILS-GT) on our test bed of $b = 24$ large instances, for the average costs of Table A.2 (5000 calls to the local search). Let X_{ij} denote the average cost of 10 runs achieved by algorithm i on instance j. First, compute the ranks R_{ij} of the X_{ij}, giving 1 to the best and k to the worst. Assign average ranks in case of ties, e.g., 2.5 for two heuristics with the same cost and ranks 2 and 3. Then calculate the sum of ranks $S_j = \sum_{i=1}^{b} R_{ij}$ for each heuristic j, the mean of these squared sums $B_2 = \frac{1}{b}\sum_{j=1}^{k} S_j^2$ and the sum of squared ranks $A_2 = \sum_{i=1}^{b}\sum_{j=1}^{k} R_{ij}^2$. The test statistic is given by Eq. (10).

$$T_2 = \frac{(b-1)[B_2 - bk(k+1)^2/4]}{A_2 - B_2} \tag{10}$$

The null hypothesis (all heuristics have equivalent performances) can be rejected with a confidence level α if T_2 is greater than the $1 - \alpha$ quantile of the F distribution with $k_1 = k - 1$ and $k_2 = (k-1)(b-1)$ degrees of freedom. The results are given in Table 4. As $T_2 > F_{0.99,3,69}$ we can conclude with a confidence level of 1% that the heuristics are not equivalent.

Pairwise tests must be applied to separate the heuristics. Calculate the absolute difference of the sums of ranks of metaheuristics i and j and declare i and j different if this difference exceeds the critical value CV defined in Equation (11), where $t_{1-\alpha/2}$ denotes the $1 - \alpha/2$ quantile of the t distribution with $(b-1)(k-1)$ degrees of freedom. The heuristic with the smallest sum of ranks is the best of the two.

$$|S_i - S_j| > CV = t_{1-\frac{\alpha}{2}}\sqrt{\frac{2b(A_2 - B_2)}{(b-1)(k-1)}} \tag{11}$$

In our case, we find $CV = 13.69$ and the differences in Table 5. The significant differences (underlined) indicate that (a) MS-ILS-GT dominates the other algorithms, (b) ILS is dominated by the others, (c) MS-ILS and GRASP are not significantly different.

When the same running time is allocated to all metaheuristics (results in Table 3), the statistical tests, not detailed here to save space, find the following strict ordering, from the best method to the worst: MS-ILS-GT, MS-ILS, ILS, GRASP. MS-ILS-GT is still the best algorithm but, this time, GRASP becomes the worst performer.

8.5. Performance on CVRP instances

Our algorithms were not designed for the classical (non-robust) CVRP but their performance can be easily checked by setting the number of scenarios to 1. We selected for this purpose the 14 CMT (Christofides-Mingozzi-Toth) instances (www.vrp-rep.org), which include 7 instances with $n \in \{50, 75, 100, 150, 1999\}$ and $m \in \{5, 7, 8, 10, 12, 17\}$, plus the same instances with an additional maximum route duration. As our RVRP does not consider route duration constraints, the test has been limited to the 7 unrestricted problems (instances 1 to 5, 11 and 12). The results are given in Table 6, using the same format as in Table 2 (large RVRP instances), the same conditions (5000 calls to the local search) and the same parameters. The best solutions are here the proven optima recently found by an exact method [45].

The four metaheuristics still perform well on the CVRP. The best one, MS-ILS-GT, is even able to retrieve 6 out of 7 optima. GRASP displays again the largest gaps and running time. Its longer duration comes from the $O(n^3)$ complexity of the RCW heuristic used to sample the search space. As explained in Section 5.1, a faster implementation in $O(n^2 \log n)$ is possible for the CVRP, but we made no changes in our RVRP codes. Another reason is that each perturbed solution in ILS-based methods is reoptimized in a few moves by the local search, while the independent solutions sampled by RCW in GRASP require more moves on average. Hence, the average time spent per call to the local search is greater in the GRASP.

