

The Vehicle Routing Problem with Service Level Constraints

Teobaldo Bulhões *

Instituto de Computação, Universidade Federal Fluminense, Niterói, Brazil

tbulhoes@ic.uff.br

Minh Hoàng Hà

FPT Technology Research Institute, FPT University, Hanoi, Viet Nam

hoanghm@fpt.edu.vn

Rafael Martinelli

Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro, Brazil

martinelli@puc-rio.br

Thibaut Vidal

Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Brazil

vidalt@inf.puc-rio.br

Abstract. We consider a vehicle routing problem which seeks to minimize cost subject to service level constraints on several groups of deliveries. This problem captures some essential challenges faced by a logistics provider which operates transportation services for a limited number of partners and should respect contractual obligations on service levels. The problem also generalizes several important classes of vehicle routing problems with profits. To solve it, we propose a compact mathematical formulation, a branch-and-price algorithm, and a hybrid genetic algorithm with population management, which relies on problem-tailored solution representation, crossover and local search operators, as well as an adaptive penalization mechanism establishing a good balance between service levels and costs. Our computational experiments show that the proposed heuristic returns very high-quality solutions for this difficult problem, matches all optimal solutions found for small and medium-scale benchmark instances, and improves upon existing algorithms for two important special cases: the vehicle routing problem with private fleet and common carrier, and the capacitated profitable tour problem. The branch-and-price algorithm also produces new optimal solutions for all three problems.

Keywords. Routing, collaborative logistics, service level constraints, integer programming, genetic algorithms.

1. Introduction

In a recent industrial application, the authors have been confronted with a complex variant of the vehicle routing problem (VRP) which received, until now, only limited attention in the academic literature. We take the viewpoint of a third-party logistics provider (3PL), which operates long-haul transportation services for a number of business partners. The company operates on a planning horizon and delivers products to various delivery locations as requested by the partners a few days in advance. A strict delivery deadline, in the form of a last possible delivery day, is set for each transportation request. The company has established agreements with each partner specifying, among others, a minimum level of on-time deliveries for its group of requests.

The efficient management of collaborative logistics has stimulated a rich set of studies over the years [34, 42, 55]. In our case, even in the presence of precise information on delivery requests, the company faces the following optimization challenge: because of limited available resources, i.e., fleet size and vehicle capacity, all services cannot be realistically fulfilled. It is necessary to select a subset of deliveries and determine cost-efficient vehicle routes in such a way that the overall activity is profitable and that the agreements with the partners are respected. A natural strategy of the company consists in attempting to fulfill the service levels on a rolling horizon with some safety margin, hence ensuring overall satisfaction of contractual clauses on a larger time period (e.g., one month of activity). As such, the firm seeks to balance quality of service and operational costs, subject to a minimum threshold on quality of service for some groups of requests. These group requirements create linking constraints between the service selection decisions, which need to be carefully considered during routing optimization.

We now introduce a simple and deterministic variant of the VRP which captures some essential decisions in this situation. The study of this simplified problem will help to identify key properties and methods. The VRP with service levels (VRP-SL) can be formulated as follows. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a complete undirected graph with $|\mathcal{V}| = n + 1$ nodes. The node $v_0 \in \mathcal{V}$ represents a depot, where a fleet of m identical vehicles is based. Each other node v_i for $i \in \{1, \dots, n\}$ represents a customer, associated with a demand q_i , a profit p_i , and a service weight s_i which represents its relative importance in the group service level constraint.

The set of customers is distributed into K subsets: $\mathcal{V} - \{v_0\} = \bigcup_{k=1, \dots, K} \mathcal{V}_k$, such that $\mathcal{V}_k \cap \mathcal{V}_{k'} = \emptyset$ for any $k \neq k'$. Each subset represents the deliveries of one partner and is associated with a requested service level α_k . Any edge $(i, j) \in \mathcal{E}$ represents a possible trip between a node $v_i \in \mathcal{V}$ and a node

$v_j \in \mathcal{V}$ with a distance cost d_{ij} . The goal of the VRP-SL is to find *up to* m vehicle routes starting and ending at the depot, such that

- each customer is serviced at most one time,
- the total demand quantity of any route does not exceed a vehicle capacity Q ,
- the service level of each group k is attained, i.e., the total service weight of the deliveries to this group reaches $\alpha_k \sum_{v_i \in \mathcal{V}_k} s_i$, and
- the sum of travel costs and lost profits is minimized.

This problem belongs to the wide class of vehicle routing problems with profits, which also includes the team orienteering problem (TOP), the profitable VRP (VRPP), the VRP with private fleet and common carrier (VRPPFCC) and the capacitated profitable tour problem (CPTP). Interestingly, as highlighted in Section 2, this problem fills a gap in the literature, since most known multi-vehicle problems with customer selections either aim to maximize service levels subject to distance constraints (TOP) or seek a weighted optimization of distance and service levels, through penalties for outsourcing or lost profits (VRPPFCC and CPTP). To this date, very few works on deterministic settings [58, 68, 43] have addressed multi-vehicle routing optimization subject to a service level (SL) constraint. Finally, the VRP-SL is finally a natural extension of the generalized VRP (GVRP, see [29, 5, 10, 31]), which also models a rich set of VRP applications.

To address the VRP-SL, we introduce a compact ILP formulation and a branch-and-price algorithm which can solve to optimality small- and medium-scale instances, as well as a hybrid population metaheuristic inspired by the Unified Hybrid Genetic Search (UHGS) framework of [62, 64]. Previous applications of UHGS have led to efficient algorithms for several VRPs, including the GVRP, TOP, VRPP and VRPPFCC, among others [64, 66]. However, UHGS relies heavily on the fact that problem objectives and constraints are either cumulated on or separately applied to each route, hence allowing to optimize customer-selection decisions independently via a dedicated route evaluation operator. This decomposition does not apply to the VRP-SL without a Lagrangian relaxation of the group constraints, which would make it impossible to find the optimal solution in some cases and impede the overall method performance. Because of these characteristics, we had to revise most of the operators of the method while keeping the general principles. We thus use a two-chromosome solution representation, which contains a service level chromosome and a giant-tour chromosome. We derive a new crossover, use dedicated local search moves as well as a penalization strategy to represent, optimize and inherit customer-selection decisions. The contributions of this article are the following:

1. We introduce the VRP-SL, a rich VRP connected with important applications in collaborative logistics, generalizing many classes of routing problems with profits.
2. We propose a compact mathematical formulation for the problem, a branch-and-price algorithm, as well as an efficient hybrid genetic search (HGS) which exploits problem-tailored selection representation, crossover, local searches, and penalty management operators to find a good balance between service levels and costs.
3. We conduct extensive computational experiments to evaluate the performance of the proposed methods on new benchmark instances for the VRP-SL as well as classical benchmark instances for the VRPPFCC and CPTP. The proposed HGS finds all known optimal solutions for the considered problems, outperforms previous methods for the VRPPFCC and CPTP, and generates solutions of consistent quality on large instances. Several solutions of the VRPPFCC and CPTP are also proven optimal for the first time by the branch-and-price algorithm.

The remaining parts of the article are organized as follows. Section 2 reviews the related literature. Section 3 presents some mathematical formulations for the problem. Sections 4 and 5 describe the branch-and-price and the hybrid genetic algorithm, respectively. Section 6 reports the experimental analyses, and Section 7 concludes.

2. Related literature and subproblems

Vehicle routing problems with profits are the subject of an extensive literature, surveyed in [3, 23, 60, 63]. Generally, one seeks to jointly minimize cost and maximize customer’s service levels, two objectives which are conflicting when the profits do not render all deliveries profitable, or in the presence of additional vehicle constraints (e.g., distance or capacity limits). The related literature can thus be classified according to three fundamental solution techniques for this bi-objective problem:

- I) **Weighted sum** – minimizing a weighted difference of costs and service levels;
- II) **Constraints on cost/distance** – maximizing service levels subject to distance constraints (independently for each route);
- III) **Constraints on service levels** – minimizing distance subject to service level constraints.

Table 1 presents an overview of these main classes of methods, for the single and multi-vehicle routing problems with profits (variants of the TSP and VRP, respectively). Objectives I and II are the subjects of a vast literature, which includes a very large variety of exact methods and metaheuristics. The TOP, in particular, seeks the maximization of service levels subject to distance constraints, and has been studied in dozens of articles in the past decade (see [1, 2, 13, 19, 22, 37, 38, 40, 44, 54, 59, 66], among others). In contrast, a larger methodological gap remains for objective III, which aims to

	Min $\alpha \times \text{cost} - \beta \times \text{service level}$	Max service level s.t. cost constraints	Min cost s.t. service level constraints
	<p>Ib) Profitable tour problem (PTP)</p> <p><i>No dedicated heuristics, but:</i> Reductions to elementary shortest path with possible negative cycles, or asymmetric TSP Various approximation algorithms [3] Extensions into PCTSP with profits considered in the objective</p> <p>Exact: Branch-and-cut (+ capacity constraint) [35] Bounding method via Lagrangian relax. [20]</p>	<p>IIb) Orienteering problem (OP)</p> <p>Heuristic: GRASP [14] Guided local search [59] Ant colony optimization and VNS [53] Tabu search and others... [28]</p> <p>Exact: Branch-and-cut [24] Branch-and-cut + Lagrangian relaxation [36]</p>	<p>IIIb) Prize-collecting TSP (PCTSP)</p> <p>Heuristic: Tabu search using GENIUS neighborhood [49] Clustering search [15] Lagrangian heuristic [20] Tabu Search (steel industry) [45]</p> <p>Exact: Branch-and-cut [11] Bounding method via Lagrangian relax. [20] Formulation + Additive bounds [25]</p>
T			
S			
P			
	<p>Ia) VRPPFCC and CPTP</p> <p>Heuristic VRPPFCC: Hybrid GA with large neighborhoods [66] Adaptive variable neighborhood search [32] Adaptive variable neighborhood search [57] Tabu with possible ejection chains [18, 51] and others...</p> <p>Heuristic CPTP: VNS and tabu search [2]</p> <p>Exact: Branch-and-price [1, 2]</p>	<p>IIa) Team-orienteering problem (TOP)</p> <p>Heuristic: Pareto mimic algorithm [38] Hybrid GA with large neighborhoods [66] Multi-start simulated annealing [44] Augmented large neighborhood search [40] Particle swarm optimization [19] Path relinking [54] Guided local search [59] Ant colony optimization and others... [37]</p> <p>Exact: Branch-and-price [1, 2, 13] Branch-and-cut-and-price [39] Branch-and-cut [22]</p>	<p>IIIa) Prize collecting VRP (PCVRP)</p> <p><i>Limited number of works on this variant</i></p> <p>Heuristic: Adaptive VNS for a related problem with a minimum delivery level [56] Guided local search (steel industry) [68] Iterated local search (steel industry) [58] Two-stage self-adaptive VNS [43]</p> <p>Exact: None to our knowledge</p>
V			
R			
P			

Table 1: Synthesis of state-of-the-art heuristics and exact approaches for single- and multi-vehicle routing problems with profits.

minimize travel cost subject to service level constraints. The multi-vehicle case of this objective has been mostly studied in the context of hot strip mill scheduling for steel production [68, 69]. More recently, [43] investigated a special case with one group of customers (all of them) and one service level constraint, usually referred to as the prize-collecting VRP (PCVRP). The authors proposed a self-adaptive VNS and reported computational experiments on new instances. These instances, however, are now unavailable. The VRP-SL generalizes this problem by introducing multiple groups and associating profits to deliveries. By doing so, the problem remains concise and simple to define, but generalizes many VRP classes:

- The PCTSP and PCVRP correspond to VRP-SL instances with a single group.
- The CPTP, PTP and VRPPFCC all correspond to instances with 0% service levels.
- The CVRP corresponds to instances with 100% service levels.
- The generalized VRP (GVRP) can be reduced to the VRP-SL by imposing a small service level for each group, hence forcing at least one delivery.
- Any instance of the periodic vehicle routing problem (PVRP) such that, for each customer i , any combination of f_i different days in a set D_i of available days is feasible, can be transformed into a VRP-SL instance with $\sum_i |D_i|$ customers. This is done by duplicating each customer i into as many vertices as possible visit days, defining a group for the resulting visits with a service level $\alpha_i = f_i/|D_i|$, and setting a large cost M for the edges that link two vertices from different days. This reduction follows the same principles as the reduction from PVRP instances with frequency $f_i = 1$ into GVRP instances, discussed in [5], but is more general as it allows to deal with frequency values greater than 1.

