Constraints manuscript No.
(will be inserted by the editor)

A Branch-and-Price-and-Check Model for the Vehicle Routing Problem with Location Congestion

Edward Lam · Pascal Van Hentenryck

Received: date / Accepted: date

Abstract This paper considers a vehicle routing problem with pickup and delivery, time windows and location congestion. Locations provide a number of cumulative resources that are utilized by vehicles either during service (e.g., forklifts) or for the entirety of their visit (e.g., parking bays). Locations can become congested if insufficient resources are available, upon which vehicles must wait until a resource becomes available before proceeding. The problem is challenging from a computational standpoint since it incorporates the vehicle routing problem and the resource-constrained project scheduling problem. The main contribution of this paper is a branch-and-price-and-check model that uses a branch-and-price algorithm that solves the underlying vehicle routing problem, and a constraint programming subproblem that checks the feasibility of the location resource constraints, and then adds combinatorial nogood cuts to the master problem if the resource constraints are violated. Experimental results show the benefits of the branch-and-price-and-check approach.

Keywords Vehicle Routing Problem · Synchronization

1 Introduction

A Vehicle Routing Problem (VRP) is a combinatorial optimization problem that aims to construct routes for a fleet of vehicles that service customer requests while minimizing some cost function. The family of VRPs is extensive and includes variants that specify additional side constraints, such as time

E. Lam
University of Melbourne, Parkville, VIC 3010, Australia
NICTA, West Melbourne, VIC 3003, Australia
E-mail: edward.lam@nicta.com.au

P. Van Hentenryck
University of Michigan, Ann Arbor, MI 48109-2117, USA
E-mail: pvanhent@umich.edu

window constraints that restrict the time at which service of a request can commence, and precedence constraints that require one request to be serviced before another.

This paper explores a rich variant named the Vehicle Routing Problem with Pickup and Delivery, Time Windows, and Location Congestion (VRPPDTWLC, or VRPLC for short). The VRPLC is motivated by applications in humanitarian and military logistics, where Air Force bases have limited parking spots, fuel reserve, and landing and takeoff times for airplane operations. At the modeling level, the VRPLC is based on the traditional VRP with Pickup and Delivery, and Time Windows (VRPPDTW), but includes a cumulative resource constraint at every location to limit the number of vehicles present and/or in service at any given time. Examples of resources may include parking bays for the first case, and forklifts for the second case. If all resources at a location are in use, incoming vehicles cannot proceed but must wait until a resource becomes available. This leads to temporal dependencies between vehicles and a scheduling substructure not present in conventional VRPs. In particular, a delay on one route may entail a delay on another route if both routes visit a common location. Furthermore, the delay on the second route may cause it to become infeasible because of a time window constraint, for example.

This paper studies a mixed integer programming model, a constraint programming model and a branch-and-price-and-check (BPC) model for the VRPLC. The BPC approach, inspired by a branch-and-cut-and-price method for the VRPPDTW [17], combines a branch-and-price algorithm that solves the VRPPDTW, and a constraint programming subproblem that lifts the VRPPDTW to a VRPLC by checking the VRPPDTW solutions against the location resource constraints. If these constraints are violated, a combinatorial Benders cut (or nogood) is added to the master problem to prohibit this infeasible VRPPDTW solution.

The three models are evaluated on instances with up to 300 requests (150 pickup and delivery requests) and both types of resources. Results indicate that the BPC algorithm scales better than both the mixed integer programming and the constraint programming models, optimally solves instances with 160 requests in under 10 minutes, and finds high quality solutions to larger problems. The BPC model nicely exploits the strengths of constraint programming for scheduling and branch-and-price for vehicle routing.

The rest of this paper is structured as follows. Section 2 reviews prior work on relevant problems and solution techniques. Section 3 describes the problem. Sections 4 and 5 present the mixed integer programming model and the constraint programming model. Section 6 discusses the BPC approach. Section 7 reports experimental results, and Section 8 concludes this paper.

2 Literature Review

Vehicle routing problems are studied extensively in the literature and can be solved using a range of methods [21]. A standard formulation is a flow-based mixed integer programming (MIP) model that uses big-M rewritings to model logical constraints. Since the linear relaxation of this model is typically weak, it cannot scale to larger instances. This issue is mitigated in branch-and-cut approaches, which periodically solve a separation subproblem during search to generate valid inequalities that strengthen this formulation (e.g., [1,18]).

VRPs have also been approached using column generation, in which the flow-based MIP model undergoes Dantzig-Wolfe decomposition to form a master problem and a pricing subproblem [7]. Column generation iterates between the two problems until convergence. The pricing subproblem solves a resource-constrained shortest path problem to generate a number of individual routes. These routes are added to a large set of routes, and on this set, the master problem solves the linear relaxation of a set covering or set partitioning problem to select a subset that visits all requests. Since the master problem is a linear program, the final solution of column generation may select a fraction of a route. In this case, integrality constraints are enforced by embedding column generation in a branch-and-bound search tree, giving us the branch-and-price scheme (e.g., [9]).

These two approaches are merged in the branch-and-cut-and-price method, which iterates between the master, separation and pricing problems (e.g., [17]). However, because the master problem is not a flow model, the cuts generated by the separation subproblem must be translated into a form that is suitable for the set covering or set partitioning master problem.

In addition to the aforementioned mathematical programming methods, VRPs can be solved using other technologies, such as constraint programming (CP), which can quickly obtain high quality solutions to large VRPs using large neighborhood search ([19,5]), but has considerable difficulty in proving optimality.

Related to routing problems are scheduling problems. In these problems, activities must be processed by a number of machines and the objective may involve minimizing makespan or tardiness. Examples of scheduling problems include the Job Shop Scheduling Problem and the Resource-Constrained Project Scheduling Problem (RCPSP), which requires activities of fixed duration to be scheduled on a number of machines such that precedence constraints and resource capacities are respected. Many benchmark instances of these problems are closed using CP.

Some scheduling problems can be translated into routing problems, and vice versa [3,4]. However, few problems exhibit both routing and scheduling constraints simultaneously. The VRPLC is one such example: it overlays either a regular RCPSP or a variable-duration version on top of a VRP. In the VRPLC, vehicles must communicate their schedules in order to satisfy resource capacities at locations. Hence, the VRPLC is classified as a rich VRP with

synchronization [8]. Synchronization refers to the interaction between vehicles and the transfer of information from one vehicle to another.

Location resource scheduling within VRPs have received little attention in the literature. One method of modeling these resource constraints is to redefine individual routes as segments within a single long route [11]. In this approach, the resource constraints can only appear at destination depot locations, unlike the VRPLC, in which resource constraints can appear at any location.

The Log Truck Scheduling Problem is a related problem and features aspects of the VRPLC. In particular, it allows resource constraints at intermediate locations. This problem can be solved using a variety of methods, of which two are relevant. The first method models the location resources using scheduling constraints in a CP model [10], while the second uses a time-indexed column generation model [16].

Since the VRPLC incorporates both routing and scheduling, it may be advantageous to hybridize MIP and CP to capitalize on their unique strengths in routing and scheduling. One method for hybridization is branch-and-check [20, 2], a form of logic-based Benders decomposition [12]. In branch-and-check, the problem is separated into "easy" and "hard" parts. The easy parts are solved normally, and the hard parts are delayed until a solution for the easy parts is found. If the hard parts are infeasible for the given solution to the easy parts, a constraint prohibiting this solution is added to the master problem. Branch-and-check iterates between the master problem and the checking subproblem until a globally optimal solution is found. Since branch-and-check adds no-good cuts into the master problem, it is essentially a branch-and-cut search that uses a Benders subproblem as its separation subproblem.

3 The VRPLC Problem

This section describes the VRPLC problem, which augments a regular VRP with Pickups and Deliveries, and Time Windows (VRPPDTW) with two major additions: the explicit modeling of locations and their resource constraints.