9. Conclusion

This article has presented a pragmatic approach to compute high-quality solutions for the robust CVRP with uncertain travel costs defined by discrete scenarios. Indeed, most large cities have a traffic control center which measures and records vehicle flows in the streets. The accumulated data can be mined to provide many real scenarios. The lexicographic min–max criterion selected protects the solutions against the worst scenario, like the min–max approach, but also against the second worst scenario, the third etc. Moreover, as this criterion gives an optimal solution which is also Pareto-optimal for the multi-objective interpretation of the problem, it is likely to be preferred by decision makers.

The proposed algorithms join the very small family of metaheuristics published for robust vehicle routing problems. Ours are compact and relatively simple since they share the same components and require very few parameters. Their efficiency has been confirmed using small RVRP instances and CVRP instances with known optimal solutions. Although multiple scenarios and lexicographic comparisons increase the complexities of initial heuristics and local search moves (compared to the CVRP), the running times remain reasonable.

Another interesting conclusion is that adding an alternation between giant tours and RVRP solutions in MS-ILS brings a statistically significant improvement. So, the RVRP can be added to the growing list of routing problems solved via this approach [48].

The positive impact of the giant tour approach should be confirmed on other metaheuristics like hybrid genetic algorithms. This task will be probably more difficult because such algorithms involve more components, whose respective contribution must be isolated. Another possible direction for future research is to design an exact approach for the RVRP, but the lexicographic min–max objective function makes the task quite challenging.

Acknowledgements

Elyn Solano-Charris has been funded by a Ph.D. Research Grant from the Champagne-Ardenne Regional Council (France).

Appendix A. Detailed results for each instance

see Tables A.1 and A.2.

Table A.1
Detailed results for the 18 small instances.

File	MILP				CW	RCW			GRASP			ILS			MS-ILS			MS-ILS-GT		
n-m-p	z_{lb}	z_{ub}	Gap	t	z	z_{min}	z_{avg}	t_{avg}	z_{min}	z_{avg}	t_{avg}	z_{min}	z_{avg}	t_{avg}	z_{min}	z_{avg}	t_{avg}	z_{min}	z_{avg}	t_{avg}
10-2-10	217	*217	0.00	8.2	263	228	229.9	0.01	217	217.0	1.19	217	217.0	0.50	217	217.0	0.49	217	217.0	0.73
10-2-20	284	*284	0.00	32.2	305	285	294.0	0.01	284	284.0	2.00	284	284.0	0.91	284	284.0	0.91	284	284.0	1.16
10-2-30	301	*301	0.00	35.8	327	310	315.4	0.02	301	301.0	2.98	301	301.0	1.39	301	301.0	1.36	301	301.0	1.78
15-2-10	346	*346	0.00	313.2	412	375	384.6	0.03	346	346.0	3.78	346	346.1	1.35	346	346.0	1.32	346	346.0	1.76
15-2-20	373	*373	0.00	6560.0	420	398	404.7	0.05	373	373.0	6.35	373	373.1	2.36	373	373.1	2.27	373	373.0	3.02
15-2-30	398	404	1.51	–	483	429	439.8	0.07	404	404.2	9.52	404	404.0	3.47	404	404.1	3.36	404	404.0	4.55
20-2-10	419	423	0.95	–	523	465	480.2	0.06	**422**	428.1	9.14	**422**	424.8	2.89	**422**	423.4	2.87	**422**	425.3	3.99
20-2-20	460	470	2.17	–	535	525	531.6	0.11	475	481.1	14.93	470	476.0	4.35	470	476.1	4.26	470	476.0	5.83
20-2-30	481	501	4.16	–	580	541	558.5	0.16	501	504.8	22.48	**497**	501.1	6.64	**497**	500.3	6.70	**497**	501.0	9.06
10-3-10	255	*255	0.00	10.4	304	267	268.8	0.01	255	255.0	1.00	255	255.0	0.41	255	255.0	0.41	255	255.0	0.67
10-3-20	316	*316	0.00	24.7	356	316	324.1	0.01	316	316.0	1.78	316	316.0	0.72	316	316.0	0.71	316	316.0	0.96
10-3-30	337	*337	0.00	38.9	367	342	346.0	0.02	337	337.0	2.58	337	337.0	1.13	337	337.0	1.12	337	337.0	1.44
15-3-10	381	*381	0.00	2200.0	439	412	429.3	0.02	381	384.1	3.35	381	384.0	0.79	381	381.0	0.80	381	381.0	1.48
15-3-20	399	*399	0.00	6871.0	454	412	423.3	0.04	399	400.6	5.78	399	400.2	1.77	399	399.4	1.75	399	400.0	2.45
15-3-30	426	433	1.64	–	476	455	464.2	0.07	433	433.2	8.72	433	434.6	2.50	433	433.0	2.61	433	433.2	3.77
20-3-10	443	448	1.13	–	562	503	514.5	0.06	448	456.6	8.47	453	456.3	2.05	448	450.2	2.09	448	451.9	3.50
20-3-20	481	497	3.33	–	580	539	555.9	0.11	497	504.0	13.83	501	503.5	3.35	497	502.4	3.36	497	502.6	4.83
20-3-30	508	528	3.94	–	595	568	581.0	0.16	530	533.8	20.94	**523**	529.5	5.42	**523**	528.3	5.60	**523**	529.0	7.56