Finally, since the name *prize-collecting VRP* is not consistently used in the literature and far from self-explanatory, we opted for the name *VRP with service levels* (VRP-SL) for the proposed problem with several groups, which is clearer and more distinctive.

3. Mathematical Formulations

Compact formulation. We first propose a mixed integer linear formulation of the VRP-SL. Our mathematical model is based on the two-commodity flow formulation of [7], which has already been extended to several closely related VRP variants in [30, 31]. The graph \mathcal{G} is first extended into $\bar{\mathcal{G}} = (\bar{\mathcal{V}}, \bar{\mathcal{E}})$ by adding a new vertex v_{n+1} , representing a copy of the depot. We thus define $\bar{\mathcal{V}} = \mathcal{V} \cup \{v_{n+1}\}$, $\mathcal{V}' = \bar{\mathcal{V}} \setminus \{v_0, v_{n+1}\}$, $\bar{\mathcal{E}} = \mathcal{E} \cup \{(v_i, v_{n+1}), v_i \in \mathcal{V}'\}$, and $d_{i,n+1} = d_{0,i}$ for all $v_i \in \mathcal{V}'$. For each edge $(v_i, v_j) \in \bar{\mathcal{E}}$, we define a binary variable x_{ij} , set to 1 if and only if a vehicle travels on this edge, as well as two flow variables f_{ij} and f_{ji} . When a vehicle travels from v_i to v_j , the flow f_{ij}

represents the current load in the vehicle, and the flow f_{ji} represents the residual capacity of the vehicle ($f_{ji} = Q - f_{ij}$). Finally, y_i is a binary variable which is set to 1 if and only if $v_i \in \mathcal{V} \setminus \{v_0\}$ is serviced. The VRP-SL can be formulated as:

$$\text{Minimize} \quad \sum_{(v_i, v_j) \in \bar{\mathcal{E}}} d_{ij} x_{ij} + \sum_{v_i \in \mathcal{V} \setminus \{v_0\}} p_i (1 - y_i) \quad (1)$$

$$\text{Subject to} \quad \sum_{v_i \in \mathcal{V}_k} s_i y_i \geq \alpha_k \sum_{v_i \in \mathcal{V}_k} s_i \quad k \in \{1, \dots, K\} \quad (2)$$

$$\sum_{v_i \in \bar{\mathcal{V}}, i < k} x_{ik} + \sum_{v_j \in \bar{\mathcal{V}}, j > k} x_{kj} = 2y_k \quad v_k \in \mathcal{V}' \quad (3)$$

$$\sum_{v_j \in \bar{\mathcal{V}}} (f_{ji} - f_{ij}) = 2q_i y_i \quad v_i \in \mathcal{V}' \quad (4)$$

$$\sum_{v_j \in \mathcal{V}'} f_{0j} = \sum_{v_i \in \mathcal{V}'} q_i y_i \quad (5)$$

$$\sum_{v_j \in \mathcal{V}'} f_{n+1j} = zQ \quad (6)$$

$$f_{ij} + f_{ji} = Qx_{ij} \quad (v_i, v_j) \in \bar{\mathcal{E}} \quad (7)$$

$$x_{ij} \in \{0, 1\} \quad (v_i, v_j) \in \mathcal{E} \quad (8)$$

$$y_i \in \{0, 1\} \quad v_i \in \mathcal{V} \setminus \{v_0\} \quad (9)$$

$$f_{ij} \geq 0, f_{ji} \geq 0 \quad (v_i, v_j) \in \bar{\mathcal{E}} \quad (10)$$

$$z \leq m, z \in \mathbb{N} \quad (11)$$

The objective of Equation (1) aims to minimize the sum of transportation costs and lost profits. With this objective, the optimal solution value is always non-negative (no negative terms). Moreover, the optimal CVRP solution is a feasible solution of this model, with a cost greater or equal to the VRP-SL optimum. Constraints (2) impose the service levels for each group. Constraints (3) ensure that each vertex of $\mathcal{V} \setminus \{v_0\}$ is visited at most once. These constraints also connect the path variables (x_i) to the customer selection variables (y_i) in order to evaluate the profits. Constraints (4)–(7) define a feasible two-commodity flow between the source v_0 and the sink v_{n+1} . Specifically, Constraints (4) state that the inflow minus the outflow at each vertex $v_i \in \mathcal{V}'$ is equal to $2q_i$ if v_i is used, and 0 otherwise. The outflow at the source vertex v_0 , computed in Constraint (5), is set to the total demand of the vertices that are serviced in the solution, and the outflow at the sink v_{n+1} , calculated in Constraint (6), corresponds to the total capacity of the vehicle fleet. Finally,

Constraints (7) establish the link between the flows f_{ij} and f_{ji} , and Constraint (11) sets a bound on the number of vehicles.

The linear relaxation of the VRP-SL can be strengthened via some simple valid inequalities. If s_i is integer for all $v_i \in \mathcal{V} \setminus \{v_0\}$, then the service level constraints (2) can be transformed into:

$$\sum_{v_i \in \mathcal{V}_k} s_i y_i \geq \left\lceil \alpha_k \sum_{v_i \in \mathcal{V}_k} s_i \right\rceil \quad k \in \{1, \dots, K\}. \quad (12)$$

Moreover, the following flow inequalities from [7, Equation 64] are used:

$$f_{ij} \geq q_j x_{ij} \text{ and } f_{ji} \geq q_i x_{ij} \quad i, j \neq v_0 \text{ and } i, j \neq v_{n+1}. \quad (13)$$

The first inequality explicitly forces f_{ij} to contain the delivery quantity q_j for the next customer, and the second inequality forces the residual capacity f_{ji} to be greater than the delivery quantity q_i of the previous customer. Finally, for each group $k \in \{1, \dots, K\}$, we define Z_k as the minimum load quantity which allows to satisfy the service level constraint:

$$Z_k = \min_{y_i \in \{0,1\}} \left\{ \sum_{v_i \in \mathcal{V}_k} q_i y_i \mid \sum_{v_i \in \mathcal{V}_k} s_i y_i \geq \alpha_k \sum_{v_i \in \mathcal{V}_k} s_i \right\} \quad (14)$$

These values can be evaluated in pseudo-polynomial time. Then, the following capacity cut is valid for each group k :

$$\sum_{v_i \in \mathcal{V}_k, v_j \in \mathcal{V} \setminus \mathcal{V}_k} x_{ij} \geq 2 \left\lceil \frac{Z_k}{Q} \right\rceil \quad k \in \{1, \dots, K\}. \quad (15)$$

Note that we also investigated in preliminary experiments a similar formulation based on one-commodity flows [27] for directed graphs, hence associating two variables x_{ij} and x_{ji} for each edge (v_i, v_j) . Although both formulations should produce similar bounds [41], the suggested two-commodity flow formulation of the VRP-SL led to overall better results in our context, possibly due to the smaller number of binary variables.

Set Partitioning Formulation. We now present a set-partitioning based formulation of the VRP-SL, which corresponds to a Dantzig-Wolfe decomposition of the model of Equations (1)–(11). Similar formulations have been successfully used in the VRP literature in the past years to obtain stronger

linear relaxations [6, 17, 26, 47]. The drawback of this formulation comes from its exponential number of variables, which must be tackled using a column generation algorithm.

Let Ω be the set of all feasible routes for the problem. A route $r \in \Omega$ is a closed walk starting and ending at the depot, visiting a set of customers only once and respecting the vehicle's capacity (this definition will be extended afterwards to allow visiting a customer more than once). Then for each route, we define a binary variable λ_r indicating whether the route r is used in the solution. The resulting formulation is as follows:

$$\text{Minimize} \quad \sum_{r \in \Omega} c_r \lambda_r + \sum_{v_i \in \mathcal{V}'} p_i (1 - y_i) \quad (16)$$

$$\text{Subject to} \quad \sum_{v_i \in \mathcal{V}_k} s_i y_i \geq \alpha_k \sum_{v_i \in \mathcal{V}_k} s_i \quad k \in \{1, \dots, K\} \quad (17)$$

$$\sum_{r \in \Omega} a_i^r \lambda_r = y_i \quad v_i \in \mathcal{V}' \quad (18)$$

$$\sum_{r \in \Omega} \lambda_r \leq m \quad (19)$$

$$\lambda_r \in \{0, 1\} \quad r \in \Omega \quad (20)$$

$$y_i \in \{0, 1\} \quad v_i \in \mathcal{V}' \quad (21)$$

The objective function (16) minimizes the total cost of the active routes plus the lost profits. Constraints (17) are the same as Constraints (2). In Constraints (18), a_i^r (boolean) represents the number of times route r visits customer i . Then each constraint forces one of the routes which visits customer i to be active if variable y_i is set to 1. Constraints (19) limit the number of active routes to the number of available vehicles.