In traditional VRPs, requests and locations are synonymous; each request is assumed to be at its own location, even if it overlaps another location. In the VRPLC, requests are grouped by location, and locations cannot overlap. Distance costs and travel times are defined between locations. Moreover, every location features a number of cumulative resources, with a fixed capacity that cannot be exceeded at any given time. Two types of resources are considered: a *service* resource is used while a request is in service, and a *presence* resource is used by a vehicle from the moment it arrives at a location until it departs from the location. Service resources are fully encompassed by presence resources, since the start of a visit is necessarily before the start of service, and the end of a visit after the end of service. Even though the problem studied within this paper only considers one of these two types of resources, the formulation naturally generalizes to multiple copies of these types of resources.

Contrary to traditional VRPs, delaying a route in a VRPLC may affect the feasibility of other routes since, for example, a delayed vehicle may postpone the availability of a resource required by another vehicle. The delay incurred by the second vehicle may induce, for instance, a time window violation on this second route. These temporal interactions between vehicles require reasoning about the timing and scheduling of the vehicle visits, and make VRPLCs more challenging to solve than their more traditional counterparts.

4 The Mixed Integer Programming Model

This section presents the mixed integer programming model of the VRPLC. The model is based on the regular three-index VRP formulation, but uses additional time variables to record the arrival and departure times at locations, and includes the location resource constraints.

Table 1 lists the data and decision variables of the MIP model. Every request $i \in \mathcal{R}$ is grouped into a location $l \in \mathcal{L}$. Locations have either a service resource or a presence resource with capacity C_l. The resources are scheduled using event variables [13]. Every request $i \in \mathcal{R}_l$ at l is associated with two events: a start event and an end event. For service resources, the start and end events correspond to the start and end of service. For presence resources, the start and end events correspond to the arrival and departure of the servicing vehicle. The location resources are then scheduled by tracking the number of start and end events.

The primary decision variables of the model are the flow variables, which are defined on the arc set

$$\begin{aligned}\mathcal{A} =&\{(s,i) : i \in \mathcal{P}\} \cup \{(i,j) : i \in \mathcal{R}, j \in \mathcal{R}, i \neq j\} \cup \\ &\{(i,e) : i \in \mathcal{D}\} \cup \{(s,e)\}.\end{aligned} \tag{1}$$

Figure 1 depicts the model, where M is an appropriate big-M constant. The objective function minimizes the total travel distance (Eq. (2)). Constraints (3) to (5) are the flow constraints. Constraint (6) links flow variables to visit indicator variables. Constraint (7) is the request cover constraint. Constraints (8) and (9) are the pickup-delivery constraints. Constraints (10) to (12) order the arrival, service start, service end, and departure times at each request. Constraint (13) constrains the start node and end node to one common arrival/service/departure time. Constraints (14) and (15) are the travel time constraints. Constraints (16) to (19) are the usual load constraints.

The novelty of the model lies in Constraints (20) to (30), which model service resources. Constraint (20) requires every event to be classified as either a start event or an end event. Constraints (21) and (22) state that every request has a start event and an end event. Constraint (23) initializes the resource use count. Constraint (24) counts the resource use after every event. Constraint (25) initializes the event time variables. Constraint (26) orders the events. Constraints (27) to (30) are implication constraints that link the

Variable	Description
$T > 0$	Time horizon.
$\mathcal{T} = [0, T]$	Time interval.
$V \in \{1, \dots, \infty\}$	Number of vehicles.
$\mathcal{V} = \{1, \dots, V\}$	Set of vehicles.
$Q \geq 0$	Vehicle capacity.
$P \in \{1, \dots, \infty\}$	Total number of pickup-delivery pairs.
$R = 2P$	Total number of requests.
$\mathcal{P} = \{1, \dots, P\}$	Set of pickup requests/nodes.
$\mathcal{D} = \{P+1, \dots, R\}$	Set of delivery requests/nodes.
$\mathcal{R} = \mathcal{P} \cup \mathcal{D}$	Set of all requests.
s	Start node.
e	End node.
$\mathcal{N} = \mathcal{R} \cup \{s, e\}$	Set of all nodes.
$L \in \{1, \dots, \infty\}$	Number of locations, excluding the depot location.
$\mathcal{L} = \{1, \dots, L\}$	Set of locations.
$C_l \in \{1, \dots, \infty\}$	Resource capacity of location $l \in \mathcal{L}$.
$\mathcal{R}_l = \{i \in \mathcal{R} : l_i = l\}$	Requests at location $l \in \mathcal{L}$.
$\mathcal{K}_l = \{1, \dots, 2\lvert\mathcal{R}_l\rvert\}$	Set of events at location $l \in \mathcal{L}$.
$\mathcal{A}$	Set of arcs. Defined in Eq. (1).
$l_i \in \mathcal{L}$	Location of $i \in \mathcal{R}$.
$d_{i,j} \in \mathcal{T}$	Distance and travel time along the arc $(i, j) \in \mathcal{A}$.
$a_i \in \mathcal{T}$	Earliest start of service at $i \in \mathcal{N}$.
$b_i \in \mathcal{T}$	Latest start of service at $i \in \mathcal{N}$.
$t_i \in \mathcal{T}$	Service duration of $i \in \mathcal{N}$.
$q_i \in [-Q, Q]$	Demand at $i \in \mathcal{N}$.
$\text{flow}_{v,i,j} \in \{0, 1\}$	Indicates if vehicle $v \in \mathcal{V}$ traverses $(i, j) \in \mathcal{A}$.
$\text{visit}_{v,i} \in \{0, 1\}$	Indicates if vehicle $v \in \mathcal{V}$ visits $i \in \mathcal{R}$.
$\text{arr}_{v,i} \in \mathcal{T}$	Arrival time of vehicle $v \in \mathcal{V}$ at $i \in \mathcal{N}$.
$\text{serv}_{v,i} \in [a_i, b_i]$	Start of service by vehicle $v \in \mathcal{V}$ at $i \in \mathcal{N}$.
$\text{servEnd}_{v,i} \in [a_i + t_i, b_i + t_i]$	End of service by vehicle $v \in \mathcal{V}$ at $i \in \mathcal{N}$.
$\text{dep}_{v,i} \in \mathcal{T}$	Departure time of vehicle $v \in \mathcal{V}$ at $i \in \mathcal{N}$.
$\text{load}_{v,i} \in [0, Q]$	Load of vehicle $v \in \mathcal{V}$ after servicing $i \in \mathcal{N}$.
$\text{start}_{l,k,i} \in \{0, 1\}$	Indicates if $k \in \mathcal{K}_l$ represents the start event of $i \in \mathcal{R}_l$.
$\text{end}_{l,k,i} \in \{0, 1\}$	Indicates if $k \in \mathcal{K}_l$ represents the end event of $i \in \mathcal{R}_l$.
$\text{use}_{l,k} \in \{0, \dots, C_l\}$	Resource use after event $k \in \mathcal{K}_l \cup \{0\}$ at $l \in \mathcal{L}$.
$\text{time}_{l,k} \in \mathcal{T}$	Time of event $k \in \mathcal{K}_l \cup \{0\}$ at $l \in \mathcal{L}$.

