Optim Lett (2013) 7:1569–1581
DOI 10.1007/s11590-012-0570-9

ORIGINAL PAPER

Branch-cut-and-price for the vehicle routing problem with simultaneous pickup and delivery

Anand Subramanian · Eduardo Uchoa · Artur Alves Pessoa · Luiz Satoru Ochi

Received: 30 September 2011 / Accepted: 21 September 2012 / Published online: 21 October 2012

Abstract This work proposes a Branch-cut-and-price (BCP) approach for the Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD). We also deal with a particular case of the VRPSPD, known as the Vehicle Routing Problem with Mixed Pickup and Delivery. The BCP algorithm was tested in well-known benchmark instances involving up to 200 customers. Four instances were solved for the first time and some LBs were improved.

Keywords Vehicle routing · Simultaneous pickup and delivery · Column generation · Cutting planes

1 Introduction

Let $G = (V, E)$ be an undirected complete graph, where $V = \{0, 1, \ldots, n\}$ is the set of vertices and E is the set of edges. Vertex 0 represents the depot and the remaining ones the customers. Each edge $\{i, j\} \in E$ has a non-negative cost c_{ij} and each customer $i \in V' = V - \{0\}$ has non-negative integer demands d_i for delivery and p_i for pickup.

A. Subramanian (✉)
Departamento de Engenharia de Produção, Centro de Tecnologia,
Universidade Federal da Paraíba, Campus I-Bloco G, Cidade Universitária, João Pessoa,
PB 58051-970, Brazil
e-mail: anand@ct.ufpb.br; anandsubraman@gmail.com

A. Subramanian · L. S. Ochi
Instituto de Computação, Universidade Federal Fluminense, Rua Passo da Pátria 156,
Bloco E-3ºandar, São Domingos, Niterói, RJ 24210-240, Brazil

E. Uchoa · A. A. Pessoa
Departamento de Engenharia de Produção, Universidade Federal Fluminense, Rua Passo da Pátria 156,
Bloco E-4ºandar, São Domingos, Niterói, RJ 24210-240, Brazil

Consider also a fleet of identical vehicles with integer capacity Q. The Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD) consists in designing a set of routes that minimizes the sum of the travel costs in such a way that: (1) every route starts and ends at the depot; (2) the vehicle's capacity is not exceeded in any part of a route; (3) each customer is visited exactly once. This problem is $\mathcal{NP}$-hard since it includes the Capacitated Vehicle Routing Problem (CVRP) as a special case when all the pickup (or delivery) demands are equal to zero. In this work we also consider a particular case of the VRPSPD, known as the Vehicle Routing Problem with Mixed Pickup and Delivery (VRPMPD) or the Vehicle Routing Problem with Mixed Backhauls. In the VRPMPD, each customer $i \in V'$ either have a pickup or a delivery demand, i.e., if $d_i > 0$, then $p_i = 0$ and vice-versa.

Berbeglia et al. [1] proposed a general scheme for classifying Pickup and Delivery Problems (PDPs) based on the characteristic of each problem. According to their framework, the VRPSPD is a multi-vehicle one-to-many-to-one PDP with combined demands, while the VRPMPD is referred as the multi-vehicle one-to-many-to-one PDP with single demands and mixed solutions.

A number of applications of the VRPSPD can be found in the beverage industry, where filled bottles are delivered while the empty ones are collected; in grocery stores, where pallets or containers are collected for re-use in merchandise transportation, etc. Moreover, some customers can demand that the delivery and pickup services should performed at the same time, since, if it is carried out separately, it may imply in additional costs and operational efforts for these customers.

The VRPSPD received a lot of attention from the literature in the past 10 years. Among the existing solution strategies, heuristic methods are by far the most usual approach applied for solving the problem. The recent works of Subramanian et al. [11] and Zachariadis and Kiranoudis [15] are those that produced the best known upper bounds for the VRPSPD. In spite of being a particular case of the VRPSPD, the VRPMPD was not dealt by many authors. To the best of our knowledge, the best known upper bounds were found by Gajpal and Abad [6].

In contrast to heuristic algorithms, there are few exact approaches devised for the VRPSPD. Dell'Amico et al. [3] proposed a branch-and-price algorithm that solved instances with up to 40 customers. Subramanian et al. [12] presented a Branch-and-Cut (BC) over one-commodity and two-commodity flow formulations and they were capable to solve instances with up to 100 customers. An extended version of this work that includes the results found for the VRPMPD can be found in [13]. More recently, Subramanian et al. [14] suggested a BC algorithm where the constraints that ensure that the capacity of the vehicle is not violated in the middle of a route are applied in a lazy fashion. The authors managed to solve VRPSPD and VRPMPD instances with up to 200 and 120 customers, respectively. Both BCs included cuts from the CVRPSEP library [7].