4. Branch-and-price algorithm

The set partitioning formulation has an exponential number of variables. To circumvent this issue, we use a column generation algorithm which starts with no variables and solves a pricing sub-problem to generate new ones. For the formulation presented in the last section, the pricing sub-problem is an Elementary Shortest Path Problem with Resource Constraints (ESPPRC). Ideally, one would want to price only elementary routes, but since the ESPPRC is known to be strongly NP-hard [21], we solve a pseudo-polynomial relaxation, the Shortest Path Problem with Resource Constraints (SPPRC) [16], allowing the routes to visit a given customer more than once. For both problems, the objective is to find a route with negative reduced cost. Given the dual variables β_i

and γ associated to Constraints (18) and (19), a constant b_{ij}^r representing the number of times route r traverses edge (v_i, v_j) and setting $\beta_0 = \gamma$, the original reduced cost of a route and its reformulation as a function of the edges are stated in Equations (22) and (23):

$$\bar{c}_r = c_r - \gamma - \sum_{i \in \mathcal{V}'} a_i^r \beta_i \quad (22)$$

$$\bar{c}_r = \sum_{(v_i, v_j) \in \mathcal{E}} b_{ij}^r \left(c_{ij} - \frac{\beta_i + \beta_j}{2} \right). \quad (23)$$

When relaxing the ESPPRC into the SPPRC, the bounds obtained by column generation deteriorate in most cases. Some alternative relaxations have thus been proposed to find a better balance between efficiency and solution quality [16, 33, 46]. We use the *ng*-route relaxation [8], which has been successfully applied to multiple VRP variants in the last years. For each vertex i , a set $NG_i \subseteq \mathcal{V}'$ is defined to represent the “memory” of i . These NG_i sets usually contain a subset of vertices closest from i . The pricing sub-problem is solved by a forward dynamic programming algorithm, which maintains a memory of past visits to prohibit some vertices, but uses the NG_i sets to reduce this memory size and thus the size of the state space. During the algorithm, each path P has an associated label $\mathcal{L}(P)$ containing the last customer visited $v(P)$, the total reduced cost $\bar{c}(P)$, the current load $q(P)$ and the customers which have been visited and *remembered* $\Pi(P)$. When extending a path from customer $v(P)$ to customer v_j , the extension is only allowed if $j \notin \Pi(P)$, and the label for the new path P' can be obtained as:

$$\mathcal{L}(P') = (v_j, \bar{c}(P) + \bar{c}_{ij}, q(P) + q_{v_j}, \Pi(P) \cap NG_j \cup \{v_j\}) \quad (24)$$

Lastly, we use a dominance rule to fathom labels which cannot lead to an optimal solution. A label $\mathcal{L}(P_1)$ dominates a label $\mathcal{L}(P_2)$ if

$$\{v(P_1) = v(P_2)\} \wedge \{\bar{c}(P_1) \leq \bar{c}(P_2)\} \wedge \{q(P_1) \leq q(P_2)\} \wedge \{\Pi(P_1) \subseteq \Pi(P_2)\}.$$

Other techniques can be used to further improve the overall column generation efficiency. We use two approaches. The first one is a simple heuristic pricing. This algorithm stores only the label with best reduced cost for each customer v_i and load q during the dynamic programming. The second approach is dual stabilization [50]. We use a parameter $\alpha \in [0, 1[$, as shown in Equations (25)–(26), to avoid a large variation in the values of the dual variables between two iterations of the column generation. Our algorithm starts with $\alpha = 0.9$ and reduces the value of α by 0.1 each time

the pricing sub-problem returns an invalid route.

$$\gamma = \alpha\gamma^{k-1} + (1 - \alpha)\gamma^k \quad (25)$$

$$\beta_i = \alpha\beta_i^{k-1} + (1 - \alpha)\beta_i^k \quad \forall v_i \in \mathcal{V}' \quad (26)$$

Finally, to improve the bounds obtained by the column generation algorithm, we embedded it into a branch-and-bound (B&B) procedure. At each node of the branch-and-bound tree, the column generation algorithm is called to obtain the solution of the linear relaxation, taking into account possible fixed variables due to branching. This algorithm framework is usually called branch-and-price (B&P). A key difference with classic B&B algorithms relates to the fact that it is not possible, in our context, to branch on the λ_r variables, since a fixing of $\lambda_r = 0$ would result in repricing the variable. To overcome this difficulty, the B&P branches on the original x_{ij} variables as well as the y_i variables, choosing at each node on the tree the most fractional variable to branch, but always giving priority to y_i variables. Fixing a y_i variable is straightforward. The x_{ij} variables, however, are not explicitly present in the formulation, and thus we add Equation (27) to the set partitioning formulation for each fixed edge. In this equation, b_{ij}^r represents the number of times route r traverses edge (v_i, v_j) , and \bar{x}_{ij} is the value which should be fixed for x_{ij} .

$$\sum_{r \in \Omega} b_{ij}^r \lambda_r = \bar{x}_{ij} \quad (27)$$

The presence of Equation (27) introduces a new dual variable ρ_{ij} which must be considered when computing the reduced costs, thus changing Equation (23) into Equation (28).

$$\bar{c}_r = \sum_{(v_i, v_j) \in \mathcal{E}} b_{ij}^r \left(c_{ij} - \frac{\beta_i + \beta_j}{2} - \rho_{ij} \right) \quad (28)$$

This combination of techniques leads to an efficient exact method, whose performance will be analyzed in Section 6.

5. Population-based metaheuristic

The VRP-SL is known to be NP-hard as it generalizes the CVRP, and even sophisticated exact approaches can only solve small- and medium-scale problem instances within a reasonable CPU time. To fill this methodological gap and to solve the larger instances which arise in practice, we introduce a dedicated hybrid genetic search with advanced diversity control.

5.1. General structure of the method

The method, illustrated in Algorithm 1, uses the same resolution strategy as the unified hybrid genetic search (UHGS) of [62, 64]. Starting from an initial population, it iteratively selects two parents to generate an offspring individual via a crossover operator. This offspring is improved by means of a local search procedure and inserted in the population. This sequence of operations is performed until the termination of the method, once It_{NI} successive iterations without improvement have been performed.

Algorithm 1 Hybrid Genetic Search (HGS) for the VRP–SL

- 1: Initialize the population with random solutions
 - 2: **while** not It_{NI} consecutive iterations without improvement of the best solution **do**
 - 3: Select two parents P_1 and P_2
 - 4: Generate an offspring C by applying the crossover on P_1 and P_2
 - 5: Educate C using local search
 - 6: Insert C into the population
 - 7: **if** C is not feasible **then**
 - 8: With 50% probability, repair C and insert it into the population
 - 9: **if** It_{DIV} iterations elapsed since the last diversification **then**
 - 10: Diversify the population
 - 11: **return** best feasible solution
-

The method exploits penalized infeasible solutions in the population, which is divided into two sub-populations of feasible and infeasible individuals. Whenever one sub-population reaches a maximum size, a number of individuals are eliminated to retain the best solutions. A repair procedure is also applied on infeasible solutions to restore feasibility and generate additional feasible individuals. Finally, the approach uses an adaptive diversity management, which has been shown to be particularly successful when solving VRPs [62]. Parents and survivors selections are driven by two criteria, cost and diversity contribution, rather than solely on cost as in traditional GAs, and additional diversification phases are implemented after every It_{DIV} iterations without improvement to provide new solution characteristics to the population.

Finally, the proposed algorithm also significantly differs from UHGS in the definition of its basic building blocks: solution representation, crossover, local search moves, distance measure between individuals, and penalties allowed. These components are described below.

5.2. Solution Representation and Evaluation

In the proposed HGS, each individual in the population is represented by two chromosomes: a *service level chromosome*, which gives the current service level of each group $k \in \{1, \dots, K\}$, and the

giant-tour chromosome, which provides a permutation of visits for the serviced customers, without occurrences of the depot. This solution representation is illustrated in Figure 1. It is incomplete in the sense that some additional information, the locations of the visits to the depot, is needed to perform cost evaluations. Still, this information can be quickly recovered by means of a dynamic-programming algorithm, called *Split*, which optimally subdivides the giant-tour chromosome into separate routes. In the proposed HGS, the Split algorithm strictly respects the sequence of visits, i.e., it cannot select or exclude customer visits and modify the service levels. This is the classical context of application of the algorithm of [9] and [52], which reduces the splitting problem into the search of a shortest path in an acyclic graph in which each arc represents a possible route, i.e., a sequence of consecutive visits in the giant tour, connected to the depot. The reader is referred to [52] and [61] for a detailed description of the Split algorithm as well as an efficient $\mathcal{O}(n)$ implementation.

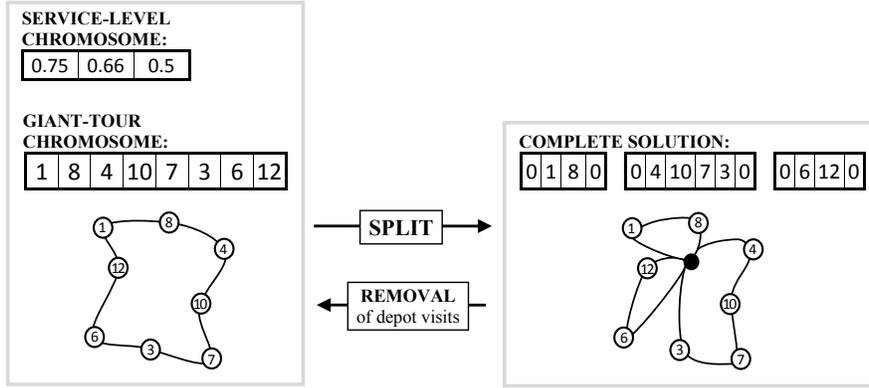


Figure 1: Solution representation and decoding via the Split algorithm. To compute the service levels, three groups are considered in the example: $\mathcal{V}_1 = \{1, 2, 3, 4\}$, $\mathcal{V}_2 = \{5, 6, 7, 8, 9, 10\}$ and $\mathcal{V}_3 = \{11, 12\}$. For all i , $s_i = 1$.

We now describe the cost function used for route and solution evaluations. As observed in our computational experiments and in [65], the exploration of penalized infeasible solutions during the search has a significantly positive impact on solution quality. Thus, the cost $\phi(r)$ of a route r involves the distance, the total profit associated to the customers which are visited, and a possible excess of load in a route multiplied by a penalty factor w^Q . Let $\phi^D(r) = \sum_{i=1}^{|r|-1} d_{r(i), r(i+1)}$, $\phi^Q(r) = \sum_{i=1}^{|r|} q_{r(i)}$ and $\phi^P(r) = \sum_{i=1}^{|r|} p_{r(i)}$ be, respectively, the total distance, load and profit collected in route r , then the route cost is defined as:

$$\phi(r) = \phi^D(r) - \phi^P(r) + w^Q \max\{0, \phi^Q(r) - Q\}. \quad (29)$$

We also explore infeasible solutions with respect to service level constraints. These linking constraints involve the whole solution, and thus the penalty is defined at the level of the solution

evaluation rather than the route cost. The penalized cost $\phi^{\text{COST}}(S)$ of a solution S , described as a set of routes $r \in S$, can thus be evaluated as:

$$\phi^{\text{COST}}(S) = \phi_P + \sum_{r \in S} \phi(r) + \sum_{k=1}^K w_k^S \max(\alpha_k - \phi^k(S), 0), \quad (30)$$

$$\text{with } \phi^k(S) = \sum_{v_i \in \mathcal{V}_k \cap S} s_i / \sum_{v_i \in \mathcal{V}_k} s_i \quad k \in \{1, \dots, K\}. \quad (31)$$

In this equation, $\phi_P = \sum_{i=1}^n p_i$ is the total profit of all customers (a constant), $\phi^k(S)$ is the weight ratio of group k in solution S , and w_k^S for $k \in \{1, \dots, K\}$ are the penalty coefficients associated to service level violations. These penalty coefficients are automatically adjusted by the method during the search, as explained in Section 5.4.

5.3. Generation of New Individuals

Each new solution is generated by a successive application of the *Selection*, *Crossover*, and *Education* operators, followed by a possible *Repair*.

Selection and Crossover. To generate a new solution, the algorithm first selects two parents P_1 and P_2 in the population via a binary tournament based on the *biased fitness* measure described in Section 5.4. A new offspring solution C is then obtained by crossover of P_1 and P_2 . For this purpose, we propose an *adapted order crossover* (AOX), which extends the well-known order crossover with the ability to transmit customer selection and visit sequence decision from both parents. This crossover operator is illustrated in Figure 2.

In a first step, the crossover inherits the service level information from both parents. This is done by crossing the service level chromosomes of both parents using an *extended intermediate recombination* [48], i.e., a target weight ratio $\alpha_k^T(C)$ is randomly chosen between $\alpha_k(P_1)$ and $\alpha_k(P_2)$ for each group k , where $\alpha_k(P)$ is the weight ratio of group k in individual P .

Then, in a second step, the giant-tour chromosome of the child C is initialized with the longest size among both parent and inherits, as in the *order crossover* (OX), a fragment of P_1 .