Table 1 Data and decision variables of the mixed integer programming model.

event time variables to the route time variables. To model presence resources, Constraints (27) to (30) can be replaced with

$$\text{time}_{l,k} - \text{arr}_{v,i} \leq M(2 - \text{start}_{l,k,i} - \text{visit}_{v,i}), \qquad \forall l \in \mathcal{L}, k \in \mathcal{K}_l, v \in \mathcal{V}, i \in \mathcal{R}_l,$$
$$\text{arr}_{v,i} - \text{time}_{l,k} \leq M(2 - \text{start}_{l,k,i} - \text{visit}_{v,i}), \qquad \forall l \in \mathcal{L}, k \in \mathcal{K}_l, v \in \mathcal{V}, i \in \mathcal{R}_l,$$
$$\text{time}_{l,k} - \text{dep}_{v,i} \leq M(2 - \text{end}_{l,k,i} - \text{visit}_{v,i}), \qquad \forall l \in \mathcal{L}, k \in \mathcal{K}_l, v \in \mathcal{V}, i \in \mathcal{R}_l,$$
$$\text{dep}_{v,i} - \text{time}_{l,k} \leq M(2 - \text{end}_{l,k,i} - \text{visit}_{v,i}), \qquad \forall l \in \mathcal{L}, k \in \mathcal{K}_l, v \in \mathcal{V}, i \in \mathcal{R}_l,$$

which link the event time variables to arrival and departure times instead of service start and end times.

$$\min \sum_{v\in\mathcal{V}} \sum_{(i,j)\in\mathcal{A}} d_{i,j}\text{flow}_{v,i,j} \tag{2}$$

subject to

$$\sum_{j:(s,j)\in\mathcal{A}} \text{flow}_{v,s,j} = 1, \qquad \forall v \in \mathcal{V}, \tag{3}$$

$$\sum_{h:(h,i)\in\mathcal{A}} \text{flow}_{v,h,i} = \sum_{j:(i,j)\in\mathcal{A}} \text{flow}_{v,i,j}, \qquad \forall v \in \mathcal{V}, i \in \mathcal{R}, \tag{4}$$

$$\sum_{h:(h,e)\in\mathcal{A}} \text{flow}_{v,h,e} = 1, \qquad \forall v \in \mathcal{V}, \tag{5}$$

$$\sum_{h:(h,i)\in\mathcal{A}} \text{flow}_{v,h,i} = \text{visit}_{v,i}, \qquad \forall v \in \mathcal{V}, i \in \mathcal{R}, \tag{6}$$

$$\sum_{v\in\mathcal{V}} \text{visit}_{v,i} = 1, \qquad \forall i \in \mathcal{P}, \tag{7}$$

$$\text{visit}_{v,i} = \text{visit}_{v,P+i}, \qquad \forall v \in \mathcal{V}, i \in \mathcal{P}, \tag{8}$$

$$\text{dep}_{v,i} + d_{i,P+i} \leq \text{arr}_{v,P+i}, \qquad \forall v \in \mathcal{V}, i \in \mathcal{P}, \tag{9}$$

$$\text{arr}_{v,i} \leq \text{serv}_{v,i}, \qquad \forall v \in \mathcal{V}, i \in \mathcal{R}, \tag{10}$$

$$\text{serv}_{v,i} + t_i = \text{servEnd}_{v,i}, \qquad \forall v \in \mathcal{V}, i \in \mathcal{R}, \tag{11}$$

$$\text{servEnd}_{v,i} \leq \text{dep}_{v,i}, \qquad \forall v \in \mathcal{V}, i \in \mathcal{R}, \tag{12}$$

$$\text{arr}_{v,i} = \text{serv}_{v,i} = \text{servEnd}_{v,i} = \text{dep}_{v,i}, \qquad \forall v \in \mathcal{V}, i \in \{s,e\}, \tag{13}$$

$$\text{dep}_{v,i} + d_{i,j} - \text{arr}_{v,j} \leq M(1 - \text{flow}_{v,i,j}), \qquad \forall v \in \mathcal{V}, (i,j) \in \mathcal{A}, \tag{14}$$

$$\text{arr}_{v,j} - \text{dep}_{v,i} - d_{i,j} \leq M(1 - \text{flow}_{v,i,j}), \qquad \forall v \in \mathcal{V}, (i,j) \in \mathcal{A}, \tag{15}$$

$$\text{load}_{v,i} = 0, \qquad \forall v \in \mathcal{V}, i \in \{s,e\}, \tag{16}$$

$$q_i \leq \text{load}_{v,i} \leq Q, \qquad \forall v \in \mathcal{V}, i \in \mathcal{P}, \tag{17}$$

$$0 \leq \text{load}_{v,i} \leq Q + q_i, \qquad \forall v \in \mathcal{V}, i \in \mathcal{D}, \tag{18}$$

$$\text{load}_{v,i} + q_j - \text{load}_{v,j} \leq M(1 - \text{flow}_{v,i,j}), \qquad \forall v \in \mathcal{V}, (i,j) \in \mathcal{A}, \tag{19}$$

$$\sum_{i\in\mathcal{R}_l} \text{start}_{l,k,i} + \sum_{i\in\mathcal{R}_l} \text{end}_{l,k,i} = 1, \qquad \forall l \in \mathcal{L}, k \in \mathcal{K}_l, \tag{20}$$

$$\sum_{k\in\mathcal{K}_l} \text{start}_{l,k,i} = 1, \qquad \forall l \in \mathcal{L}, i \in \mathcal{R}_l, \tag{21}$$

$$\sum_{k\in\mathcal{K}_l} \text{end}_{l,k,i} = 1, \qquad \forall l \in \mathcal{L}, i \in \mathcal{R}_l, \tag{22}$$

$$\text{use}_{l,0} = 0, \qquad \forall l \in \mathcal{L}, \tag{23}$$

$$\text{use}_{l,k} = \text{use}_{l,k-1} + \sum_{i\in\mathcal{R}_l} \text{start}_{l,k,i} - \sum_{i\in\mathcal{R}_l} \text{end}_{l,k,i}, \qquad \forall l \in \mathcal{L}, k \in \mathcal{K}_l, \tag{24}$$

$$\text{time}_{l,0} = 0, \qquad \forall l \in \mathcal{L}, \tag{25}$$

$$\text{time}_{l,k-1} \leq \text{time}_{l,k}, \qquad \forall l \in \mathcal{L}, k \in \mathcal{K}_l, \tag{26}$$

$$\text{time}_{l,k} - \text{serv}_{v,i} \leq M(2 - \text{start}_{l,k,i} - \text{visit}_{v,i}), \qquad \forall l \in \mathcal{L}, k \in \mathcal{K}_l, v \in \mathcal{V}, i \in \mathcal{R}_l, \tag{27}$$

$$\text{serv}_{v,i} - \text{time}_{l,k} \leq M(2 - \text{start}_{l,k,i} - \text{visit}_{v,i}), \qquad \forall l \in \mathcal{L}, k \in \mathcal{K}_l, v \in \mathcal{V}, i \in \mathcal{R}_l, \tag{28}$$

$$\text{time}_{l,k} - \text{servEnd}_{v,i} \leq M(2 - \text{end}_{l,k,i} - \text{visit}_{v,i}), \qquad \forall l \in \mathcal{L}, k \in \mathcal{K}_l, v \in \mathcal{V}, i \in \mathcal{R}_l, \tag{29}$$

$$\text{servEnd}_{v,i} - \text{time}_{l,k} \leq M(2 - \text{end}_{l,k,i} - \text{visit}_{v,i}), \qquad \forall l \in \mathcal{L}, k \in \mathcal{K}_l, v \in \mathcal{V}, i \in \mathcal{R}_l. \tag{30}$$

Fig. 1 Constraints of the mixed integer programming model.