Table A.2
Detailed results for the 24 large instances.

File	BS	CWH		RCWH			ILS			MS-ILS			MS-ILS-GT		
n-m-p-d	z	z	t	z_{min}	z_{avg}	t_{avg}	z_{min}	z_{avg}	t_{avg}	z_{min}	z_{avg}	t_{avg}	z_{min}	z_{avg}	t_{avg}
50-5-10-10	8633	8986	0.02	8708	8814.9	0.89	8637	8637.0	8.77	8633	8636.2	9.04	8633	8635.0	19.34
50-5-10-50	9936	10687	0.02	10366	10519.0	0.92	9936	9947.9	8.01	9936	9948.2	8.95	9936	9936.0	21.40
50-5-10-100	11525	12544	0.02	11987	12268.9	0.95	11525	11572.8	9.61	11525	11542.6	10.02	11525	11525.0	21.83
50-5-20-10	7966	8453	0.03	8218	8288.9	1.69	7966	8089.9	8.79	7966	7972.3	9.22	7966	7966.0	22.64
50-5-20-50	10014	10609	0.03	10419	10556.1	1.76	10014	10014.0	9.12	10014	10015.0	10.24	10014	10014.0	24.93
50-5-20-100	10913	12099	0.03	11448	11607.3	1.73	10933	10979.8	9.82	10913	10933.5	10.78	10913	10923.0	25.97
50-10-10-10	10882	11469	0.02	11194	11243.0	0.83	10882	10885.2	6.86	10884	10889.7	7.76	10882	10882.0	18.26
50-10-10-50	12849	13861	0.02	13252	13471.6	0.86	12849	12909.9	7.88	12849	12859.0	8.30	12849	12849.0	19.40
50-10-10-100	16212	16852	0.02	16545	16821.3	0.85	16212	16243.0	7.95	16212	16219.8	8.39	16212	16212.0	20.04
50-10-20-10	11576	12043	0.03	11901	11988.9	1.53	11576	11679.1	8.27	11621	11630.7	8.87	11576	11590.6	21.18
50-10-20-50	14337	15089	0.03	14562	14718.4	1.58	14346	14371.0	8.01	14337	14349.0	9.07	14337	14338.5	21.82
50-10-20-100	15021	16775	0.03	15805	16067.6	1.62	15115	15131.3	10.51	15036	15105.7	10.59	15042	15076.2	27.81
100-10-10-10	12767	13535	0.13	13328	13490.1	7.30	12880	12966.7	25.30	12863	12935.1	28.55	12767	12789.7	73.35
100-10-10-50	15082	16863	0.13	16297	16622.0	7.31	15104	15454.1	27.78	15108	15345.4	32.21	15082	15084.4	81.39
100-10-10-100	18198	20973	0.13	20213	20621.8	7.39	18473	18739.5	29.84	18358	18700.3	32.96	18206	18280.7	89.97
100-10-20-10	13614	14648	0.24	14220	14345.4	13.39	13806	14004.0	22.79	13779	13835.7	31.30	13614	13663.1	81.32
100-10-20-50	15876	16938	0.26	16938	16938.0	14.07	16204	16329.9	28.03	16160	16273.0	33.39	15890	15983.2	88.83
100-10-20-100	18373	21890	0.25	20573	20786.9	14.05	18510	19017.7	32.55	18703	18848.6	35.95	18397	18457.8	95.79
100-20-10-10	22234	22925	0.12	22925	22925.0	6.48	22293	22433.7	24.78	22319	22423.0	28.29	22252	22275.1	67.09
100-20-10-50	24690	26504	0.12	26202	26416.4	6.65	24895	25137.0	25.41	24870	25077.2	28.32	24690	24745.3	72.87
100-20-10-100	29068	31711	0.13	31276	31597.5	6.92	29403	29775.4	27.70	29251	29566.1	30.72	29139	29243.3	76.33
100-20-20-10	19942	20582	0.22	20526	20575.6	12.32	19942	20008.2	26.43	19990	20082.0	32.42	19942	19959.4	73.12
100-20-20-50	24715	27086	0.23	26287	26548.0	12.39	24972	25128.2	28.53	24844	24992.1	32.77	24715	24758.9	78.45
100-20-20-100	29223	32546	0.23	31815	32186.5	12.45	29688	29833.3	29.43	29602	29813.4	34.20	29223	29467.4	87.48