In the third and final step, the giant-tour chromosome of C is completed by sweeping circularly the deliveries of P_2 and inheriting them, starting one index after the end of the fragment from P_1 . Each insertion of a visit i of a group k is done under the condition that i does not already exist in C , and that the target service level $\alpha_k^T(C)$ has not yet been reached. To complete the representation, the service level chromosome of the child is finally derived from the giant-tour chromosome.

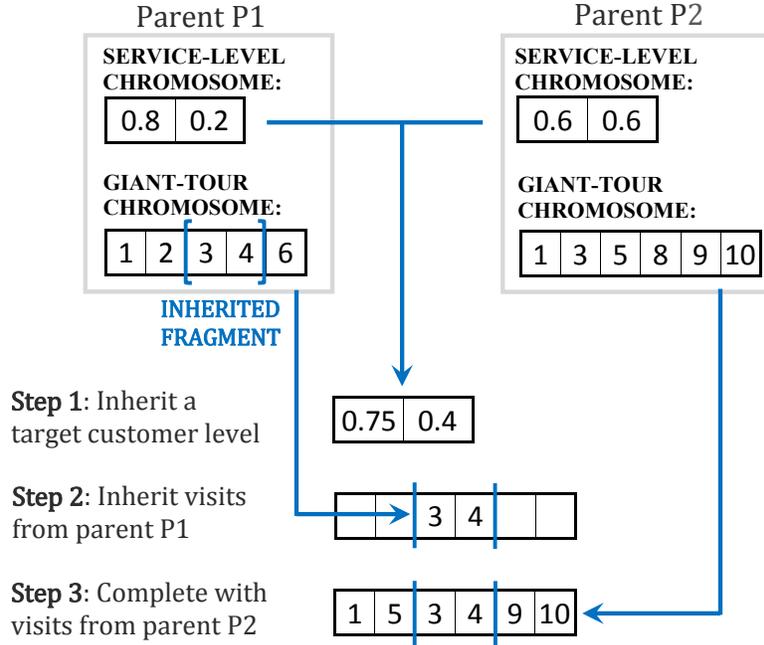


Figure 2: AOX crossover – In this example, $\mathcal{V}_1 = \{1, 2, 3, 4, 5\}$, $\mathcal{V}_2 = \{6, 7, 8, 9, 10\}$ and $s_i = 1$ for all i .

Education. The goal of the crossover operator was to generate new solutions which inherit common characteristics from both parents while introducing a significant level of randomness. As such, the crossover operator is not the main force which drives solution improvement, this role being assumed by a subsequent local search-based education procedure.

The local search (LS) is applied on the complete solution representation, including the visits to the depot. Therefore, the Split algorithm has to be run beforehand. The LS uses the same classical vehicle routing neighborhoods as in [62]: 2-OPT, 2-OPT*, SWAP, RELOCATE as well as generalized SWAP and RELOCATE involving two consecutive nodes, and limited to moves between close services. These classical neighborhoods only involve services which are present in the current solution. To also optimize the decision subset related to customer selections in the LS, we include three additional neighborhoods:

- REMOVE: If u is a visited customer, then remove u from the solution.
- ADD: If u is a visited customer and v is a non-visited customer, add v after u .
- REPLACE: If u is a visited customer and v is a non-visited customer, replace u by v .

All neighborhoods are explored in a random order with a first improvement move acceptance policy. The LS stops when no improving move can be found in the entire neighborhood, and the resulting solution is converted back into a giant-tour and service level individual representation, which is inserted in the population.

Repair. Finally, it is possible for a solution to remain infeasible after education. When this situation happens, a *Repair* operator is applied with 50% chance. As in [62], this operator simply consists in running the local search with higher penalty values, first by a factor of 10, and then 100, with the aim of converging towards a feasible solution.

5.4. Population Management

The population is formed of two sub-populations, designed to host feasible and infeasible individuals, respectively. The algorithm starts by generating $4 \times \mu$ random initial solutions. Each sub-population is then managed to contain at least μ and at most $\mu + \lambda$ individuals. Whenever a population reaches its maximum size, λ individuals are eliminated to produce the next generation. This is done by iteratively selecting out either a *clone* solution, with a distance of 0 to another solution, or the worst solution according to a biased fitness function $\phi^{\text{BIAS}}(S)$ when there are no more clones. The biased fitness measure evaluates every solution S based on its cost ϕ^{COST} (Equation 30) and its contribution to the sub-population diversity, defined as:

$$\phi_P^{\text{DIV}}(S) = \frac{\sum_{S_2 \in N_P(S)} \delta(S, S_2)}{n^{\text{CLOSE}}}, \quad (32)$$

where $N_P(S)$ is the set of the n^{CLOSE} individuals in the sub-population P closest to S with respect to a distance measure $\delta(S_1, S_2)$. Our distance measure counts the percentage of common edges between two solutions. Let $E(S_1)$ and $E(S_2)$ be the set of edges used in the solutions S_1 and S_2 , then the distance is expressed as:

$$\delta(S_1, S_2) = 1 - \frac{|E(S_1) \cap E(S_2)|}{|E(S_1) \cup E(S_2)|}. \quad (33)$$

Finally, let n^{ELITE} be a parameter controlling how elitist $\phi^{\text{BIAS}}(S)$ is, and let $R(S, P, f)$ be an application which returns the rank of an individual S in the sub-population P relatively to a measure f . Then, the biased fitness of S in P is evaluated as in UHGS [64]:

$$\phi_P^{\text{BIAS}}(S) = R(S, P, \phi^{\text{COST}}) + \left(1 - \frac{n^{\text{ELITE}}}{|P|}\right) R(S, P, \phi^{\text{DIV}}). \quad (34)$$

Each distance measure can be evaluated in $\mathcal{O}(n)$ time. As such, when a new individual enters the population, its distance from every other solution can be computed in $\mathcal{O}(n|P|)$ time, and the computational complexity of updating the biased fitness measures is $\mathcal{O}(n|P| + |P| \log |P|)$.

Adaptive Penalty Coefficients. The exploration of infeasible solutions contributes positively to the search if the ratio of feasible solutions is adequately controlled. To that extent, the penalty parameters are adapted to achieve a ratio of feasible solutions within a predefined interval $[\xi^{\text{MIN}}, \xi^{\text{MAX}}]$.

We rely on $1 + K$ penalty parameters for the VRP-SL: one for the capacity constraints, and K penalties for the service level constraints, one for each group. Let ξ^c be the ratio of feasible individuals with respect to a constraint c , measured among the last 100 individuals generated by local search. In order to drive the search towards feasible solutions, every 100 iterations we update the penalty coefficient of constraint c using the following rule:

$$w^c = \begin{cases} w^c \times 1.2 & \text{if } \xi^c \leq \xi^{\text{MIN}}, \\ w^c \times 0.85 & \text{if } \xi^c \geq \xi^{\text{MAX}}, \\ w^c & \text{otherwise.} \end{cases} \quad (35)$$

Initially, all penalty parameters are set to 10. We aim to obtain around 25% feasible solutions after LS. Since all constraints need to be respected to obtain a globally feasible solution, we used $\xi^{\text{MIN}} = 0.15^{\frac{1}{1+K}}$ and $\xi^{\text{MAX}} = 0.35^{\frac{1}{1+K}}$ to achieve this goal.

Diversification procedure. Finally, after each consecutive It_{DIV} iterations without improvement of the best solution, we apply the same diversification procedure as in [62] to only keep the best $\mu/3$ individuals in each subpopulation and reintroduce new random initial solutions. This procedure complements the biased fitness function so as to avoid a premature convergence of the method due to the strong intensification of the LS.

6. Experimental Analyses

This section aims to 1) introduce a set of instances for the VRP-SL derived from classical vehicle routing instances, 2) evaluate the performance of the proposed methods on the VRP-SL instances, 3) evaluate the performance of the metaheuristic and the branch-and-price on classical problems generalized by the VRP-SL, namely the VRPPFCC and CPTP, and finally 4) examine the impact of some key parameters and design choices. All algorithms were coded in C/C++. The exact algorithms were run on a single thread of a 3.07-GHz Intel Xeon CPU, and the metaheuristic was run on a single thread of a 3.4-GHz Intel Core i7 CPU. We used CPLEX 12.7 for the resolution of the compact formulation and for the linear programs in the B&P algorithm.

Benchmark instances for the VRP-SL. To compensate for the unavailability of benchmark instances for the VRP-SL, we derived two sets of instances from classical CVRP and prize-collecting VRP test sets:

- The first set (**S1**) has been generated from a subset of 26 instances from [4], and includes between 31 and 80 vertices. The profit of each client has been set to $h \times q_i$, where h is a random variable uniformly generated in the interval $[0.75, 2.25]$.
- The second set (**S2**) has been derived from the 10 capacitated profitable tour instances of [1], with 51 to 200 vertices. The capacity of the vehicles has been set to 500, and the number of vehicles has been set to

$$m = \left\lceil \frac{\sum_{k=1}^K (Q_k^{\text{MIN}} + Q_k^{\text{MAX}})}{2Q} \right\rceil, \quad (36)$$

where Q_k^{MIN} is the subset of services of group k with smallest delivery quantity which allows to satisfy the service level requirements, and Q_k^{MAX} is the sum of demands of all customers of this group. The original profits were multiplied by a factor of 0.5 to obtain a good balance between profits and distance.

For each instance, we considered five configurations for the assignment of visits to groups: $\{1, 2R, 2C, 5R, 5C\}$. The first number corresponds to the number of groups, and the second letter, when applicable, corresponds to their distribution: random (R) or clustered (C). In all cases, the service levels for each group have been randomly selected in $\{0.45, 0.55, 0.65, 0.75, 0.85, 0.95, 1\}$, and the weight of each customer, reflecting its importance for the service level constraint, coincides with the demand quantity ($s_i = q_i$). Overall, this leads to $26 \times 5 = 130$ instances of set **S1**, and $10 \times 5 = 50$ instances of set **S2**. For each instance, the distance values between customers are rounded to the nearest integer. For the sake of brevity, the results presented in the paper are aggregated per group of five instances. All instances and detailed results are available in the electronic companion of this paper, also available at <https://w1.cirrelet.ca/~vidalt/en/VRP-resources.html>.

6.1. Exact solutions and lower bounds

The two exact methods have been tested on each VRP-SL instance using a single core with a time limit of two hours. Since the VRP-SL instances were designed to be challenging for both exact and heuristic approaches, only a subset of the problems could be solved to optimality. To speed up the resolution, we used the best integer solution found by the metaheuristic as a warm start for both exact methods, hence limiting the size of the branch-and-bound tree.