The objective of this work is to develop a Branch-cut-and-price (BCP) algorithm for the VRPSPD. The same algorithm is also applied to solve the VRPMD. To our knowledge this is the first attempt to solve those problems using a BCP approach. The algorithm presented in this work extends the one developed by Fukasawa et al. [5] for the CVRP as follows. The BCP algorithm presented in [5] uses a relaxation of the problem of finding an optimal route for a single vehicle as a subproblem. The feasible

solutions to this relaxation are referred to as q-routes. If q-routes were used for the VRPSPD (ignoring either pickups or deliveries), the same pricing algorithm, with time complexity of $O(n^2Q)$, can still be applied. However, the resulting lower bounds (LBs) turn out to be poor. To strengthen this relaxation, we introduce the pd-routes, which naturally extends the concept of q-routes to consider both pickup and delivery demands simultaneously. To optimize pd-routes, a straightforward generalization of the pricing algorithm presented in [5] is prohibitively expensive because its time complexity is $O(n^2Q^2)$. Hence, we develop a new dynamic programming algorithm that includes an efficient state-elimination scheme. This approach is a key contribution of this paper since it allowed for either solving or improving the LBs of open instances with up to 200 customers in a reasonable time.

The remainder of this paper is organized as follows. Section 2 describes the mathematical formulation. Section 3 explains the BCP algorithm. Section 4 contains the computational experiments. Section 5 presents the concluding remarks.

2 Mathematical formulation

Let x_{ij} be an integer variable representing the number of times that an edge $\{i, j\} \in E$ appears in a route. If a route is composed by a single customer then this number can be 2. Given a set $S \subseteq V'$, let $d(S)$ and $p(S)$ be the sum of the delivery and pickup demands, respectively, of all customers in S. Let $e(S) = \lceil d(S)/Q \rceil$ and $q(S) = \lceil p(S)/Q \rceil$. Let v be an integer variable representing the number of vehicles in the solution and $\bar{S}$ be the complementary set of S, plus the depot $\{0\}$. Consider the following Integer Program (IP):

$$\text{(F1)} \quad \min \sum_{i \in V} \sum_{j \in V, j > i} c_{ij} x_{ij} \tag{1}$$

$$\text{s.t.} \sum_{i \in V, i < k} x_{ik} + \sum_{j \in V, j > k} x_{kj} = 2 \qquad \forall k \in V' \tag{2}$$

$$\sum_{j \in V'} x_{0j} = 2v \tag{3}$$

$$\sum_{i \in S} \sum_{j \in \bar{S}, i < j} x_{ij} + \sum_{i \in \bar{S}} \sum_{j \in S, i < j} x_{ij} \geq 2e(S) \qquad \forall S \subseteq V' \tag{4}$$

$$\sum_{i \in S} \sum_{j \in \bar{S}, i < j} x_{ij} + \sum_{i \in \bar{S}} \sum_{j \in S, i < j} x_{ij} \geq 2q(S) \qquad \forall S \subseteq V' \tag{5}$$

$$\sum_{\{i,j\} \in R} x_{ij} \leq |R| - 1 \qquad \forall R \in \mathscr{R} \tag{6}$$

$$v \in \mathbb{Z}_+ \tag{7}$$

$$x_{ij} \in \{0, 1\} \qquad \forall \{i, j\} \in E, i > 0 \tag{8}$$

$$x_{ij} \in \{0, 1, 2\} \qquad \forall \{0, j\} \in E. \tag{9}$$

Constraints (2)–(4) are classical CVRP constraints. Constraints (5) are the rounded capacity cuts with respect to the pickup demands. Constraints (4) and (5) are not sufficient to ensure that the capacity of the vehicle is not exceeded in the middle of a route. Let $\mathscr{R}$ be the set composed by the edge-sets (excluding the edges adjacent to the depot) of those unfeasible routes. Constraints (6) are added in order to forbid such routes.

A q-route is a walk that starts at the depot vertex, traverses a sequence of customer vertices with total delivery (or pickup) demand at most Q, and returns to the depot. Some customers may be visited more than once, so the set of valid CVRP routes is strictly contained in the set of q-routes. For the VRPSPD, the direct use of q-routes does not yield strong relaxations because the pickup (or delivery) demands are ignored. Therefore, we introduce the concept of pd-route, which extends the definition of q-route by considering both pickup and delivery demands, as given by Definition 1.