Name	Description
$T \in \{1, \ldots, \infty\}$	Time horizon.
$\mathcal{T} = \{0, \ldots, T\}$	Set of time values.
$V \in \{1, \ldots, \infty\}$	Number of vehicles.
$\mathcal{V} = \{1, \ldots, V\}$	Set of vehicles.
$Q \in \{0, \ldots, \infty\}$	Vehicle capacity.
$P \in \{1, \ldots, \infty\}$	Total number of pickup-delivery pairs.
$R = 2P$	Total number of requests.
$\mathcal{P} = \{1, \ldots, P\}$	Set of pickup requests/nodes.
$\mathcal{D} = \{P+1, \ldots, R\}$	Set of delivery requests/nodes.
$\mathcal{R} = \mathcal{P} \cup \mathcal{D}$	Set of all requests.
$\mathcal{S} = \{R+1, \ldots, R+V\}$	Set of vehicle start nodes.
$\mathcal{E} = \{R+V+1, \ldots, R+2V\}$	Set of vehicle end nodes.
$s(v) = R + v$	Start node of vehicle $v \in \mathcal{V}$.
$e(v) = R + V + v$	End node of vehicle $v \in \mathcal{V}$.
$\mathcal{N} = \mathcal{R} \cup \mathcal{S} \cup \mathcal{E}$	Set of all nodes.
$L \in \{1, \ldots, \infty\}$	Number of locations, excluding the depot location.
$\mathcal{L} = \{1, \ldots, L\}$	Set of locations.
$C_l \in \{1, \ldots, \infty\}$	Resource capacity of location $l \in \mathcal{L}$.
$\mathcal{R}_l = \{i \in \mathcal{R} : l(i) = l\}$	Requests at location $l \in \mathcal{L}$.
$l(i) \in \mathcal{L}$	Location of $i \in \mathcal{R}$.
$d(i, j) \in \mathcal{T}$	Distance and travel time from $i \in \mathcal{N}$ to $j \in \mathcal{N}$.
$a(i) \in \mathcal{T}$	Earliest start of service at $i \in \mathcal{N}$.
$b(i) \in \mathcal{T}$	Latest start of service at $i \in \mathcal{N}$.
$t(i) \in \mathcal{T}$	Service duration of $i \in \mathcal{N}$.
$q(i) \in \{-Q, \ldots, Q\}$	Demand at $i \in \mathcal{N}$.
$\text{succ}(i) \in \mathcal{N}$	Successor node of $i \in \mathcal{N}$.
$\text{veh}(i) \in \mathcal{V}$	Vehicle that visits $i \in \mathcal{N}$.
$\text{arr}(i) \in \mathcal{T}$	Arrival time at $i \in \mathcal{N}$.
$\text{serv}(i) \in \{a(i), \ldots, b(i)\}$	Start of service at $i \in \mathcal{N}$.
$\text{dep}(i) \in \mathcal{T}$	Departure time at $i \in \mathcal{N}$.
$\text{dur}(i) \in \mathcal{T}$	Visit duration at $i \in \mathcal{R}$.
$\text{load}(i) \in \{0, \ldots, Q\}$	Load of vehicle $\text{veh}(i)$ after servicing $i \in \mathcal{N}$.

Table 2 Data and decision variables of the constraint programming model.

5 The Constraint Programming Model

This section presents the constraint programming model for the VRPLC, which is adapted from the standard constraint programming model of VRPs using successor variables.

Table 2 lists the data and decision variables in the model. Note the inputs $\mathcal{R}_l$ and C_l which, respectively, represent the set of requests and the capacity of the resource at location l. The primary decision variables are the successor variables, which determine the route of each vehicle. The secondary decision variables are the arrival, service, and departure times.

Figure 2 depicts the model. The objective function minimizes the total travel distance (Eq. (32)). Constraints (33) to (35) are the domain restrictions that ensure the requests along a route are correctly ordered. Constraints (36) and (37) link the end of a route to the start of the next route, and en-

$$\min \sum_{i \in \mathcal{R} \cup \mathcal{S}} d(i, \text{succ}(i)) \tag{32}$$

subject to

$$\text{succ}(s(v)) \in \mathcal{P} \cup \{e(v)\}, \qquad \forall v \in \mathcal{V}, \tag{33}$$
$$\text{succ}(i) \in \mathcal{P} \cup \mathcal{D}, \qquad \forall i \in \mathcal{P}, \tag{34}$$
$$\text{succ}(i) \in \mathcal{P} \cup \mathcal{D} \cup \mathcal{E}, \qquad \forall i \in \mathcal{D}, \tag{35}$$
$$\text{succ}(e(v)) = s(v+1), \qquad \forall v \in \{1, \ldots, V-1\}, \tag{36}$$
$$\text{succ}(e(V)) = s(1), \tag{37}$$
$$\textsc{Circuit}(\text{succ}(\cdot)), \tag{38}$$
$$\text{veh}(s(v)) = \text{veh}(e(v)) = v, \qquad \forall v \in \mathcal{V}, \tag{39}$$
$$\text{veh}(i) = \text{veh}(\text{succ}(i)), \qquad \forall i \in \mathcal{R} \cup \mathcal{S}, \tag{40}$$
$$\text{veh}(i) = \text{veh}(P+i), \qquad \forall i \in \mathcal{P}, \tag{41}$$
$$\text{dep}(i) + d(i, P+i) \leq \text{arr}(P+i), \qquad \forall i \in \mathcal{P}, \tag{42}$$
$$\text{arr}(i) \leq \text{serv}(i), \qquad \forall i \in \mathcal{R}, \tag{43}$$
$$\text{serv}(i) + t(i) \leq \text{dep}(i), \qquad \forall i \in \mathcal{R}, \tag{44}$$
$$\text{dur}(i) = \text{dep}(i) - \text{arr}(i), \qquad \forall i \in \mathcal{R}, \tag{45}$$
$$\text{arr}(i) = \text{serv}(i) = \text{dep}(i), \qquad \forall i \in \mathcal{S} \cup \mathcal{E}, \tag{46}$$
$$\text{dep}(i) + d(i, \text{succ}(i)) = \text{arr}(\text{succ}(i)), \qquad \forall i \in \mathcal{R} \cup \mathcal{S}, \tag{47}$$
$$\text{load}(i) = 0, \qquad \forall i \in \mathcal{S} \cup \mathcal{E}, \tag{48}$$
$$q(i) \leq \text{load}(i) \leq Q, \qquad \forall i \in \mathcal{P}, \tag{49}$$
$$0 \leq \text{load}(i) \leq Q + q(i), \qquad \forall i \in \mathcal{D}, \tag{50}$$
$$\text{load}(i) + q(\text{succ}(i)) = \text{load}(\text{succ}(i)), \qquad \forall i \in \mathcal{R} \cup \mathcal{S}, \tag{51}$$
$$\textsc{Cumulative}(\{\text{serv}(i) : i \in \mathcal{R}_l\}, \{t(i) : i \in \mathcal{R}_l\}, \mathbf{1}, C_l), \qquad \forall l \in \mathcal{L}. \tag{52}$$

Fig. 2 Constraints of the constraint programming model.

ables the Circuit global constraint to eliminate subtours in Constraint (38). Constraints (39) and (40) track the requests visited by each vehicle. Constraints (41) and (42) are the pickup and delivery constraints. Constraints (43) and (44) order the arrival, service and departure times at each request. Constraint (45) calculates the duration of each visit. Constraint (46) enforces a common arrival/departure time at the start and end depot nodes. Constraint (47) implements travel times. Constraints (48) to (51) are the usual load constraints. Constraint (52) models the service resources. For presence resources, it can be replaced with

$$\textsc{Cumulative}(\{\text{arr}(i) : i \in \mathcal{R}_l\}, \{\text{dur}(i) : i \in \mathcal{R}_l\}, \mathbf{1}, C_l), \qquad \forall l \in \mathcal{L}. \tag{31}$$

The resource constraints are modeled using the $\textsc{Cumulative}(\mathbf{s}, \mathbf{d}, \mathbf{r}, C)$ global constraint, where $\mathbf{s}$, $\mathbf{d}$, and $\mathbf{r}$ are vectors that, respectively, represent the start time, duration, and resource requirement of each activity, and C is the capacity of the resource. It is important to note that the service durations are fixed by the instance data, whereas the presence durations are variables.

$$\min \sum_{r \in \Omega} c_r x_r \tag{53}$$

subject to

$$\sum_{r \in \Omega} a_{i,r} x_r = 1, \qquad \forall i \in \mathcal{P}, \tag{54}$$

$$\sum_{r \in \Omega} B_r x_r \leq |B| - 1, \qquad \forall B \in \mathcal{B}, \tag{55}$$

$$x_r \in [0, 1], \qquad \forall r \in \Omega. \tag{56}$$

Fig. 3 Master problem of the branch-and-price-and-check model.