References

[1] A. Agra, L.M. Hvattum, M. Christiansen, R. Figuereido, M. Poss, C. Requejo, The robust vehicle routing problem with time windows, Comput. Oper. Res. 40 (3) (2013) 856–866.
[2] H. Aissi, C. Bazgan, D. Vanderpooten, Complexity of the min–max and min–max regret assignment problems, Oper. Res. Lett. 33 (6) (2005) 634–640.
[3] H. Aissi, C. Bazgan, D. Vanderpooten, Min–max and min–max regret versions of combinatorial optimization problems: a survey, Eur. J. Oper. Res. 197 (2) (2009) 427–438.
[4] R. Baldacci, E. Bartolini, A. Mingozzi, R. Roberti, An exact solution framework for a broad class of vehicle routing problems, Comput. Manag. Sci. 7 (3) (2010) 229–269.
[5] J.E. Beasley, Route-first cluster-second methods for vehicle routing, Omega 11 (1983) 403–408.
[6] A. Ben-Tal, L. El Ghaoui, A. Nemirovski, Robust Optimization, Princeton University Press, Princeton, NJ, 2009.
[7] A. Ben-Tal, A. Nemirovski, Robust solutions to uncertain linear programs, Oper. Res. Lett. 25 (1) (1999) 1–13.
[8] D. Bertsimas, D.B. Brown, C. Caramanis, Theory and applications of robust optimization, SIAM Rev. 53 (3) (2011) 464–501.
[9] D. Bertsimas, M. Sim, Robust discrete optimization and network flows, Math. Progr. Ser. B 98 (2003) 49–71.
[10] L. Bianchi, M. Dorigo, L.M. Gambardella, W.J. Gutjahr, A survey on metaheuristics for stochastic combinatorial optimization, Nat. Comput. 8 (2) (2009) 239–287.
[11] J. Birge, F. Louveaux, Introduction to Stochastic Programming, Springer-Verlag, New York, 1997.
[12] A. Candia-Véjar, E. Alvarez-Miranda, N. Maculan, Minmax regret combinatorial optimization problems: an algorithmic perspective, RAIRO Oper. Res. 45 (2011) 101–129.
[13] E. Cao, M. Lai, H. Yang, Open vehicle routing problem with demand uncertainty and its robust strategies, Expert Syst. Appl. 41 (7) (2014) 3569–3575.
[14] L. Chen, M. Hoang Ha, A. Langevin, M. Gendreau, Optimizing road network daily maintenance operations with stochastic service and travel times, Trans. Res. E 64 (2014) 88–102.
[15] G. Clarke, J.W. Wright, Scheduling of vehicles from a central depot to a number of delivery points, Oper. Res. 12 (4) (1964) 568–581.
[16] W. Conover, Practical Nonparametric Statistics, 3rd ed., Wiley, New-York, 1999.
[17] J.F. Cordeau, M. Gendreau, G. Laporte, J.Y. Potvin, F. Semet, A guide to vehicle routing heuristics, J. Oper. Res. Soc. 53 (5) (2002) 512–522.
[18] G.B. Dantzig, J.H. Ramser, The truck dispatching problem, Manag. Sci. 6 (1) (1959) 80–91.
[19] B. Eksioglu, A.V. Vural, A. Reisman, The vehicle routing problem: a taxonomic review, Comput. Ind. Eng. 57 (4) (2009) 1452–1483.
[20] T.A. Feo, M.G.C. Resende, Greedy randomized adaptive search procedures, J. Glob. Opt. 6 (2) (1995) 109–133.