Definition 1 A *pd-route* is a sequence of vertices $v_1, v_2, \ldots, v_k$ with $v_i \neq v_{i+1}$, $i = 1, 2, \ldots, k-1$, $v_1 = v_k = 0$, $v_i \in V'$, $i = 2, \ldots, k-1$, and $\sum_{j=2}^{i} p_{v_j} + \sum_{j=i+1}^{k-1} d_{v_j} \leq Q$, $i = 1, \ldots, k-1$. The cost of this pd-route is given by the sum of its edge costs, i.e., $\sum_{i=1}^{k-1} c_{v_i, v_{i+1}}$

Formulation F1 can be strengthened by adding an exponential number of variables corresponding to the pd-routes. Enumerate all possible pd-routes from 1 to P and let a_{ij}^t be the number of times an edge $\{i, j\}$ appears in the tth pd-route. Define λ_t as a positive variable associated to the tth pd-route. In order to improve F1 we add the following constraints:

$$\sum_{t=1}^{P} a_{ij}^t \lambda_t - x_{ij} = 0 \qquad \forall \{i, j\} \in E \tag{10}$$

$$\lambda_t \geq 0 \qquad (t = 1, \ldots, P) \tag{11}$$

Constraints (10) state that x is as a weighted sum of edge-incidence vectors of pd-routes. Notice that constraints (6) can be dropped from F1 since they are implied by (10). Hence, define F2 as the IP formulation composed by (1)–(5) and (7)–(11).

When solving the linear relaxation of F2 by column and row generation, a more compact Master LP is obtained if every occurrence x_{ij} in (2)–(5) is replaced by its equivalent given by (10). The resulting LP will be referred to as the Dantzig–Wolfe Master (DWM):

$$\text{(DWM)} \quad \min \sum_{t=1}^{P} \left(\sum_{i \in V} \sum_{j \in V, j > i} c_{ij} a_{ij}^t \right) \lambda_t \tag{12}$$

$$\text{s.t.} \quad \sum_{t=1}^{P} \left(\sum_{i \in V, i < k} a_{ik}^t + \sum_{j \in V, j > k} a_{kj}^t \right) \lambda_t = 2 \qquad \forall k \in V' \tag{13}$$

$$\sum_{t=1}^{P}\left(\sum_{j\in V'} a_{0j}^{t}\right)\lambda_t = 2v \tag{14}$$

$$\sum_{t=1}^{P}\left(\sum_{i\in S}\sum_{j\in\bar{S},i<j} a_{ij}^{t} + \sum_{i\in\bar{S}}\sum_{j\in S,i<j} a_{ij}^{t}\right)\lambda_t \geq 2e(S) \qquad \forall S \subseteq V' \tag{15}$$

$$\sum_{t=1}^{P}\left(\sum_{i\in S}\sum_{j\in\bar{S},i<j} a_{ij}^{t} + \sum_{i\in\bar{S}}\sum_{j\in S,i<j} a_{ij}^{t}\right)\lambda_t \geq 2q(S) \qquad \forall S \subseteq V' \tag{16}$$

$$\sum_{t=1}^{P} a_{ij}^{t}\lambda_t \leq 1 \qquad \forall\{i,j\} \in E, i > 0 \tag{17}$$

$$\lambda_t \geq 0 \qquad (t = 1,\ldots,P) \tag{18}$$

The reduced cost of a given λ variable is the sum of the reduced costs of the edges in the corresponding *pd*-route. Let μ, ν, π, ω and σ be the dual variables associated with constraints (13), (14), (15), (16) and (17), respectively. Define $S : \bar{S}$ as the set composed by the edges with one end in S and the other end in $\bar{S}$. The reduced cost $\bar{c}_{ij}$ of an edge is given by:

$$\bar{c}_{ij} = \begin{cases} c_{ij} - \mu_i - \mu_j - \sum\limits_{S|S:\bar{S}\ni\{i,j\}} \pi_S - \sum\limits_{S|S:\bar{S}\ni\{i,j\}} \omega_S - \sigma_{ij} & \forall\{i,j\} \in E, i > 0 \\ c_{ij} - \nu - \sum\limits_{S|j\in S} \pi_S - \sum\limits_{S|j\in S} \omega_S & \forall\{0,j\} \in E \end{cases}$$

In general, a cut of the form $\sum_{t=1}^{P}(\sum_{\{i,j\}\in E} h_{ij}a_{ij}^{t})\lambda_t \geq b$, with dual variable α, contributes with $-h_{ij}\alpha$ to the value of $\bar{c}_{ij}$.