Table 2 presents average results for each group of instances. The first columns list the characteristics of each group, followed by the minimal number of visits n_{MIN} required to satisfy service level

Data	n	m	n _{MIN}	Compact Formulation				Branch-and-Price							
				LB ₀	T ₀ (s)	LB	Tree	Gap	T(s)	LB ₀	T ₀ (s)	LB	Tree	Gap	T(s)
A1	31	5	19.4	643.1	0.50	701.6	756k	0.78	3423.93	685.9	0.13	707.4	319	0.00	33.82
A2	32	6	17.6	610.0	0.37	652.7	650k	0.77	2288.52	643.8	0.12	656.1	2k	0.30	1443.06
A3	35	5	21.8	627.1	0.40	666.2	407k	0.00	1806.92	652.4	0.15	666.2	1k	0.00	224.77
A4	36	6	18.0	757.7	0.56	802.4	386k	0.28	2482.49	792.6	0.13	805.0	103	0.00	5.93
A5	38	5	18.8	629.3	0.55	667.7	584k	0.25	3212.89	649.7	0.21	667.9	3k	0.22	1861.53
A6	43	7	25.4	739.9	0.88	772.7	414k	1.94	4477.12	772.9	0.25	788.6	4k	0.11	2890.31
A7	44	7	28.2	956.9	1.66	989.3	560k	3.25	7200.00	1006.1	0.31	1020.8	8k	0.21	4118.13
A8	47	7	26.6	874.5	1.40	905.9	670k	4.47	7200.00	926.9	0.21	944.0	7k	0.44	3394.61
A9	53	7	34.2	993.4	1.94	1015.1	439k	5.30	7200.00	1045.0	0.67	1066.0	8k	0.64	4905.96
A10	59	9	37.4	1136.6	3.18	1163.1	390k	5.67	7200.00	1208.3	0.61	1226.3	14k	0.60	5981.17
A11	61	8	33.0	1033.1	3.62	1057.3	249k	5.90	7200.00	1094.7	0.74	1112.1	7k	1.04	4953.56
A12	62	9	35.2	1055.1	2.54	1080.0	350k	4.58	7200.00	1113.6	0.60	1128.3	18k	0.35	6236.87
A13	64	9	41.6	1030.2	2.17	1052.7	529k	4.40	7200.00	1074.9	0.56	1090.4	11k	1.00	5764.22
A14	79	10	43.2	1432.9	6.00	1453.1	177k	4.70	7200.00	1494.4	1.39	1508.6	7k	1.09	7200.00
B1	30	5	21.6	526.0	0.38	576.3	1786k	1.42	5794.55	550.3	0.15	569.7	8k	2.37	5762.20
B2	34	5	16.8	695.4	0.26	736.0	1128k	1.51	3476.90	709.6	0.31	733.9	6k	1.91	5760.99
B3	38	6	18.0	434.4	0.52	460.6	1673k	7.24	7200.00	466.5	0.60	477.9	10k	3.84	7200.00
B4	42	6	22.0	618.2	0.83	650.7	766k	2.29	6352.01	652.0	0.46	663.2	3k	0.44	2705.18
B5	44	5	22.8	548.8	0.82	587.0	1040k	6.13	7200.00	606.2	0.77	614.8	7k	1.68	7200.00
B6	49	7	30.0	599.1	1.17	630.1	1049k	5.74	7200.00	646.5	0.91	657.2	7k	1.69	5140.14
B7	50	7	35.4	863.7	0.58	908.5	1259k	4.87	7200.00	916.5	0.91	933.3	6k	2.23	5950.49
B8	55	7	37.2	563.0	1.29	574.3	1117k	9.76	7200.00	605.9	1.45	613.1	7k	3.65	7200.00
B9	56	9	35.2	1291.6	1.34	1316.1	724k	4.32	7200.00	1332.6	0.63	1347.1	14k	2.04	5761.09
B10	63	9	34.0	726.2	1.97	761.4	731k	6.22	7200.00	781.9	1.07	793.5	8k	2.27	7200.00
B11	66	10	42.0	878.6	1.85	907.8	787k	4.54	7200.00	923.4	1.21	936.1	8k	1.57	7200.00
B12	77	10	43.0	1003.3	3.77	1020.3	450k	6.49	7200.00	1060.6	1.91	1069.1	9k	2.02	7200.00
p03	100	3	53.8	542.1	3.48	554.4	176k	0.11	2734.21	545.0	92.27	548.4	209	1.21	7200.00
p06	50	2	30.0	370.4	0.44	387.0	1k	0.00	5.26	374.1	11.61	381.0	418	1.54	7200.00
p07	75	2.8	42.8	506.9	1.41	519.6	69k	0.00	301.26	511.4	28.14	515.0	519	0.89	6489.66
p08	100	3	55.2	531.3	3.33	544.3	109k	0.09	1784.18	534.4	44.19	537.6	232	1.34	7200.00
p09	150	4.6	88.2	669.4	17.11	684.5	2170k	1.58	7200.00	681.9	306.57	684.0	79	1.64	7200.00
p10	199	6	114.4	773.8	61.80	783.4	16k	3.31	7200.00	798.2	699.11	799.6	45	1.32	7200.00
p13	120	3	75.0	529.0	10.03	538.9	114k	15.77	7200.00	587.3	837.90	588.9	27	7.96	7200.00
p14	100	4	55.2	491.8	4.83	504.9	227k	14.13	7200.00	560.3	90.53	568.5	575	3.31	7200.00
p15	100	3.8	65.6	493.4	4.72	506.9	212k	13.92	7200.00	565.4	108.80	572.1	557	2.85	7200.00
p16	199	6.2	108.2	769.7	66.16	782.2	204k	2.69	7200.00	792.7	571.70	794.0	37	1.22	7200.00

Table 2: Results of the exact methods for the VRP-SL instances

constraints, the value LB_0 of the lower bound and the processing time T_0 at the root node, the value LB of the best overall lower bound, the number of nodes in the search tree, the final integrality gap and the total CPU time. For each instance, the best lower bound is highlighted in boldface. We do not indicate the best upper bound found by each method, since no improvement was found over the initial value obtained by the metaheuristic.

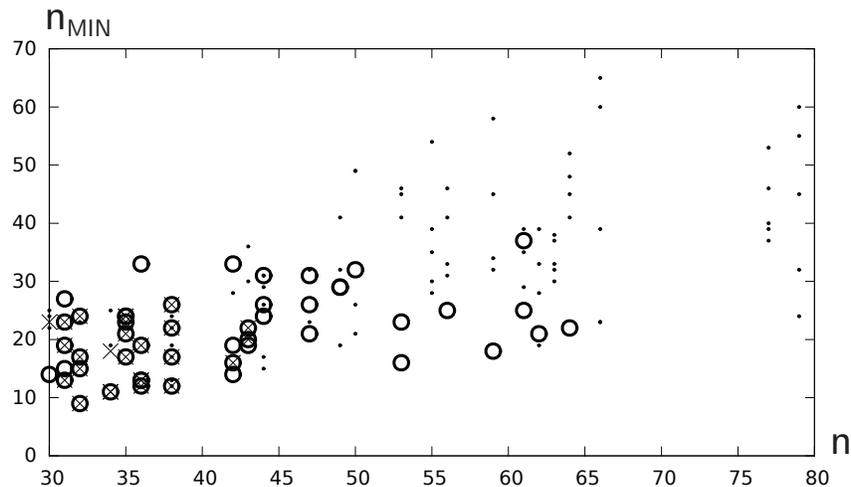


Figure 3: Instances of set **S1** currently solved to optimality using the compact formulation “x”, via branch-and-price “o”, or still open “.”, as a function of the number of visits n and the minimum number of deliveries n_{MIN} needed to satisfy the service level constraint.

As illustrated by the experiments, the branch-and-price algorithm outperformed the compact formulation-based method on most instances. More precisely, it performed better on all instances of the set **S1** with the exception of B1 and B2. For the set **S2**, the compact formulation performed better for the first five groups of instances and worse for the remaining ones. As expected, the branch-and-price algorithm is very effective when the solution includes short routes, while the compact formulation performs better on instances with few vehicles. The total number of instances solved to optimality for configurations $\{1, 2R, 2C, 5R, 5C\}$ were $\{10, 9, 9, 9, 8\}$ for the compact formulation and $\{15, 9, 9, 8, 9\}$ for the branch-and-price algorithm. These results show that the exact methods’ performances are rather insensitive to the distribution of nodes in V , as well as the number of groups K . However, the complexity generally increases with n , m and n_{MIN} , as visualized in Figure 3.

6.2. Performance of the hybrid genetic algorithm

In this subsection, we report the results obtained with the proposed HGS on the VRP-SL instances. For each instance, the algorithm was run ten times with different seeds, using the same parameter

setting and termination criterion as in [62] for the CVRP, that is $(n^{\text{ELITE}}, \mu, \lambda) = (10, 25, 40)$, $It_{\text{MAX}} = 2 \times 10^4$ and $It_{\text{DIV}} = 0.4 \times It_{\text{MAX}}$. Moreover, to accelerate the convergence, the parameters governing the population size have been halved when dealing with problem instances containing 200 or more services. Table 3 reports, for each group of instances, the worst, average and best solution quality of HGS over 10 runs (Wor-10, Avg-10 and Best-10), the percentage gap between the average solutions and the best ones found (Gap_{BKS}), the percentage gap between the average solutions and the best lower bounds found by the exact methods (Gap_{LB}), the average CPU time in seconds, the best known solutions (BKS) and lower bounds (BKLB).

Inst	n	m	Wor-10	Avg-10	Best-10	Gap _{BKS}	Gap _{LB}	T(s)	BKS	BKLB
A1	31	5	707.4	707.40	707.4	0.00	0.00	9.06	707.4	707.4
A2	32	6	658.2	658.20	658.2	0.00	0.32	8.72	658.2	656.1
A3	35	5	666.2	666.20	666.2	0.00	0.00	9.70	666.2	666.2
A4	36	6	805.0	805.00	805.0	0.00	0.00	9.51	805.0	805.0
A5	38	5	669.4	669.40	669.4	0.00	0.23	10.18	669.4	667.9
A6	43	6	789.6	789.60	789.6	0.00	0.13	12.78	789.6	788.6
A7	44	7	1023.0	1023.00	1023.0	0.00	0.22	13.90	1023.0	1020.8
A8	47	7	948.2	948.20	948.2	0.00	0.44	12.99	948.2	944.0
A9	53	7	1073.2	1073.20	1073.2	0.00	0.68	18.49	1073.2	1066.0
A10	59	9	1234.8	1234.50	1234.4	0.01	0.67	21.21	1234.4	1226.3
A11	61	8	1124.8	1124.10	1124.0	0.01	1.08	20.69	1124.0	1112.1
A12	62	10	1132.6	1132.44	1132.4	0.00	0.36	19.54	1132.4	1128.3
A13	64	9	1101.8	1101.70	1101.6	0.01	1.04	22.64	1101.6	1090.4
A14	79	10	1526.2	1525.30	1525.2	0.01	1.11	32.05	1525.2	1508.6
B1	30	5	584.8	584.80	584.8	0.00	0.57	9.53	584.8	581.5
B2	34	5	748.2	748.20	748.2	0.00	1.10	8.34	748.2	740.0
B3	38	5	497.2	497.20	497.2	0.00	3.98	10.24	497.2	478.2
B4	42	6	666.2	666.20	666.2	0.00	0.46	12.00	666.2	663.2
B5	44	5	625.0	625.00	625.0	0.00	1.66	11.71	625.0	614.8
B6	49	7	668.8	668.80	668.8	0.00	1.77	14.25	668.8	657.2
B7	50	7	956.2	956.20	956.2	0.00	2.38	15.13	956.2	934.0
B8	55	7	637.4	637.40	637.4	0.00	3.97	18.26	637.4	613.1
B9	56	9	1378.2	1376.72	1376.2	0.04	2.03	22.17	1376.2	1349.4
B10	63	9	812.0	812.00	812.0	0.00	2.33	20.20	812.0	793.5
B11	66	10	951.8	951.56	951.4	0.02	1.65	24.15	951.4	936.1
B12	77	10	1092.6	1091.94	1091.4	0.05	2.13	28.92	1091.4	1069.1
p03	100	-	555.0	555.00	555.0	0.00	0.10	22.67	555.0	554.4
p06	50	-	387.0	387.00	387.0	0.00	0.00	10.74	387.0	387.0
p07	75	-	519.6	519.60	519.6	0.00	0.00	17.51	519.6	519.6
p08	100	-	544.8	544.80	544.8	0.00	0.09	22.50	544.8	544.3
p09	150	-	695.6	695.60	695.6	0.00	1.50	46.12	695.6	685.3
p10	199	-	810.8	810.44	810.2	0.03	1.36	86.30	810.2	799.6
p13	120	-	640.8	640.20	639.8	0.06	8.70	44.05	639.8	588.9
p14	100	-	588.0	588.00	588.0	0.00	3.43	27.04	588.0	588.5
p15	100	-	588.8	588.80	588.8	0.00	2.93	26.78	588.8	572.1
p16	199	-	805.0	804.24	803.8	0.05	1.29	92.85	803.8	794.0
All						0.01	1.38	22.58		