6 The Branch-and-Price-and-Check Model

This section describes the branch-and-price-and-check (BPC) approach to the VRPLC. The BPC algorithm builds upon the ideas of the branch-and-cut-and-price model of the VRPPDTW [17], which employs column generation to produce routes, and a separation subproblem to generate valid inequalities that forbid certain classes of infeasible routes. Although column generation itself can solve the problem, the cuts add problem-specific knowledge and greatly improve convergence. In other words, these cuts are *implied cuts* since they are not necessary to correctly formulate the problem.

The BPC model follows a similar approach; the difference is that the cuts generated by the separation subproblem enforce the location resource constraints. These cuts are not implied cuts since they are necessary to solve the problem correctly, and they do not appear elsewhere in the problem. In that sense, BPC is related to Benders decomposition with the difference that the subproblem is solved at each node of the branch-and-price tree.

The rest of this section is organized as follows. Sections 6.1 to 6.3 introduce the master, pricing, and separation problems. Section 6.4 discusses the search tree and branching rules.

6.1 The Master Problem

The master problem, depicted in Figure 3, is the linear relaxation of a set partitioning problem. It selects a subset of routes from a main pool Ω of routes such that this subset satisfies certain constraints. The variable x_r denotes whether route $r \in \Omega$ is selected. The total cost of the subset is minimized by the objective function (Eq. (53)), where c_r is the cost of route r.

Constraint (54) ensures that all pickup requests are visited. The coefficient $a_{i,r}$ is equal to 1 if route r visits request $i \in \mathcal{P}$ and equal to 0 otherwise. For column generation models of VRPs, the request cover constraints are usually written as inequalities, rather than strict equalities, because the linear relaxation of the set covering problem with inequalities is easier to solve than the set

partitioning problem [14]. For the VRPLC, the cover constraints are expressed as equalities because of the location resource constraints. Indeed, covering a request multiple times would artificially inflate the number of resources required, which should be avoided.[1] Constraint (55) imposes the nogood cuts. A nogood cut has an associated set B of arcs, and coefficient B_r that denotes the number of arcs in B that are traversed by route r. When a set of routes is determined to be infeasible by the separation subproblem, one nogood cut is added to the master problem, with B containing the arcs traversed by the routes in this set. Hence, the nogood cut prohibits this set of routes by allowing at most $|B| - 1$ of their arcs to be used in any feasible solution.

6.2 The Pricing Subproblem

The pricing subproblem generates routes to add to Ω by solving an elementary resource-constrained shortest path problem using a labeling algorithm (e.g., [9, 17]). Each route must:

1. leave the start node, visit a number of requests, and return to the end node;
2. satisfy the service start time window, travel time, load, and pickup-delivery constraints; and
3. have negative cost with respect to the reduced cost matrix $\overline{d}_{i,j}$, which is defined as

$$\overline{d}_{i,j} = \begin{cases} d_{i,j} - \pi_i + \sum_{B \in \mathcal{B}} 1_{B,i,j} \mu_B, & \forall i \in \mathcal{P}, j \in \mathcal{N}, \\ d_{i,j} \quad\quad + \sum_{B \in \mathcal{B}} 1_{B,i,j} \mu_B, & \forall i \in \mathcal{N} \setminus \mathcal{P}, j \in \mathcal{N}, \end{cases}$$

   where $\mathcal{P}$ is the set of pickups, $\mathcal{N}$ is the set of all nodes (pickups, deliveries, one start node, and one end node), $d_{i,j}$ is the distance cost from $i \in \mathcal{N}$ to $j \in \mathcal{N}$, π_i is the dual value of Constraint (54), $1_{B,i,j}$ is equal to 1 if the arc (i, j) appears in B, and equal to 0 otherwise, and μ_B is the dual value of Constraint (55),

The labeling algorithm begins with a label at the start node. This label represents a subpath consisting of only the start node. The label is then extended to each pickup node in turn, giving subpaths consisting of the start node and one pickup node. Provided that these subpaths are feasible with respect to the constraints mentioned above, their corresponding labels are extended to other nodes. This process is repeated until a number of subpaths reach the end node, or an early termination criterion stops the algorithm.

[1] An alternative approach would be to transfer only one of these requests to the separation subproblem but this would introduce some random choices in the communication between the master and the subproblem.

Name	Description
$T \in \{1, \ldots, \infty\}$	Time horizon.
$\mathcal{T} = \{0, \ldots, T\}$	Set of time values.
$V \in \{1, \ldots, \infty\}$	Number of vehicles.
$R \in \{1, \ldots, \infty\}$	Total number of requests.
$\mathcal{R} = \{1, \ldots, R\}$	Set of all requests.
$\mathcal{S} = \{R+1, \ldots, R+V\}$	Set of vehicle start nodes.
$\mathcal{E} = \{R+V+1, \ldots, R+2V\}$	Set of vehicle end nodes.
$\mathcal{N} = \mathcal{R} \cup \mathcal{S} \cup \mathcal{E}$	Set of all nodes.
$L \in \{1, \ldots, \infty\}$	Number of locations, excluding the depot location.
$\mathcal{L} = \{1, \ldots, L\}$	Set of locations.
$C_l \in \{1, \ldots, \infty\}$	Resource capacity of location $l \in \mathcal{L}$.
$\mathcal{R}_l = \{i \in \mathcal{R} : l(i) = l\}$	Requests at location $l \in \mathcal{L}$.
$l(i) \in \mathcal{L}$	Location of $i \in \mathcal{R}$.
$d(i, j) \in \mathcal{T}$	Distance and travel time from $i \in \mathcal{N}$ to $j \in \mathcal{N}$.
$a(i) \in \mathcal{T}$	Earliest start of service at $i \in \mathcal{N}$.
$b(i) \in \mathcal{T}$	Latest start of service at $i \in \mathcal{N}$.
$t(i) \in \mathcal{T}$	Service duration of $i \in \mathcal{N}$.
$\text{succ}(i) \in \mathcal{R} \cup \mathcal{E}$	Successor node of $i \in \mathcal{R} \cup \mathcal{S}$ as per the routes in the master problem.
$\text{arr}(i) \in \mathcal{T}$	Arrival time at $i \in \mathcal{N}$.
$\text{serv}(i) \in \{a(i), \ldots, b(i)\}$	Start of service at $i \in \mathcal{N}$.
$\text{dep}(i) \in \mathcal{T}$	Departure time at $i \in \mathcal{N}$.

Table 3 Data and decision variables in the separation subproblem.

6.3 The Separation Subproblem

This section describes the separation subproblem for both service and presence resource constraints. The subproblem is modeled using constraint programing and contains similar constraints to those in the constraint programming model.

6.3.1 Service Resources

For the case of service resources, the constraints in the separation subproblem are extracted from the time and scheduling constraints of the CP model of the VRPLC. Table 3 lists the data and decision variables. The number V of vehicles and the successor variables $\text{succ}(\cdot)$ are not decision variables; they are fixed according to the routes selected by the master problem. Figure 4 lists the constraints. The decision variables are the arrival, service and departure times. Constraints (57) to (60) are the time constraints and serve as activity precedence constraints. Constraint (61) schedules the resources.