3 The BCP algorithm

As already mentioned, the BCP algorithm presented in this work is mainly based on the one developed in [5] for the CVRP. Some elements such as branching rules, node selection strategy and strong branching scheme remained practically unchanged with respect to the original version. On the other hand, the column generation procedure had to be completely redesigned in order to deal with both pickup and delivery demands.

3.1 Pricing subproblem

The pricing subproblem can be seen as a shortest path problem with resource constraints, which is known to be strongly $\mathcal{NP}$-hard when cycles are forbidden. However, it can be solved in pseudo-polynomial time when *pd*-routes are considered. Therefore, the pricing subproblem consists of finding *pd*-routes with minimum reduced costs.

In this work, we use a dynamic programming based algorithm to solve the pricing subproblem. The load resources are denoted by $\mathscr{D}$ and $\mathscr{P}$, where the first indicates the total amount of load to be delivered, while the second indicates the cumulative amount of load that was collected. Resource $\mathscr{D}$ is initialized with a value less or equal to Q and resource $\mathscr{P}$ is initialized with 0. Every time a customer j is visited the value of $\mathscr{D}$ decreases by d_j and the value of $\mathscr{P}$ increases by p_j. A state is denoted by a triple $(j, \mathscr{D}, \mathscr{P})$ and it represents that customer j has just been visited, with $\mathscr{D}$ demand units delivered and $\mathscr{P}$ demand units picked up. For each state there is an associated label containing three elements: the customer identifier j; the minimum reduced cost $\bar{c}(j, \mathscr{D}, \mathscr{P})$ of a walk that starts at the depot and ends at state $(j, \mathscr{D}, \mathscr{P})$; and a pointer to a label representing the walk up to the previous customer.

The developed dynamic programming algorithm starts by considering all feasible expansions from the depot given by states $(0, \mathscr{D}, 0)$, $\mathscr{D} \in \{1, 2, \ldots, Q\}$, to each customer $j \in V'$, thus generating the states $(j, \mathscr{D}-d_j, p_j)$. This procedure is repeated for every new generated state and a label is updated when $\bar{c}(i, \mathscr{D}, \mathscr{P}) + \bar{c}_{ij} < \bar{c}(j, \mathscr{D} - d_j, \mathscr{P} + p_j)$. Of course, a state can be only expanded when its optimal walk had been already computed. The algorithm terminates when feasible expansions are no longer possible. We then extend the optimal walk of each customer to the depot in order to obtain a pd-route. The columns associated to pd-routes with negative reduced costs are added to the master problem. The computational complexity of the dynamic programming approach is $O(n^2Q^2)$.

A crucial aspect regarding the performance of our dynamic programming algorithm is the state elimination. A particular expansion can be avoided if we know in advance that the optimal walk of the state generated from such expansion will never be in a pd-route with negative reduced cost. In order to apply this type of strategy we must compute a valid LB for every possible state. In view of this, we have formulated a relaxed version of the pricing subproblem by only considering the delivery demands. We solve this relaxed problem also using dynamic programming to fill a matrix where each entry $M(j, \mathscr{D})$ corresponds to the least costly walk given by $\bar{c}(M(j, \mathscr{D}))$ starting from the depot and ending at customer j using total delivery demand exactly $\mathscr{D}$. Therefore we can avoid all expansions from a state $(i, \mathscr{D}, \mathscr{P})$ if $\bar{c}(i, \mathscr{D}, \mathscr{P}) + \bar{c}(M(i, \mathscr{D} + d_i)) \geq 0$. Furthermore, we can also avoid an expansion from state $(i, \mathscr{D}, \mathscr{P})$ to state $(j, \mathscr{D} - d_j, \mathscr{P} + p_j)$ if $\bar{c}(i, \mathscr{D}, \mathscr{P}) + \bar{c}_{ij} + \bar{c}(M(j, \mathscr{D})) \geq 0$.

The full execution of our dynamic programming algorithm can be very time consuming. A common speed-up strategy is to employ heuristic acceleration with a view of finding columns (pd-routes) with negative reduced costs more faster. We then only apply the full dynamic programming procedure when the heuristics fail to obtain pd-routes with negative reduced costs. Our heuristic strategy consists of a combination between two techniques: scaling and sparsification. Let $g > 1$ be the scaling factor. The scaling consists of modifying the customers original delivery and pickup demands to $d'_i(g) = \lceil d_i/g \rceil$, $i \in V'$ and $p'_i(g) = \lceil p_i/g \rceil$, $i \in V'$ respectively, and the original vehicle capacity to $Q' = \lfloor Q/g \rfloor$. Hence, the computational complexity will be in function of the capacity $Q' \leq Q$. The sparsification limits the number of possible expansions from a state $(i, \mathscr{D}, \mathscr{P})$ to those states associated with the vertices that are in the neighborhood of i. These neighbors are computed only once in the beginning

of the algorithm using the Minimum Spanning Tree based approach presented in [5].