Table 3: Performance of the HGS on the VRP-SL instance sets

In the absence of results from previously published heuristics, we consider three main indicators of method performance: its ability to reach known optimal solutions, the stability of the solution quality over several runs, illustrated by the gap between the average solution quality and the BKS, and the deviation from the lower bounds produced by the mathematical programming algorithms. As observed in our experiments, HGS finds all of the 70 known optimal solutions on all test runs. The method also returns solutions of consistent high quality: it found for 25/36 groups of instances the value of the best known solution on all runs of all instances. The percentage gaps between the average and best known solutions are close to zero (0.01% overall), and the instances with fewer groups appear to be generally easier to solve. The gaps to the lower bounds are also small (1.38% in average), and thus the solutions of HGS are guaranteed to be close to the optima. This gap is more likely to be due to the quality of the lower bounds, since over all 2h-runs of the exact approaches not a single best solution of the metaheuristic was improved. Finally, the average CPU time never exceeds 93 seconds, the worst case being observed on problem p16 with 199 services.

6.3. Comparative analyses on key subproblems

As discussed in Section 2, the VRP-SL generalizes several important problem classes. Two such problems in particular, the VRPPFCC and CPTP, have been the focus of a wide literature, opening the way to some comparative performance analyses.

Experiments on the VRPPFCC. We rely on the two sets of instances from [12]. Set CE includes up to 199 customers, while Set G includes larger instances with up to 483 customers. For these instances, the convention is to compute all distances with double precision and report the final solution with two digits. Our heuristic method is compared to the two best current metaheuristics in the literature: the UGHS proposed by [66], which uses an *exhaustive* solution representation (with all visits) with a route-evaluation operator in charge of customer selections, and the recent AVNS of [32]. To the best of our knowledge there are no known exact results for this problem in the literature. The results of our methods are displayed in Table 4. The column BKS reports the best known solutions found in the literature before this paper. Avg-10 and Best-10 represent the average and best solutions found by the metaheuristic over ten runs. For each instance, the best heuristic method is highlighted in boldface. For the branch-and-price results, columns LB_0 and LB follow the same convention as Table 2.

In these experiments, the proposed metaheuristic appears to outperform previous methods in terms of solution quality, with an average gap of 0.141%, in comparison to 0.445% for UGHS with the exhaustive solution representation, and 0.345% for AVNS. During these tests, eight new best

Instance	n	m	UGHS – [66]			AVNS – [32]			This paper – HGS			This paper – B&P			BKS
			Avg-10	Best-10	T(s)	Avg-10	Best-10	T(s)	Avg-10	Best-10	T(s)	LB ₀	LB	T(s)	
CE-01	50	4	1119.66	1119.47	38.5	1119.47	1119.47	47.4	1119.47	1119.47	13.01	1112.17	1119.47	154.30	1119.47
CE-02	75	9	1815.63	1814.52	59.6	1814.52	1814.52	95.8	1814.52	1814.52	24.00	1795.13	1807.79	7200.00	1814.52
CE-03	100	6	1922.88	1919.05	476.8	1919.05	1919.05	236.7	1919.05	1919.05	45.16	1896.81	1904.30	7200.00	1919.05
CE-04	150	9	2509.82	2505.39	934.7	2505.39	2518.14	2509.81	642.7	2510.35	98.95	2478.50	2483.50	7200.00	2505.39
CE-05	199	13	3095.58	3081.59	1289.3	3081.59	3101.40	3090.49	1437.2	3094.78	213.69	3051.13	3053.92	7200.00	3081.59
CE-06	50	4	1207.47	1207.47	38.5	1207.47	1207.47	44.2	1207.47	1207.47	12.42	1199.81	1207.47	196.41	1207.47
CE-07	75	9	2012.33	2006.52	73.2	2006.52	2009.58	2004.53	98.6	2006.32	25.71	1984.80	1997.90	7200.00	2004.53
CE-08	100	6	2057.57	2052.05	500.3	2052.05	2059.56	2052.05	239.6	2058.18	48.30	2029.89	2037.66	7200.00	2052.05
CE-09	150	10	2428.19	2425.32	1241.0	2425.32	2426.35	2420.71	933.9	2422.88	123.43	2389.86	2394.60	7200.00	2419.84
CE-10	199	13	3387.12	3381.67	1229.9	3381.67	3388.22	3373.84	1704.1	3384.03	238.95	3340.29	3343.29	7200.00	3373.84
CE-11	120	6	2331.13	2330.94	1202.9	2330.94	2330.94	2330.94	399.0	2330.95	71.94	2317.21	2321.26	7200.00	2330.94
CE-12	100	8	1953.13	1952.86	150.8	1952.86	1952.86	1952.86	183.6	1953.27	34.28	1939.32	1946.63	7200.00	1952.86
CE-13	120	6	2859.07	2858.83	1184.9	2858.83	2858.83	2858.83	437.8	2858.94	65.11	2843.90	2847.50	7200.00	2858.83
CE-14	100	7	2213.02	2213.02	189.8	2213.02	2213.78	2213.02	185.2	2213.41	33.11	2193.70	2202.21	7200.00	2213.02
G-01	240	7	14151.51	14131.18	2405.9	14163.43	14129.48	2433.9	2433.9	14135.93	210.40	14024.30	14038.00	7200.00	14111.95
G-02	320	8	19190.77	19166.58	2409.9	19254.23	19140.69	6630.0	6630.0	19134.06	302.27	18974.40	18978.70	7200.00	19140.69
G-03	400	8	24588.29	24409.02	2418.1	24566.00	24406.67	11062.8	11062.8	24372.81	431.70	—	—	7200.00	24368.29
G-04	480	8	34517.47	34362.8	2421.1	34425.00	34231.56	15875.4	15875.4	34116.56	401.61	—	—	7200.00	34231.56
G-05	200	4	14296.07	14223.63	2408.0	14261.06	14229.50	1591.3	1591.3	14268.87	71.98	—	—	7200.00	14223.63
G-06	280	5	21488.29	21396.60	2411.4	21440.38	21357.16	4337.2	4337.2	21376.19	125.28	—	—	7200.00	21357.16
G-07	360	7	23463.05	23373.38	2414.9	23440.51	23263.22	7585.3	7585.3	23285.33	354.48	—	—	7200.00	23263.22
G-08	440	8	29918.06	29823.18	2415.6	29864.19	29657.38	12316.5	12316.5	29603.54	415.88	—	—	7200.00	29657.38
G-09	255	11	1332.63	1328.65	2323.4	1325.79	1320.29	3852	3852	1324.06	335.90	1308.14	1308.59	7200.00	1319.72
G-10	323	13	1603.82	1597.61	2342.3	1592.14	1588.05	6922.9	6922.9	1587.42	647.96	1567.66	1568.01	7200.00	1583.50
G-11	399	14	2192.68	2182.01	2405.2	2172.45	2163.50	12303.3	12303.3	2166.52	861.43	2137.91	2138.06	7200.00	2159.78
G-12	483	15	2529.84	2522.64	2407.2	2493.94	2483.06	19555.0	19555.0	2486.56	956.42	2455.51	2455.64	7200.00	2479.62
G-13	252	21	2261.50	2258.02	1412.7	2264.92	2261.66	2260.2	2260.2	2266.41	325.29	2232.70	2232.87	7200.00	2258.02
G-14	320	23	2687.50	2683.73	1935.7	2689.99	2684.66	4838.5	4838.5	2696.11	682.99	2654.94	2655.23	7200.00	2683.73
G-15	396	26	3152.00	3145.11	2301.4	3156.84	3150.67	8198.0	8198.0	3153.76	858.02	3107.12	3107.42	7200.00	3145.11
G-16	480	29	3632.04	3620.71	2450.1	3633.76	3624.56	12741.7	12741.7	3635.54	954.65	3578.85	3579.04	7200.00	3620.71
G-17	240	18	1671.72	1666.31	1805.3	1672.81	1666.31	1878	1878	1667.81	160.84	1666.31	1666.31	19.69	1666.31
G-18	300	22	2733.12	2730.55	2035.0	2734.86	2731.28	3916.6	3916.6	2733.49	459.92	2717.24	2717.95	7200.00	2730.55
G-19	360	26	3504.26	3497.20	1989.5	3499.55	3494.28	5225.4	5225.4	3495.68	839.89	3477.87	3478.79	7200.00	3494.27
G-20	420	31	4319.37	4312.45	2523	4314.48	4307.63	8081.6	8081.6	4307.61	1115.09	4277.33	4278.20	7200.00	4306.85
Avg. Gap(%)			0.445	0.210		0.345	0.058			0.141	-0.012	—	—		
Avg. T(s)			1583.70		4656.22										
CPU			Xe 3.07GHz		Intel i5 2.6GHz					Intel i7 3.4GHz			Intel i7 3.4GHz		
															6575.60

Table 4: Performance of the HGS on VRPPFC benchmark instances, in comparison with the current state-of-the-art algorithms

known solutions (BKS) have been found, as underlined in the table. Finally, the average CPU time is markedly faster than previous approaches which were run on processors of a similar generation, with 340 seconds on average, compared to 1584 and 4656 seconds for the other methods. The proposed branch-and-price algorithm is the first in the literature to report optimal VRPPFCC solutions for three instances: CE-01, CE-06 and G-17. On the other hand, on six instances, the algorithm was not able to solve the root node within a time limit of two hours. This is due to their size (up to 480 customers), which can be considered very large for the current exact methods. Finally, the average integrality gap calculated over the tractable instances was 0.667%, confirming the good performance of the approach.

Experiments on the CPTP. For this problem, we compare the proposed HGS with the previous two best methods in the literature, the UHGS with *exhaustive* solution representation and the multi-start ILS of [66]. We also compare the proposed branch-and-price algorithm with the exact approach of [1], which generated, to this date, the best bounds for the CPTP. In that work, the authors proposed a branch-and-price algorithm using a q -route relaxation, and reported detailed results with and without a primal heuristic. We rely on the ten test cases of [2], each case being used to produce 12 instances with a different fleet size and vehicle capacity, for a total of 120 instances. Tables 5 and 6 present a summary of our computational experiments on these instances, using the same conventions as previously. Each line in the table corresponds to an average measure over a group of 12 instances.