6.3.2 Presence Resources

Consider now the separation subproblem for presence resources. Although it is possible to implement Constraint (31) in this subproblem, it is more effective to preprocess the routes selected by the master program and introduce the

$$\text{arr}(i) \leq \text{serv}(i), \qquad \forall i \in \mathcal{R}, \quad (57)$$
$$\text{serv}(i) + t(i) \leq \text{dep}(i), \qquad \forall i \in \mathcal{R}, \quad (58)$$
$$\text{arr}(i) = \text{serv}(i) = \text{dep}(i), \qquad \forall i \in \mathcal{S} \cup \mathcal{E}, \quad (59)$$
$$\text{dep}(i) + d(i, \text{succ}(i)) = \text{arr}(\text{succ}(i)), \qquad \forall i \in \mathcal{R} \cup \mathcal{S}, \quad (60)$$
$$\textsc{Cumulative}(\{\text{serv}(i) : i \in \mathcal{R}_l\}, \{t(i) : i \in \mathcal{R}_l\}, \mathbf{1}, C_l), \qquad \forall l \in \mathcal{L}. \quad (61)$$

Fig. 4 Constraints of the separation subproblem for service resources.

Name	Description
$\mathcal{K}$	Set of visits.
$l(k) \in \mathcal{L}$	Location of visit $k \in \mathcal{K}$.
$\mathcal{K}_l = \{k \in \mathcal{K} : l(k) = l\}$	Set of visits to location $l \in \mathcal{L}$.
$\text{first}(k) \in \mathcal{R}$	First request of the visit $k \in \mathcal{K}$.
$\text{last}(k) \in \mathcal{R}$	Last request of the visit $k \in \mathcal{K}$.
$\text{start}(k) \in \mathcal{T}$	Start time of visit $k \in \mathcal{K}$.
$\text{end}(k) \in \mathcal{T}$	End time of visit $k \in \mathcal{K}$.
$\text{dur}(k) \in \mathcal{T}$	Duration of visit $k \in \mathcal{K}$.

Table 4 Additional data and decision variables for visits in the separation subproblem for presence resources.

concept of a *location visit* to reduce the number of activities in the subproblem. A location visit starts when a vehicle arrives at a location and lasts until the vehicle departs from the location. In other words, a location visit is a sequence of requests at a given location. The start time of a visit is the arrival time at the first request in the visit, and the end time is the departure time at the last request in the visit. The visits are transferred to the subproblem as data, but the arrival and departure times of the visits are variables.

The visits require additional data and decision variables, which are listed in Table 4. The subproblem is shown in Figure 5. Constraints (62) to (65) are the time constraints. Constraints (66) and (67) link the start and end times of a visit to the arrival and departure times at the first and last requests of the visit. Constraint (68) calculates the duration of a visit, and Constraint (69) is the cumulative resource constraint. Unlike for service resources, the durations of the visits are variables.

6.4 The BPC Search Algorithm

This section presents the BPC algorithm, which integrates the components presented earlier. The algorithm, described in Figure 6, begins by choosing an open node to solve (step 1). Nodes are chosen using two alternating heuristics. The first heuristic chooses the node with the largest number of flow variables fixed by branching, and the second heuristic selects the node with the smallest

$$\text{arr}(i) \leq \text{serv}(i), \qquad \forall i \in \mathcal{R}, \quad (62)$$
$$\text{serv}(i) + t(i) \leq \text{dep}(i), \qquad \forall i \in \mathcal{R}, \quad (63)$$
$$\text{arr}(i) = \text{serv}(i) = \text{dep}(i), \qquad \forall i \in \mathcal{S} \cup \mathcal{E}, \quad (64)$$
$$\text{dep}(i) + d(i, \text{succ}(i)) = \text{arr}(\text{succ}(i)), \qquad \forall i \in \mathcal{R} \cup \mathcal{S}, \quad (65)$$
$$\text{start}(k) = \text{arr}(\text{first}(k)), \qquad \forall k \in \mathcal{K}, \quad (66)$$
$$\text{end}(k) = \text{dep}(\text{last}(k)), \qquad \forall k \in \mathcal{K}, \quad (67)$$
$$\text{dur}(k) = \text{end}(k) - \text{start}(k), \qquad \forall k \in \mathcal{K}, \quad (68)$$
$$\textsc{Cumulative}(\{\text{start}(k) : k \in \mathcal{K}_l\}, \{\text{dur}(k) : k \in \mathcal{K}_l\}, \mathbf{1}, C_l), \qquad \forall l \in \mathcal{L}. \quad (69)$$

Fig. 5 Constraints of the separation subproblem for presence resources.

1. **Node Selection:** Select an open node, otherwise, terminate if no open nodes remain.
2. **Master:** Solve the master problem in Figure 3.
3. **Separation:** If $\sum_{r \in \Omega}(x_r = 1) \geq \min_{l \in \mathcal{L}} C_l$, solve the separation subproblem described in Figure 4 (or Figure 5). If the separation subproblem is infeasible, generate a nogood cut, and go back to step 2.
4. **Feasible Solution:** If all x_r variables are integral, a feasible solution is found.
5. **Pricing:** Solve the pricing subproblem described in Section 6.2. If at least one new route is generated, go back to step 2.
6. **Branching:** If the lower bound given by the master problem is smaller than the upper bound, and there is at least one route r with $0 < x_r < 1$, select one such route $r\prime = (i_1, i_2, \ldots, i_{n-1}, i_n)$. Create n open nodes, one for each (possibly empty or full) prefix of r. In the open node corresponding to prefix $(i_1, i_2, \ldots, i_j)$, require all edges in the prefix and exclude the edge (i_j, i_{j+1}). Go to step 1.

Fig. 6 The branch-and-price-and-check algorithm.

lower bound. By alternating between these two heuristics, the BPC algorithm attempts to both find feasible solutions and improve the lower bound.

In the chosen node, the BPC algorithm solves the master problem (step 2). It then counts the number of integer routes, i.e., routes r with $x_r = 1$. If there are at least $c = \min_{l \in \mathcal{L}} C_l$ integer routes, the BPC algorithm solves the separation subproblem on these routes (step 3). It is unnecessary to solve the separation subproblem if there are fewer than c integer x_r variables as the location constraints are automatically satisfied. Note that the solution to the master problem may consist of fractional values for some routes, but only the integer routes are transferred to the separation subproblem.

If no feasible schedule exists for these routes, a nogood cut is added to the master problem, which is reoptimized. Otherwise, if all routes are integral, then the routes and the schedules form a solution (step 4). Since the algorithm may iterate through this step multiple times in a node, it is possible, though unlikely, that multiple solutions are found within one node.

The algorithm then proceeds to the pricing subproblem to generate new routes (step 5). If new routes are found, they are added to the master problem, which is reoptimized. When no new routes are found, the node completes.

Branching occurs if any route in the master problem is fractional, and the lower bound of the node is lower than the incumbent solution (step 6). The BPC algorithm selects a fractional route $r\prime = (i_1, i_2, \ldots, i_{n-1}, i_n)$ of length n, and creates n children, in which certain arcs must or must not be used. Each child corresponds to a prefix $(i_1, i_2, \ldots, i_j)$, in which the edges $(i_1, i_2), \ldots, (i_{j-1}, i_j)$ must be present and the edge (i_j, i_{j+1}) must be absent. This branching scheme is easily implemented by restricting the successor matrix. Similar branching rules have previously appeared in the literature [9]. The children nodes are added to the set of open nodes.

Note that the pricing algorithm may be terminated early to avoid the tailing-off effect that prevents convergence [6,15]. In this case, the lower bound can be computed using the dual variables and reduced costs.

7 Experimental Results

This section reports experimental results for the three approaches.

The Instances A set of hard random instances are created by independently generating five locations and 150 pickup and delivery requests, and then distributing these requests among the locations. Smaller instances are created by discarding some of the requests. Resource capacities between one and eight are tested for all instances.