Finally, in order to strengthen the formulation, we forbid *pd*-routes with 2-cycle. This is done by simply keeping the label of the second least costly walk of each state [2].

3.2 Cut generation

The rounded capacity cuts (15)–(16) and the bound cuts (17) are separated in every node of the branch-and-bound tree. In addition to these cuts, we also apply the strengthened comb cuts, but only at the root node. The capacity and comb cuts are separated using the CVRPSEP package [7]. We refer to [8] for details regarding the separation routines.

4 Computational experiments

The BCP algorithm was coded in C++ and executed in a single core of an Intel Core 2 Quad with 2.4 GHz and 4 GB of RAM running under Linux 64 bits. Linear Programs were solved using CPLEX 11.2. Three sets of instances were proposed in the VRPSPD literature, namely those of Dethloff [4], Salhi and Nagy [10] and Montané and Galvão [9]. The first set contains 40 instances with 50 customers, the second one 14 instances with 50–199 customers and the third one 18 instances with 100–400 customers, but in the latter we only considered the 12 instances with 100–200 customers. For the VRPMPD, we considered the set of instances suggested in [10] containing 21 instances with 50–199 customers.

4.1 Data preprocessing

The data of the VRPSPD instances proposed in [4,10] contain fractional values with 6 decimal places. This becomes an issue to the dynamic programming algorithm presented in Sect. 3.1, since the conversion of demands and capacities to integer values results in very large values of Q. In order to avoid this, we divide the capacity by a scaling factor r such that the new capacity Q/r is integral and not greater than 250. The demands are also divided by r and rounded down. In addition, for each delivery demand that is zero, we set it to one and increment the vehicle capacity by one unit. We also ensure that the total increase on the vehicle capacity does not exceed the maximum number of customers with incremented delivery demands that could fit into a single route (considering the pickup demands). These adaptations correspond to a (usually mild) relaxation of the pricing subproblem, allowing some routes that violate the true capacities. Since the rounded capacity cuts are applied over the original instance, we can ensure that the original vehicle capacity will be never violated in the depot. Nevertheless, these cuts are still not sufficient to prevent the algorithm of generating infeasible integer solutions, i.e., those containing at least one vehicle whose capacity is violated in the middle of the route. The cuts that ensure that the vehicle capacity is not exceeded in the middle of a route, i.e. those given by (6) are separated in a lazy fashion. Very few such cuts are needed in practice.

4.2 Results

In the following tables, **Instance** is the name of the instance, **Customers** corresponds to the number of customers, **#vehicles** is the number of vehicles in the best known solution or a LB on the number of vehicles, **Root LB** indicates the root LB, **Root Time** is the CPU time in seconds spent at the root node, **Tree size** is the number of nodes opened, **Total time** is the total CPU time in seconds of the BCP procedure, **Prev. LB** is the best known LB, **LB** is the LB determined by the BCP, **UB** is the upper bound reported in [11] and [6,14] for the VRPSPD and VRPMPD, respectively. Previously known optimal solutions are highlighted in boldface. Improved LBs and new optimal solutions are underlined.

We report the LBs obtained at the root node for all VRPSPD/VRPMPD instances. For those instances with a relatively large number of customers per vehicle a time limit of 2 h was imposed (CMT11X, CMT11Y, r201, rc101, r2_2_1, rc2_2_1 and CMT11H and CMT4T). Nevertheless, this limit was exceeded in some instances because we decided not to interrupt the execution of the algorithm during a column generation procedure. A comparison is performed with the root relaxations obtained by the BC presented in [14]. We only executed the full BCP algorithm in the instances where the BC has an inferior performance.

The average number of capacity and comb cuts generated at the root node of all instances (disregarding the interrupted runs) was 316.1 and 46.3, respectively.

4.2.1 VRPSPD

Table 1 shows the results found by the BCP algorithm on the set of instances proposed in [4]. It is noteworthy to mention that the optimal solutions of all instances of this set were found by the BC presented in [14]. From Table 1, it can be observed that the BCP algorithm had a considerable better performance in terms of both root LB and computational time in those cases where the average number of customers per vehicle (given by n/v) is smaller, i.e. instances SCA8-0, SCA8-1, ..., SCA8-9, CON8-0, CON8-1,..., CON8-9. The BC and BCP average gaps between the root LBs and the optimal solutions in such subset of 20 instances were 3.24 and 1.28 %, respectively. Moreover, the average computational time of the complete execution of BCP in these instances was 197 s, whereas for BC this value was 2,883 s [14].