Instance	n	UGHS – [66]			ILS – [66]			This paper – HGS			BKS
		Avg-10	Best-10	T(s)	Avg-10	Best-10	T(s)	Avg-10	Best-10	T(s)	
p03	100	254.07	254.07	215.82	253.99	254.07	191.63	254.07	254.07	12.32	254.07
p06	50	129.13	129.13	30.78	129.09	129.11	23.35	129.13	129.13	7.18	129.13
p07	75	192.51	192.56	103.94	192.37	192.56	85.80	192.52	192.56	8.36	192.56
p08	100	254.07	254.07	216.28	253.93	254.07	191.49	253.90	254.07	12.17	254.07
p09	150	319.61	319.72	295.34	319.23	319.72	289.75	319.67	319.72	18.44	319.72
p10	199	387.87	388.41	303.55	386.67	387.73	309.23	388.79	388.85	19.90	388.41
p13	120	180.20	180.39	235.98	178.33	180.32	255.97	180.08	180.32	15.15	180.39
p14	100	246.24	246.24	116.26	246.23	246.24	111.83	246.24	246.24	10.78	246.24
p15	150	327.99	328.36	295.02	326.91	327.81	289.85	328.28	328.37	15.24	328.36
p16	199	393.09	393.75	303.88	392.21	393.32	309.74	394.03	394.04	21.16	393.75
Avg. Gap(%)		0.029	0.000		0.172	0.014		0.012	-0.002		
Avg. T(s)				211.68			205.86			14.07	
CPU		Xe 3.07GHz			Xe 3.07GHz			Intel i7 3.4GHz			

Table 5: Performance of HGS on the CPTP benchmark instances

These instances are generally smaller, with 50 to 199 service locations, and thus the performance differences between state-of-the-art heuristics are less marked. Still, we observe that the proposed

Instance	n	B&P1 – [1]			B&P2 – [1]			This paper – B&P				BKUB
		UB	LB	T(s)	UB	LB	T(s)	UB ₀	UB	LB	T(s)	
p03	100	256.90	254.07	1135.31	256.82	147.01	1135.38	258.17	255.36	254.07	1116.00	256.82
p06	50	129.75	129.13	625.08	129.64	118.09	602.38	131.77	129.39	129.13	287.89	129.64
p07	75	193.18	192.56	622.38	193.11	163.49	613.92	194.21	192.82	192.56	285.06	193.11
p08	100	256.89	254.07	1134.31	256.82	147.01	1135.00	258.17	255.36	254.07	1116.05	256.82
p09	150	324.25	316.33	1163.69	324.20	160.90	1163.85	323.66	320.83	319.72	1123.65	324.20
p10	199	392.03	377.01	1117.77	391.66	176.43	1117.77	392.07	390.43	388.85	596.37	391.66
p13	120	191.74	167.78	1916.08	191.70	116.06	1916.77	186.72	183.93	180.32	1216.18	191.70
p14	100	255.47	237.02	1110.31	255.41	116.24	1110.31	248.67	247.20	246.24	556.22	255.40
p15	150	331.98	324.19	893.54	331.93	190.18	893.77	332.36	330.67	328.37	603.52	331.93
p16	199	398.57	379.03	1118.46	398.27	178.90	1118.38	398.58	396.47	394.04	941.83	398.27
Average		273.07	263.12	1083.69	272.96	151.43	1080.75	272.44	270.25	268.74	784.28	272.96
CPU		Xe 2.26GHz			Xe 2.26GHz			Intel i7 3.4GHz				

Table 6: Performance of the proposed branch-and-price algorithm on the CPTP benchmark instances

HGS obtains average solutions of similar or better quality (0.012% gap compared to 0.029% and 0.172% gap) in a fraction of the CPU time of previous algorithms (14 seconds in average, compared to 211 or 205 seconds). Three previous BKS were improved, leading to an average gap of -0.002% for the best solution quality of 10 runs. For the branch-and-price algorithm, we set the time limit to 3600 seconds in order to make a fair comparison with [1]. The proposed B&P found similar or better solutions for all instances tested. It improved the bounds for 37 instances, with an average improvement of 0.629%, and proved optimality for 103 instances, including ten new optimality certificates.

6.4. Sensitivity analyses

Finally, this section reports additional sensitivity analyses on the impact of key components of the proposed HGS. For this purpose, we compare the results of the standard method described in Section 5 against several alternative configurations obtained by modifying one operator, design choice, or group of parameters:

EOX – An edge-recombination crossover (EOX) is used. As described in [67], this crossover maintains, for each vertex i , an *adjacency list* of non-visited vertices adjacent to i in at least one parent. After a random choice for the first vertex, EOX iteratively inserts the adjacent vertex with the shortest adjacency list. Ties are broken randomly, and a random vertex is chosen whenever the adjacency list is empty. As in our adaptation of OX, target service levels are inherited for the groups from the parents, and any vertex belonging to a group for which the target service level has already been attained is eliminated from the adjacency lists.

No SL – Service levels are not used to filter service insertions in the crossover.

No INF – All penalty coefficients are set to a large value to avoid infeasibility.

No DIV – Individual diversity contributions are not counted in the biased fitness.

No Rep – The Repair operator is not applied.

Pop ↓ – Smaller population: $(\mu^{elite}, \mu, \lambda) = (4, 12, 20)$.

Pop ↑ – Larger population: $(\mu^{elite}, \mu, \lambda) = (16, 50, 80)$.

Feas ↑ – 50% feasible solutions as a target: $(\xi^{MIN}, \xi^{MAX}) = (0.4^{\frac{1}{1+K}}, 0.6^{\frac{1}{1+K}})$.

Feas ↑↑ – 75% feasible solutions as a target: $(\xi^{MIN}, \xi^{MAX}) = (0.65^{\frac{1}{1+K}}, 0.85^{\frac{1}{1+K}})$.

Each of these algorithm configurations was run 10 times on every benchmark instance for the VRP-SL, VRPPFCC and CPTP. Table 7 presents the average percentage gap, best percentage gap and time of each method for each set of instances.

	VRP-SL Set S1			VRP-SL Set S2			VRPPFCC			CPTP		
	Best-10	Avg-10	T(s)	Best-10	Avg-10	T(s)	Best-10	Avg-10	T(s)	Best-10	Avg-10	T(s)
Standard	0.00	0.01	16.01	0.00	0.01	39.66	0.06	0.21	340.00	0.00	0.02	14.07
EOX	0.02	0.05	19.98	0.03	0.10	66.61	1.69	2.04	225.45	0.02	0.06	20.39
No SL	0.00	0.01	16.70	0.00	0.02	41.68	0.07	0.20	360.61	0.02	0.04	15.22
No INF	0.05	0.12	15.05	0.07	0.14	40.84	0.15	0.38	344.67	0.02	0.05	12.70
No DIV	0.01	0.05	13.62	0.01	0.08	29.04	0.12	0.37	155.33	0.02	0.17	11.68
No Repair	0.00	0.02	12.51	0.01	0.03	34.85	0.05	0.20	337.18	0.00	0.02	12.40
Pop ↓	0.00	0.01	12.99	0.00	0.02	31.16	0.05	0.23	343.94	0.00	0.03	11.24
Pop ↑	0.00	0.01	26.48	0.00	0.02	60.53	0.12	0.26	545.01	0.00	0.01	25.14
Feas ↑	0.00	0.01	15.12	0.00	0.02	40.01	0.10	0.26	348.12	0.01	0.03	14.06
Feas ↑↑	0.01	0.04	14.93	0.01	0.05	40.61	0.16	0.37	380.05	0.01	0.05	14.23

Table 7: Sensitivity Analysis on the components of the HGS

From these experiments, it appears that the proposed method is a sort of “local optimum” in terms of design choice and parameter settings, in the sense that any change of its main operators and parameters impacts negatively the method performance. Still, some design choices have a much larger impact than others. In particular, using the EOX crossover operator strongly deteriorates the method performance for VRPPFCC instances, while speeding-up the resolution. Such a speed-up may be a symptom of premature convergence due, in this case, to the crossover. Both diversity management and infeasible-solution management contribute significantly to the performance of the method (configurations **No INF** and **No DIV**). This confirms the earlier observations of [62, 65]. Still, although the management of infeasible solutions is critical, deactivating the repair operator or changing the target level of feasible solutions has little impact (configurations **No Rep**, **Feas** ↑ and

Feas $\uparrow\uparrow$). The control of the service levels in the crossover has a beneficial effect on performance for the CPTP (configuration **No SL**). Finally, HGS is relatively insensible to reasonable changes of population size (configurations **Pop** \downarrow and **Pop** \uparrow).

7. Conclusions

In this article, we have introduced the VRP-SL, an important VRP variant arising in collaborative logistics operations, which aims to take into account the requirements of various partners via service level constraints on groups of deliveries. To establish a basis for further research, we introduced a first set of benchmark instances, a compact mathematical formulation, a branch-and-price algorithm and a first effective hybrid genetic search. The service level constraints tend to make the selection of services more complex, and thus new problem-tailored search operators, solution representation, crossover, LS moves, and penalty management strategies were introduced in HGS. Thanks to these elements, the proposed heuristic finds all known optimal solutions for the VRP-SL, and outperforms previous algorithms for two important special cases, the VRPPFCC and the CPTP, which have been intensively studied in past literature. Finally, the proposed branch-and-price algorithm was able to produce tight bounds for all problems and new optimality certificates for 63 instances, outperforming the existing exact approaches.

The research perspectives are numerous. First, the new algorithms can still be improved via the addition of new families of cuts, better relaxations, new neighborhoods and other heuristic strategies. Moreover, the VRP-SL is only a simplification of an intricate real-life application, and the assumptions about the time constraints, the dynamics and stochasticity of the problem were voluntarily simplified to allow for reproducibility. Guaranteeing, in a stochastic and on-line context, the satisfaction of contractual service levels for prize-collecting problems is an important challenge, as the violation of such obligations can lead to large penalties or lost contracts. To circumvent this risk while mitigating the cost of robustness, it is possible to search for robust solutions on a larger planning horizon, and consider alternative transportation modes as a recourse. These aspects, and the interactions between them, will be considered in future works.

8. Acknowledgments

This research is partially supported by CNPq and CAPES in Brazil, and the National Foundation for Science and Technology Development (NAFOSTED) in Vietnam.

References

- [1] Archetti, C., N. Bianchessi, M.G. Speranza. 2013. Optimal solutions for routing problems with profits. *Discrete Applied Mathematics* **161**(4-5) 547–557.
- [2] Archetti, C., D. Feillet, A. Hertz, M.G. Speranza. 2009. The capacitated team orienteering and profitable tour problems. *Journal of the Operational Research Society* **60**(6) 831–842.
- [3] Archetti, C., M.G. Speranza, D. Vigo. 2014. Vehicle routing problems with profits. P. Toth, D. Vigo, eds., *Vehicle Routing: Problems, Methods, and Applications*. SIAM, Philadelphia, PA, 273–297.
- [4] Augerat, P., J.M. Belenguer, E. Benavent, A. Coberan, D. Naddef, G. Rinaldi. 1995. Computational results with a branch-and-cut code for the capacitated vehicle routing problem. Tech. rep., Université Joseph Fourier, Grenoble, France.
- [5] Baldacci, R., E. Bartolini, G. Laporte. 2009. Some applications of the generalized vehicle routing problem. *Journal of the Operational Research Society* **61**(7) 1072–1077.
- [6] Baldacci, R., N. Christofides, A. Mingozzi. 2008. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming* **115**(2) 351–385.
- [7] Baldacci, R., E. Hadjiconstantinou, A. Mingozzi. 2004. An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations Research* **52**(5) 723–738.
- [8] Baldacci, R., A. Mingozzi, R. Roberti. 2011. New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research* **59**(5) 1269–1283.
- [9] Beasley, J.E. 1983. Route first-cluster second methods for vehicle routing. *Omega* **11**(4) 403–408.
- [10] Bektas, T., G. Erdogan, S. Ropke. 2011. Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. *Transportation Science* **45**(3) 299–316.
- [11] Bérubé, J.F., M. Gendreau, J.Y. Potvin. 2009. A branch-and-cut algorithm for the undirected prize collecting traveling salesman problem. *Networks* **54**(1) 56–67.