The Implementations The MIP model is solved using Gurobi with default parameters. The CP model is solved using Chuffed, a state-of-the-art CP solver with nogood learning. The solver first branches on the successor variables and then on the time variables. The master problem of the BPC model is solved using Gurobi. The pricing subproblem is solved using a bespoke implementation of a labeling algorithm, and the separation subproblem is solved using Chuffed. The three problems within the BPC model are embedded in the search algorithm discussed in the previous section. All three models are run for two hours.

Summary of the Results Table 5 displays the results for service resources. The main findings are summarized as follows.

1. Both the MIP and CP models find optimal solutions to the instances with 10 pickup-delivery requests. Additionally, the MIP model finds feasible solutions to two of the instances with 20 requests, while the CP model fails to find any feasible solution to any other instance.
2. The BPC model finds optimal solutions to all instances with 10 and 20 requests, and to some instances with 40 and 80 requests.
3. The CP model proves infeasibility on the instances with 40 or more requests and resource capacities of 1. On these few problems, the scheduling is disjunctive, and the activities have fixed duration, which allow the learning CP solver to make strong deductions about the feasibility of the resource

schedules. The CP model is unable to prove infeasibility for higher resource capacities as the reasoning is weaker.
4. Neither the MIP nor the BPC model is able to prove infeasibility on any instance since the cumulative constraints in the MIP model and the nogood cuts in the BPC model are weak and do not prune a significant part of the search space.
5. The BPC solutions to the instances with 80 or fewer requests improve as the resource capacities increase and the resource constraints relax. This is a desirable property and indicates that the BPC algorithm deals with weaker capacity constraints effectively. This behavior is less apparent in the larger problems, but in these experiments, the time limit is not scaled with the size of the instances.
6. Similarly, the running times of the BPC approach for instances with up to 40 requests decreases as the resource constraints relax. The problems with 80 requests also exhibit this behavior. This is shown via the improved optimality gap instead of the run time (since the BPC approach cannot prove optimality within the time limit). The optimality gaps are shown in Figure 7.
7. For the given time limit, the optimality gap is typically under 8%, and in most cases, below 5%, whenever the BPC model finds a feasible solution
8. Except for the proofs of infeasibility by the CP model, the BPC model outperforms both the MIP and CP models on all instances.

Table 6 shows the results for the instances with presence resources. Some remarks are below:

1. The CP model is unable to prove infeasibility for any of the instances because the durations in the cumulative constraint are variable, which results in weaker propagation. It may be beneficial to add the service resources cumulative constraint as a redundant constraint since, as previously discussed, the service resources are fully contained in the presence resources.
2. For the problems with 100 or fewer requests, the optimality gap is under 3% if the BPC model is able to find a feasible solution.
3. The BPC model outperforms the MIP and CP models on all instances.

The BPC approach exploits the orthogonal strengths of mathematical programming and constraint programming by using column generation to produce routes and dual bounds, and CP to check the feasibility of the resource constraints. These results demonstrate that the BPC model outperforms both the MIP and CP models on all instances besides those with service resource capacities of 1.

8 Conclusion

This paper develops the Vehicle Routing Problem with Location Congestion, which overlays a Resource-Constrained Project Scheduling Problem on a traditional Vehicle Routing Problem. Two types of location resources are considered: service resources are used while requests are in service, and presence

		MIP			CP		BPC		
P	C_l	Obj	Time	Gap	Obj	Time	Obj	Time	Gap
10	1	479	354	0.0%	479	52	479	0	0.0%
	2	479	1572	0.0%	479	58	479	0	0.0%
20	1	-	-	-	-	-	811	417	0.0%
	2	929	-	66.1%	-	-	771	0	0.0%
	3	882	-	64.1%	-	-	771	0	0.0%
	4	-	-	-	-	-	771	0	0.0%
	5	-	-	-	-	-	771	0	0.0%
40	1	-	-	-	-	1282	-	-	-
	2	-	-	-	-	-	1243	-	7.6%
	3	-	-	-	-	-	1162	11	0.0%
	4	-	-	-	-	-	1162	10	0.0%
	5	-	-	-	-	-	1162	9	0.0%
	6	-	-	-	-	-	1162	8	0.0%
	7	-	-	-	-	-	1162	7	0.0%
	8	-	-	-	-	-	1162	4	0.0%
60	1	-	-	-	-	0	-	-	-
	2	-	-	-	-	-	-	-	-
	3	-	-	-	-	-	1657	-	3.7%
	4	-	-	-	-	-	1634	-	2.3%
	5	-	-	-	-	-	1624	-	1.6%
	6	-	-	-	-	-	1624	-	1.6%
80	1	-	-	-	-	0	-	-	-
	2	-	-	-	-	-	-	-	-
	3	-	-	-	-	-	-	-	-
	4	-	-	-	-	-	2103	-	7.9%
	5	-	-	-	-	-	1958	-	1.2%
	6	-	-	-	-	-	1955	-	0.8%
	7	-	-	-	-	-	1955	-	0.8%
	8	-	-	-	-	-	1955	5278	0.0%
100	1	-	-	-	-	0	-	-	-
	2	-	-	-	-	-	-	-	-
	3	-	-	-	-	-	-	-	-
	4	-	-	-	-	-	-	-	-
	5	-	-	-	-	-	2401	-	5.7%
	6	-	-	-	-	-	2385	-	4.9%
	7	-	-	-	-	-	2294	-	1.2%
	8	-	-	-	-	-	2299	-	1.4%
150	1	-	-	-	-	1	-	-	-
	2	-	-	-	-	-	-	-	-
	3	-	-	-	-	-	-	-	-
	4	-	-	-	-	-	-	-	-
	5	-	-	-	-	-	-	-	-
	6	-	-	-	-	-	3290	-	100.0%
	7	-	-	-	-	-	3307	-	100.0%
	8	-	-	-	-	-	3218	-	100.0%

Table 5 Solutions from the three models for service resources. The first two columns report the number of pickups, and the resource capacity of each location. For each of the three models, the table shows the best solution and the run time, which is the time when the solver proves optimality or infeasibility. For the MIP and BPC models, the table also shows the optimality gap. Some instances with higher resource capacity are omitted if increasing the resource capacity does not improve the objective value.

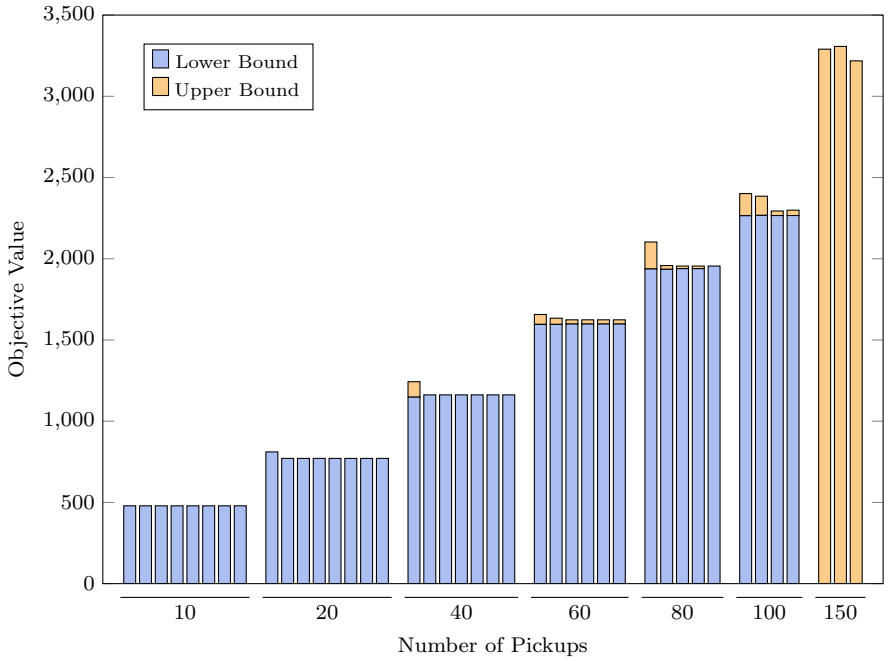

Fig. 7 Gap between the lower bound and best solution from the BPC method for instances with increasing service resource capacity.

resources are used whenever a vehicle is present at a location. The problem is formulated as a mixed integer programming (MIP) model, a constraint programming (CP) model and a branch-and-price-and-check (BPC) model. In the BPC model, the pricing subproblem generates routes for the master problem, and the separation subproblem verifies the feasibility of the routes selected by the master problem against the location resource constraints.