Table 2 illustrates the results found in the instances suggested in [10]. In these set of instances, BC appears to be more suitable, in terms of overall performance, when compared to the BCP. Yet, the best known LBs of the instances CMT5X and CMT5Y were improved at the root node when solving such problems using the BCP algorithm. For the remaining the instances there is no advantage in applying the BCP instead of a pure BC. Moreover, closing the gap between the UB and the LB of instances CMT2X and CMT2Y, both with 75 customers and 6 vehicles, still remains a challenge.

Table 3 presents the results obtained in the set of benchmark instances proposed in [9]. As it happened in the set of instances of Dethloff [4], the BCP algorithm outperformed the BC on those instances where the average number of customers per vehicle is relatively small. Three instances containing 100 customers were solved to

Table 1 Results obtained on the instances of Dethloff [4]

Instance/ customers/ #vehicles	BC		BCP					Prev. LB	UB
	Root LB	Root time (s)	Root LB	Root time (s)	Tree size	Total time (s)	LB		
SCA3-0/50/4	619.21	5.18	620.87	149	–	–	–	635.62	**635.62**
SCA3-1/50/4	682.34	1.90	682.75	27.2	–	–	–	697.84	**697.84**
SCA3-2/50/4	659.34	0.60	659.34	64.8	–	–	–	659.34	**659.34**
SCA3-3/50/4	667.56	1.76	667.56	102	–	–	–	680.04	**680.04**
SCA3-4/50/4	676.56	6.75	675.97	51.6	–	–	–	690.50	**690.50**
SCA3-5/50/4	647.90	1.16	649.56	48.5	–	–	–	659.91	**659.91**
SCA3-6/50/4	625.60	1.29	625.16	67.0	–	–	–	651.09	**651.09**
SCA3-7/50/4	654.97	3.70	655.99	72.8	–	–	–	659.17	**659.17**
SCA3-8/50/4	688.21	1.97	690.29	161	–	–	–	719.48	**719.48**
SCA3-9/50/4	671.07	2.04	670.56	85.2	–	–	–	681.00	**681.00**
SCA8-0/50/9	926.03	18.8	946.50	2.91	63	76.2	961.50	961.50	**961.50**
SCA8-1/50/9	1,002.29	15.5	1,028.22	2.71	369	437	1,049.65	1,049.65	**1,049.65**
SCA8-2/50/9	1,013.37	19.8	1,026.07	2.90	307	270	1,039.64	1,039.64	**1,039.64**
SCA8-3/50/9	956.10	11.9	975.45	1.93	11	10.6	983.34	983.34	**983.34**
SCA8-4/50/9	1,027.24	25.9	1,052.79	1.11	51	34.4	1,065.49	1,065.49	**1,065.49**
SCA8-5/50/9	998.05	13.7	1,017.77	2.75	107	115	1,027.08	1,027.08	**1,027.08**
SCA8-6/50/9	948.26	12.8	964.03	3.56	19	20.8	971.82	971.82	**971.82**
SCA8-7/50/9	1,018.20	16.4	1,028.30	2.01	547	601	1,051.28	1,051.28	**1,051.28**
SCA8-8/50/9	1,038.86	6.15	1,057.44	1.85	35	53.7	1,071.18	1,071.18	**1,071.18**
SCA8-9/50/9	1,019.25	12.0	1,037.29	1.51	245	341	1,060.50	1,060.50	**1,060.50**
CON3-0/50/4	611.30	2.58	612.16	66.9	–	–	–	616.52	**616.52**
CON3-1/50/4	546.65	8.26	546.38	104	–	–	–	554.47	**554.47**
CON3-2/50/4	505.06	3.94	507.43	71.5	–	–	–	518.01	**518.01**
CON3-3/50/4	584.28	2.05	585.16	167	–	–	–	591.19	**591.19**
CON3-4/50/4	577.52	1.04	578.09	48.4	–	–	–	588.79	**588.79**
CON3-5/50/4	552.91	3.22	555.76	114	–	–	–	563.70	**563.70**
CON3-6/50/4	486.98	5.22	487.83	67.9	–	–	–	499.05	**499.05**
CON3-7/50/4	561.62	0.97	563.49	58.8	–	–	–	576.48	**576.48**
CON3-8/50/4	514.84	5.03	516.45	50.6	–	–	–	523.05	**523.05**
CON3-9/50/4	564.78	2.51	567.69	59.0	–	–	–	578.25	**578.25**
CON8-0/50/9	830.03	41.5	850.16	2.82	53	45.2	857.17	857.17	**857.17**
CON8–1/50/9	722.38	27.7	732.24	0.98	151	85.9	740.85	740.85	**740.85**
CON8-2/50/9	685.66	39.2	700.78	2.94	249	576	712.89	712.89	**712.89**
CON8-3/50/10	787.84	39.3	806.23	1.73	95	85.2	811.07	811.07	**811.07**
CON8-4/50/9	751.95	21.2	764.50	1.16	121	107	772.25	772.25	**772.25**
CON8-5/50/9	729.92	15.1	746.25	1.69	273	358	754.88	754.88	**754.88**