- [12] Bolduc, M.-C., J. Renaud, F. Boctor, G. Laporte. 2008. A perturbation metaheuristic for the vehicle routing problem with private fleet and common carriers. *Journal of the Operational Research Society* **59**(6) 776–787.
- [13] Boussier, S., D. Feillet, M. Gendreau. 2007. An exact algorithm for team orienteering problems. *4OR* **5**(3) 211–230.
- [14] Campos, V., R. Martí, J. Sánchez-Oro, A. Duarte. 2014. GRASP with path relinking for the orienteering problem. *Journal of the Operational Research Society* **65** 1800–1813.
- [15] Chaves, A.A., L.A.N. Lorena. 2008. Hybrid metaheuristic for the prize collecting travelling salesman problem. J. van Hemert, C. Cotta, eds., *Evolutionary Computation in Combinatorial Optimization, LNCS*, vol. 4972. Springer, Berlin, Heidelberg, 123–134.
- [16] Christofides, N., A. Mingozzi, P. Toth. 1981. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming* **20** 255–282.
- [17] Contardo, C., R. Martinelli. 2014. A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discrete Optimization* **12** 129–146.
- [18] Côté, J.-F., J.-Y. Potvin. 2009. A tabu search heuristic for the vehicle routing problem with private fleet and common carrier. *European Journal of Operational Research* **198**(2) 464–469.
- [19] Dang, D.-C., R.N. Guibadj, A. Moukrim. 2013. An effective PSO-inspired algorithm for the team orienteering problem. *European Journal of Operational Research* **229**(2) 332–344.
- [20] Dell’Amico, M., F. Maffioli, P. Värbrand. 1995. On prize-collecting tours and the asymmetric travelling salesman problem. *International Transactions in Operational Research* **2**(3) 297–308.
- [21] Dror, M. 1994. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research* **42**(5) 977–978.
- [22] El-Hajj, R., D.-C. Dang, A. Moukrim. 2016. Solving the team orienteering problem with cutting planes. *Computers & Operations Research* **74** 21–30.
- [23] Feillet, D., P. Dejax, M. Gendreau. 2005. Traveling salesman problems with profits. *Transportation Science* **39**(2) 188–205.
- [24] Fischetti, M., J.J.S. Gonzalez, P. Toth. 1998. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing* **10**(2) 133–148.

- [25] Fischetti, M., P. Toth. 1988. An additive approach for the optimal solution of the prize-collecting travelling salesman problem. B.L. Golden, A.A. Assad, eds., *Vehicle Routing: Methods and Studies*. Elsevier, 319–343.
- [26] Fukasawa, R., H. Longo, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa, R.F. Werneck. 2006. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming* **106**(3) 491–511.
- [27] Gavish, B., S.C. Graves. 1978. The travelling salesman problem and related problems. Tech. rep., Massachusetts Institute of Technology, Cambridge, MA, USA.
- [28] Gendreau, M., G. Laporte, F. Semet. 1998. A tabu search heuristic for the undirected selective travelling salesman problem. *European Journal of Operational Research* **106**(2-3) 539–545.
- [29] Ghiani, G., G. Improta. 2000. An efficient transformation of the generalized vehicle routing problem. *European Journal of Operational Research* **122**(1) 11–17.
- [30] Hà, M.H., N. Bostel, A. Langevin, L.-M. Rousseau. 2013. An exact algorithm and a metaheuristic for the multi-vehicle covering tour problem with a constraint on the number of vertices. *European Journal of Operational Research* **226**(2) 211–220.
- [31] Hà, M.H., N. Bostel, A. Langevin, L.-M. Rousseau. 2014. An exact algorithm and a metaheuristic for the generalized vehicle routing problem with flexible fleet size. *Computers & Operations Research* **43**(1) 9–19.
- [32] Huijink, S., K. Goos, R. Peeters. 2014. An adaptable variable neighborhood search for the vehicle routing problem with order outsourcing. Tech. rep., Tilburg University.
- [33] Irnich, S., D. Villeneuve. 2006. The shortest-path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing* **18**(3) 391–406.
- [34] Jayanth, J., T. Keah-Choon. 2010. Supply chain integration with third-party logistics providers. *International Journal of Production Economics* **125**(2) 262–271.
- [35] Jepsen, M.K., B. Petersen, S. Spoorendonk, D. Pisinger. 2014. A branch-and-cut algorithm for the capacitated profitable tour problem. *Discrete Optimization* **14**(1) 78–96.
- [36] Kataoka, S., S. Morito. 1988. An algorithm for single constraint maximum collection problem. *Journal of the Operations Research Society of Japan* **31**(4) 515–531.

- [37] Ke, L., C. Archetti, Z. Feng. 2008. Ants can solve the team orienteering problem. *Computers & Industrial Engineering* **54**(3) 648–665.
- [38] Ke, L., L. Zhai, J. Li, F.T.S. Chan. 2015. Pareto mimic algorithm: An approach to the team orienteering problem. *Omega* **61**(1) 155–166.
- [39] Keshtkaran, M., K. Ziarati, A. Bettinelli, D. Vigo. 2016. Enhanced exact solution methods for the Team Orienteering Problem. *International Journal of Production Research* **54**(2) 591–601.
- [40] Kim, B.-I., H. Li, A.L. Johnson. 2013. An augmented large neighborhood search method for solving the team orienteering problem. *Expert Systems with Applications* **40**(8) 3065–3072.
- [41] Letchford, A.N., J.-J. Salazar-González. 2006. Projection results for vehicle routing. *Mathematical Programming* **105**(2-3) 251–274.
- [42] Leuschner, R., C.R. Carter, T.J. Goldsby, Z.S. Rogers. 2014. Third-party logistics: A meta-analytic review and investigation of its impact on performance. *Journal of Supply Chain Management* **50**(1) 21–43.
- [43] Li, K., H. Tian. 2016. A two-level self-adaptive variable neighborhood search algorithm for the prize-collecting vehicle routing problem. *Applied Soft Computing* **43**(1) 469–479.
- [44] Lin, S.-W. 2013. Solving the team orienteering problem using effective multi-start simulated annealing. *Applied Soft Computing* **13**(2) 1064–1073.
- [45] Lopez, L., M.W. Carter, M. Gendreau. 1998. The hot strip mill production scheduling problem: A tabu search approach. *European Journal of Operational Research* **106**(2-3) 317–335.
- [46] Martinelli, R., D. Pecin, M. Poggi. 2014. Efficient elementary and restricted non-elementary route pricing. *European Journal of Operational Research* **239**(1) 102–111.
- [47] Martinelli, R., D. Pecin, M. Poggi, H. Longo. 2011. A branch-cut-and-price algorithm for the capacitated arc routing problem. P. Pardalos, S. Rebennack, eds., *Experimental Algorithms, LNCS*, vol. 6630. Springer, 315–326.
- [48] Mühlenbein, H., D. Schlierkamp-Voosen. 1993. Predictive models for the breeder genetic algorithm I. Continuous parameter optimization. *Evolutionary Computation* **1**(1) 25–49.
- [49] Pedro, O., R. Saldanha, R. Camargo. 2013. A tabu search approach for the prize collecting traveling salesman problem. *Electronic Notes in Discrete Mathematics* **41**(1) 261–268.

- [50] Pessoa, A., E. Uchoa, M. Poggi de Aragão, R. Rodrigues. 2010. Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Mathematical Programming Computation* **2**(3) 259–290.
- [51] Potvin, J.-Y., M.-A. Naud. 2011. Tabu search with ejection chains for the vehicle routing problem with private fleet and common carrier. *Journal of the Operational Research Society* **62**(2) 326–336.
- [52] Prins, C. 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research* **31**(12) 1985–2002.
- [53] Schilde, M., K.F. Doerner, R.F. Hartl, G. Kiechle. 2009. Metaheuristics for the bi-objective orienteering problem. *Swarm Intelligence* **3**(1) 179–201.
- [54] Souffriau, W., P. Vansteenwegen, G. Vanden Berghe, D. Van Oudheusden. 2010. A path relinking approach for the team orienteering problem. *Computers & Operations Research* **37**(11) 1853–1859.
- [55] Stefansson, G. 2006. Collaborative logistics management and the role of third-party service providers. *International Journal of Physical Distribution & Logistics Management* **36**(2) 76–92.
- [56] Stenger, A., M. Schneider, D. Goeke. 2013. The prize-collecting vehicle routing problem with single and multiple depots and non-linear cost. *EURO Journal on Transportation and Logistics* **2**(1-2) 57–87.
- [57] Stenger, A., D. Vigo, S. Enz, M. Schwind. 2012. An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. *Transportation Science* **47**(1) 64–80.
- [58] Tang, L., X. Wang. 2006. Iterated local search algorithm based on very large-scale neighborhood for prize-collecting vehicle routing problem. *The International Journal of Advanced Manufacturing Technology* **29**(11-12) 1246–1258.
- [59] Vansteenwegen, P., W. Souffriau, G.V. Berghe, D.V. Oudheusden. 2009. A guided local search metaheuristic for the team orienteering problem. *European Journal of Operational Research* **196**(1) 118–127.
- [60] Vansteenwegen, P., W. Souffriau, D.V. Oudheusden. 2010. The orienteering problem: A survey. *European Journal of Operational Research* **209**(1) 1–10.

- [61] Vidal, T. 2016. Technical note: Split algorithm in $O(n)$ for the capacitated vehicle routing problem. *Computers & Operations Research* **69** 40–47.
- [62] Vidal, T., T.G. Crainic, M. Gendreau, N. Lahrichi, W. Rei. 2012. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research* **60**(3) 611–624.
- [63] Vidal, T., T.G. Crainic, M. Gendreau, C. Prins. 2013. Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. *European Journal of Operational Research* **231**(1) 1–21.
- [64] Vidal, T., T.G. Crainic, M. Gendreau, C. Prins. 2014. A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research* **234**(3) 658–673.
- [65] Vidal, T., T.G. Crainic, M. Gendreau, C. Prins. 2015. Time-window relaxations in vehicle routing heuristics. *Journal of Heuristics* **21**(3) 329–358.
- [66] Vidal, T., N. Maculan, L.S. Ochi, P.H.V. Penna. 2016. Large neighborhoods with implicit customer selection for vehicle routing problems with profits. *Transportation Science* **50**(2) 720–734.
- [67] Whitley, L.D., T. Starkweather, D. Fuquay. 1989. Scheduling problems and traveling salesmen: The genetic edge recombination operator. *Proceedings of the 3rd International Conference on Genetic Algorithms*. Morgan Kaufmann, San Francisco, CA, USA, 133–140.
- [68] Yadollahpour, M.R., M. Bijari, S. Kavosh, M. Mahnam. 2009. Guided local search algorithm for hot strip mill scheduling problem with considering hot charge rolling. *International Journal of Advanced Manufacturing Technology* **45**(11) 1215–1231.
- [69] Zhang, T., W. Chaovallitwongse, Y.-J. Zhang, P.M. Pardalos. 2009. The hot-rolling batch scheduling method based on the prize collecting vehicle routing problem. *Journal of Industrial and Management Optimization* **5**(4) 749–765.