[21] R. Fukasawa, H. Longo, J. Lysgaard, M.P. Arag ao, M. Reis, E. Uchoa, R.F. Werneck, Robust branch-and-cut-and-price for the capacitated vehicle routing problem, Math. Progr. Ser. A 106 (3) (2006) 491–511.
[22] M. Gendreau, G. Laporte, R. Seguin, Stochastic vehicle routing, Eur. J. Oper. Res. 88 (1) (1996) 3–12.
[23] B. Golden, S. Raghavan, E. Wasil, The Vehicle Routing Problem: Latest Advances and New Challenges, Springer, New York, 2008.
[24] A. Gómez, R. Mari no, R. Akhavan-Tabatabaei, A. Medaglia, On modeling stochastic travel and service times in vehicle routing, Transp. Sci. (2015) (in press).
[25] C.E. Gounaris, W. Wiesemann, C.A. Floudas, The robust capacitated vehicle routing problem under demand uncertainty, Oper. Res. 61 (3) (2013) 677–693.
[26] J. Han, C. Lee, S. Park, A robust scenario approach for the vehicle routing problem with uncertain travel times, Transp. Sci. 63 (2013) 1294–1306.
[27] R. Kalaï, C. Lamboray, D. Vanderpooten, Lexicographic α-robustness: an alternative to min–max criteria, Eur. J. Oper. Res. 220 (3) (2012) 722–728.
[28] A. Kasperski, A. Kurpisz, P. Zieliński, Approximating a two-machine flow shop scheduling under discrete scenario uncertainty, Eur. J. Oper. Res. 217 (2012) 36–43.
[29] A.S. Kenyon, D.P. Morton, Stochastic vehicle routing with random travel times, Transp. Sci. 37 (1) (2003) 69–82.
[30] P. Kouvelis, G. Yu, Robust Discrete Optimization and Its Applications, Kluwer Academic Publishers, Norwell, MA, 1997.
[31] R.V. Kulkarni, P.R. Bhave, Integer programming formulations for vehicle routing problems, Eur. J. Oper. Res. 20 (1) (1985) 58–67.
[32] G. Laporte, Fifty years of vehicle routing, Transp. Sci. 43 (4) (2009) 408–416.
[33] G. Laporte, M. Gendreau, J.Y. Potvin, F. Semet, Classical and modern heuristics for the vehicle routing problem, Int. Trans. Oper. Res. 7 (4-5) (2000) 285–300.
[34] C. Lecluyse, T. Van Woensel, H. Peremans, Vehicle routing with stochastic time-dependent travel times, 4OR: A Q. J. Oper. Res. 7 (2009) 363–377.
[35] C. Lee, K. Lee, S. Park, Robust vehicle routing problem with deadlines and travel time/demand uncertainty, J. OR Soc. 63 (2012) 1294–1306.
[36] H.R. Lourenço, O.C. Martin, T. Stützle, Iterated local search: framework and applications, in: M. Gendreau, J.Y. Potvin (Eds.), Handbook of Metaheuristics, Springer, 2010, pp. 363–397.
[37] V. Marques, F. Gomide, Parameter control of metaheuristics with genetic fuzzy systems, Evol. Intell. 4 (2011) 183–202.
[38] D. Mester, O. Bräysy, Active-guided evolution strategies for large-scale capacitated vehicle routing problems, Comput. Oper. Res. 34 (10) (2007) 2964–2975.
[39] C.E. Miller, A.W. Tucker, R.A. Zemlin, Integer programming formulations and traveling salesman problems, J. ACM 7 (4) (1960) 326–329.