Empirical results indicate that the BPC approach finds feasible solutions to instances with up to 150 pickup-delivery requests and proves optimality on instances with up to 80 pickup-delivery requests. It outperforms both the MIP and CP formulations, which are unable to scale to instances larger than 10 pickup-delivery requests. These results highlight the benefits of integrating the unique strengths of mathematical programming and constraint programming.

Our future work will investigate stronger nogood cuts in the master problem and whether it is possible to translate temporal nogoods generated by the learning solver in the separation subproblem to route nogoods in the master problem, and to retain nogoods in the separation subproblem from one call to the next.

		MIP			CP		BPC		
P	C_l	Obj	Time	Gap	Obj	Time	Obj	Time	Gap
10	1	479	396	0.0%	479	64	479	0	0.0%
	2	479	717	0.0%	479	77	479	0	0.0%
	3	479	1071	0.0%	479	63	479	0	0.0%
	4	479	1630	0.0%	479	57	479	0	0.0%
	5	479	1834	0.0%	479	62	479	0	0.0%
	6	479	1373	0.0%	479	81	479	0	0.0%
	7	479	736	0.0%	479	64	479	0	0.0%
	8	479	1710	0.0%	479	57	479	0	0.0%
20	2	-	-	-	-	-	771	0	0.0%
	3	-	-	-	-	-	771	0	0.0%
	4	-	-	-	-	-	771	0	0.0%
	5	-	-	-	-	-	771	0	0.0%
	6	-	-	-	-	-	771	0	0.0%
	7	-	-	-	-	-	771	0	0.0%
	8	-	-	-	-	-	771	0	0.0%
40	3	-	-	-	-	-	1162	8	0.0%
	4	-	-	-	-	-	1162	8	0.0%
	5	-	-	-	-	-	1162	7	0.0%
	6	-	-	-	-	-	1162	7	0.0%
	7	-	-	-	-	-	1162	6	0.0%
	8	-	-	-	-	-	1162	3	0.0%
60	4	-	-	-	-	-	1639	-	2.7%
	5	-	-	-	-	-	1624	-	1.6%
	6	-	-	-	-	-	1624	-	1.6%
	7	-	-	-	-	-	1624	-	1.6%
	8	-	-	-	-	-	1624	-	1.6%
80	6	-	-	-	-	-	1955	3516	0.0%
	7	-	-	-	-	-	1955	-	0.8%
	8	-	-	-	-	-	1974	-	2.0%
100	6	-	-	-	-	-	2316	-	2.2%
	7	-	-	-	-	-	2298	-	1.4%
	8	-	-	-	-	-	2292	-	1.2%
150	7	-	-	-	-	-	3294	-	100.0%
	8	-	-	-	-	-	3262	-	100.0%

Table 6 Solutions from the three models for presence resources. Instances with low resource capacity are omitted from this table if neither a feasible solution nor a proof of infeasibility is found.

Acknowledgments

We would like to thank the reviewers for their constructive comments and suggestions.

References

1. Bard, J.F., Kontoravdis, G., Yu, G.: A branch-and-cut procedure for the vehicle routing problem with time windows. Transportation Science **36**(2), pp. 250–269 (2002)
2. Beck, J.: Checking-up on branch-and-check. In: D. Cohen (ed.) Principles and Practice of Constraint Programming – CP 2010, *Lecture Notes in Computer Science*, vol. 6308, pp. 84–98. Springer Berlin Heidelberg (2010)
3. Beck, J., Prosser, P., Selensky, E.: On the reformulation of vehicle routing problems and scheduling problems. In: S. Koenig, R. Holte (eds.) Abstraction, Reformulation, and Approximation, *Lecture Notes in Computer Science*, vol. 2371, pp. 282–289. Springer Berlin Heidelberg (2002)
4. Beck, J.C., Prosser, P., Selensky, E.: Vehicle routing and job shop scheduling: What's the difference? In: ICAPS, pp. 267–276 (2003)
5. Bent, R., Van Hentenryck, P.: A two-stage hybrid local search for the vehicle routing problem with time windows. Transportation Science **38**(4), 515–530 (2004)
6. Desaulniers, G., Desrosiers, J., Solomon, M.: Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems. In: Essays and Surveys in Metaheuristics, *Operations Research/Computer Science Interfaces Series*, vol. 15, pp. 309–324. Springer US (2002)
7. Desaulniers, G., Desrosiers, J., Solomon, M.M.: Column generation, vol. 5. Springer (2005)
8. Drexl, M.: Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints. Transportation Science **46**(3), 297–316 (2012)
9. Dumas, Y., Desrosiers, J., Soumis, F.: The pickup and delivery problem with time windows. European Journal of Operational Research **54**(1), 7–22 (1991)
10. Hachemi, N.E., Gendreau, M., Rousseau, L.M.: A heuristic to solve the synchronized log-truck scheduling problem. Computers & Operations Research **40**(3), 666 – 673 (2013). Transport Scheduling
11. Hempsch, C., Irnich, S.: Vehicle routing problems with inter-tour resource constraints. In: B. Golden, S. Raghavan, E. Wasil (eds.) The Vehicle Routing Problem: Latest Advances and New Challenges, *Operations Research/Computer Science Interfaces*, vol. 43, pp. 421–444. Springer US (2008)
12. Hooker, J.: Logic-based methods for optimization. In: A. Borning (ed.) Principles and Practice of Constraint Programming, *Lecture Notes in Computer Science*, vol. 874, pp. 336–349. Springer Berlin Heidelberg (1994)
13. Hooker, J.N., Mitchell, J.E.: Integrated methods for optimization. SIAM review **50**(1), 183 (2008)
14. Kallehauge, B., Larsen, J., Madsen, O.B., Solomon, M.M.: Vehicle routing problem with time windows. In: G. Desaulniers, J. Desrosiers, M.M. Solomon (eds.) Column Generation, pp. 67–98. Springer US (2005)
15. Lübbecke, M.E., Desrosiers, J.: Selected topics in column generation. Operations Research **53**(6), 1007–1023 (2005)
16. Rix, G., Rousseau, L.M., Pesant, G.: A column generation algorithm for tactical timber transportation planning. Journal of the Operational Research Society **66**(2), 278–287 (2015)
17. Ropke, S., Cordeau, J.F.: Branch and cut and price for the pickup and delivery problem with time windows. Transportation Science **43**(3), pp. 267–286 (2009)
18. Ropke, S., Cordeau, J.F., Laporte, G.: Models and branch-and-cut algorithms for pickup and delivery problems with time windows. Networks **49**(4), 258–272 (2007)
19. Shaw, P.: Using constraint programming and local search methods to solve vehicle routing problems. In: M. Maher, J.F. Puget (eds.) Principles and Practice of Constraint Programming — CP98, *Lecture Notes in Computer Science*, vol. 1520, pp. 417–431. Springer Berlin Heidelberg (1998)
20. Thorsteinsson, E.: Branch-and-check: A hybrid framework integrating mixed integer programming and constraint logic programming. In: T. Walsh (ed.) Principles and Practice of Constraint Programming — CP 2001, *Lecture Notes in Computer Science*, vol. 2239, pp. 16–30. Springer Berlin Heidelberg (2001)
21. Toth, P., Vigo, D.: The Vehicle Routing Problem. Society for Industrial and Applied Mathematics, Philadelphia, PA (2002)