Table 1 continued

Instance/ customers/ #vehicles	BC		BCP					Prev. LB	UB
	Root LB	Root time (s)	Root LB	Root time (s)	Tree size	Total time (s)	LB		
CON8-6/50/9	648.64	18.7	666.09	1.42	151	202	678.92	678.92	**678.92**
CON8-7/50/9	793.50	15.5	804.51	1.35	25	35.7	811.96	811.96	**811.96**
CON8-8/50/9	744.52	26.8	761.79	2.93	31	66.2	767.53	767.53	**767.53**
CON8-9/50/9	773.65	45.0	798.12	1.61	419	422	809.00	809.00	**809.00**

Table 2 Results obtained on the VRPSPD instances of Salhi and Nagy [10]

Instance/ customers/ #vehicles	BC		BCP					Prev. LB	UB
	Root LB	Root time (s)	Root LB	Root time (s)	Tree size	Total time (s)	LB		
CMT1X/50/3	460.73	5.56	461.90	96.0	–	–	–	466.77	**466.77**
CMT1Y/50/3	460.69	15.5	460.86	238	–	–	–	466.77	**466.77**
CMT2X/75/6	658.38	100	661.11	94.2	–	–	–	666.57	684.21
CMT2Y/75/6	658.12	200	659.76	308	–	–	–	666.69	684.21
CMT3X/100/5	711.77	79.7	711.70	1,404	–	–	–	721.27	**721.27**
CMT3Y/100/5	711.89	140	711.49	4,519	–	–	–	721.27	**721.27**
CMT12X/100/5	635.52	50.3	638.72	3,130	–	–	–	643.76	662.22
CMT12Y/100/5	635.45	65.1	636.29	3,608	–	–	–	644.10	662.22
CMT11X/120/4	793.11	892	766.22	9,570	–	–	–	799.67	833.92
CMT11Y/120/4	793.54	1,098	765.47	9,421	–	–	–	799.02	833.92
CMT4X/150/7	826.74	1,950	829.58	11,433	–	–	–	831.18	852.46
CMT4Y/150/7	826.34	2,568	827.50	791,912	–	–	–	831.65	852.46
CMT5X/199/10	971.07	7,202	988.28	66,063	–	–	–	971.07	1,029.25
CMT5Y/199/10	937.66	7,248	980.63	260,737	–	–	–	953.56	1,029.25

optimality for the first time, namely r101, c101 and rc101. As a result, all the six 100-customer instances of this set are now solved. In addition, the LB of the instances r1_2_1, c1_2_1, c2_2_1 and rc1_2_1 were dramatically improved. On the other hand, the BCP algorithm failed completely to solve the linear relaxation of the instances r2_2_1 and rc2_2_1 (where there are about 40 customers per vehicle) in the time limit specified. These instances were solved to optimality by the pure BC.

Table 3 Results obtained on the instances of Montané and Galvão [9]

Instance/ customers/ #vehicles	BC		BCP					Prev. LB	UB
	Root LB	Root time (s)	Root LB	Root time (s)	Tree size	Total time (s)	LB		
r101/100/12	972.58	1,282	996.18	208	23,465	1.06×10^6	1,009.95	978.28	1,009.95
r201/100/3	664.80	42.9	661.95	9,968	–	–	–	666.20	**666.20**
c101/100/16	1,194.69	722	1,208.99	3.99	31,631	31,042	1,220.18	1,202.59	1,220.18
c201/100/5	657.97	24.6	659.11	17,086	–	–	–	662.07	**662.07**
rc101/100/10	1,028.06	1,067	1,049.84	78.5	653	17,491	1,059.32	1,041.06	1,059.32
rc201/100/3	671.84	6.37	668.84	15,021	–	–	–	672.92	**672.92**
r1_2_1/200/23	2,681.05	7,198	3,273.27	1,094	–	–	–	3,084.97	3,360.02
r2_2_1/200/5	1,656.62	4,477	–	–	–	–	–	1,665.58	**1,665.58**
c1_2_1/200/28	2,857.45	7,196	3,609.16	78.4	–	–	–	3,475.03	3,629.89
c2_2_1/200/9	1,638.99	7,197	1,704.00	97,487	–	–	–	1,647.83	1,726.59
rc1_2_1/200/23	2,656.78	7,196	3,241.98	888	–	–	–	3,093.30	3,306.00
rc2_2_1/200/5	1,551.54	1,932	–	–	–	–	–	1,560.00	**1,560.00**