[40] B.F. Moghaddam, R. Ruiz, S.J. Sadjadi, Vehicle routing problem with uncertain demands: an advanced particle swarm algorithm, Comput. Ind. Eng. 62 (1) (2012) 306–317.
[41] Y. Nikulin, Simulated annealing algorithm for the robust spanning tree problem, J. Heurist. 14 (4) (2008) 391–402.
[42] M. Noorizadegan, L. Galli, B. Chen, On the heterogeneous vehicle routing problem under demand uncertainty., in: 21st International Symposium on Mathematical Programming, 2012, pp. 1–25.
[43] F. Ordóñez, Robust vehicle routing, INFORMS Tutor. Oper. Res. 7 (2010) 153–178.
[44] W. Orgryczak, On the lexicographic minimax approach to location problems, Eur. J. Oper. Res. 100 (3) (1997) 566–585.
[45] D. Pecin, M. Poggi, A. Pessoa, E. Uchoa, Improved branch-cut-and-price for capacitated vehicle routing, Oper. Res. (2015) (in press).
[46] D. Pisinger, S. Röpke, A general heuristic for vehicle routing problems, Comput. Oper. Res. 34 (8) (2007) 2403–2435.
[47] C. Prins, A simple and effective evolutionary algorithm for the vehicle routing problem, Comput. Oper. Res. 31 (12) (2004) 1985–2002.
[48] C. Prins, P. Lacomme, C. Prodhon, Order-first split-second methods for vehicle routing problems: a review, Transp. Res. C 40 (2014) 179–200.
[49] I. Sungur, F. Ordóñez, M. Dessouky, A robust optimization approach for the CVRP with demand uncertainty, IIE Trans. 40 (5) (2008) 509–523.
[50] D. Taş, N. Dellaert, T. van Woensel, T. de Kok, Vehicle routing problem with stochastic travel times including soft time windows and service costs, Comput. Oper. Res. 40 (1) (2013) 214–224.
[51] N.E. Toklu, R. Montemanni, L.M. Gambardella, An ant colony system for the capacitated vehicle routing problem with uncertain travel costs, in: IEEE Symposium on Swarm Intelligence (SIS), 2013, pp. 32–39.
[52] N.E. Toklu, R. Montemanni, L.M. Gambardella, A robust multiple ant colony system for the capacitated vehicle routing problem, in: IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2013, pp. 1871–1876.
[53] P. Toth, D. Vigo, Vehicle routing: Problems, Methods and Applications, 2nd ed., Philadelphia, SIAM, 2014.
[54] T. Vidal, T.G. Crainic, M. Gendreau, N. Lahrichi, W. Rei, A hybrid genetic algorithm for multi-depot and periodic vehicle routing problems, Oper. Res. 60 (3) (2012) 611–624.
[55] T. Vidal, T.G. Crainic, M. Gendreau, C. Prins, Heuristics for multi-attribute vehicle routing problems: a survey and synthesis, Eur. J. Oper. Res. 231 (1) (2013) 1–21.
[56] T. Vidal, T.G. Crainic, M. Gendreau, C. Prins, A unified solution framework for multi-attribute vehicle routing problems, Eur. J. Oper. Res. 234 (3) (2014) 658–673.
[57] A. Wald, Contributions to the theory of statistical estimation and testing hypotheses, Ann. Math. Stat. 10 (3) (1939) 299–326.
[58] G. Yu, J. Yang, On the robust shortest path problem, Comput. Oper. Res. 25 (6) (1998) 457–468.