4.2.2 VRPMPD

The results found in the VRPMPD instances generated in [10] are depicted in Table 4. The best known upper bounds that are not proven optimal were taken from [6]. However, the number of vehicles was not specified in the referred work. We report a LB on the number of vehicles in such cases. From Table 4, it is possible to verify that the BCP algorithm had an overall performance similar to the one observed in the VRPSPD instances of Salhi and Nagy [10] (see Table 2). The optimality of the instance CMT2T was proved for the first time and LBs of the instances CMT4T, CMT5H, CMT5Q and CMT5T were improved.

5 Concluding remarks

This work presented a BCP approach for the VRPSPD that is also capable of solving the VRPMPD. The proposed algorithm is an adaptation of the one developed in [5] for solving the CVRP. The fundamental difference of our BCP relies on the column generation procedure that was redesigned to deal with the delivery and pickup demands simultaneously. The BCP algorithm was tested in well-known benchmark instances and it was found capable of proving the optimality of 4 problems for the first time, where three of them are VRPSPD instances containing 100 customers and one of them corresponds to a VRPMPD instance containing 75 customers. Also, the best known LBs of other 10 instances involving 150–200 customers were improved. Moreover, the BCP clearly outperformed previous methods in a subset of instances proposed

Table 4 Results obtained on the VRPMPD instances of Salhi and Nagy [10]

Instance/ customers/ #vehicles	BC		BCP					Prev. LB	UB
	Root LB	Root time (s)	Root LB	Root time (s)	Tree size	Total time (s)	LB		
CMT1H/50/3	460.10	7.14	461.32	64.7	–	–	–	465.02	**465.02**
CMT1Q/50/4	488.15	5.65	488.26	31.2	–	–	–	489.74	**489.74**
CMT1T/50/5	512.61	4.98	515.20	5.81	–	–	–	520.06	**520.06**
CMT2H/75/6	643.28	101	648.63	174	–	–	–	653.79	662.63
CMT2Q/75/8	707.66	100	710.00	14.0	–	–	–	717.36	732.76
CMT2T/75/9	759.92	151	767.77	7.31	14,627	109,348	782.77	770.35	782.77
CMT3H/100/3	691.32	58.0	691.42	3,376	–	–	–	700.94	**700.94**
CMT3Q/100/6	744.50	95.5	744.84	854	–	–	–	747.15	**747.15**
CMT3T/100/5	784.13	342	784.75	346	–	–	–	798.07	**798.07**
CMT12H/100/5	623.00	40.3	623.33	881	–	–	–	629.37	**629.37**
CMT12Q/100/7	721.81	83.9	722.35	948	–	–	–	729.25	**729.25**
CMT12T/100/9	787.27	37.7	787.52	52.7	–	–	–	787.52	**787.52**
CMT11H/120/4	801.24	353	776.36	7,680	–	–	–	806.46	820.35
CMT11Q/120/6	928.28	653	899.45	7,831	–	–	–	939.36	**939.36**
CMT11T/120/7	984.81	359	985.75	2,884	–	–	–	989.32	998.80
CMT4H/150/6	799.19	1,122	795.22	8,465	–	–	–	804.10	831.39
CMT4Q/150/9	892.99	2,920	895.15	13,090	–	–	–	898.28	913.93
CMT4T/150/11	953.41	4,246	960.59	228	–	–	–	956.54	990.39
CMT5H/199/9	931.02	7,209	948.46	276,318	–	–	–	931.02	992.37
CMT5Q/199/12	1,056.74	7,206	1,069.91	97,259	–	–	–	1,056.74	1,134.72
CMT5T/199/15	1,145.61	7,192	1,175.62	755	–	–	–	1,145.63	1,232.08

in [4] and [9]. The main characteristic of such instances is the fact of having a small average number of customers per vehicle, which is known to be favorable to column generation based techniques.

Acknowledgments This work was partially supported by the following Brazilian research agencies: CNP_q, CAPES and FAPERJ.

References

1. Berbeglia, G., Cordeau, J.F., Gribkovskaia, I., Laporte, G.: Static pickup and delivery problems: a classification scheme and survey. Top **15**, 1–31 (2007)
2. Christofides, N., Mingozzi, A., Toth, P.: Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. Math. Program. **20**, 255–282 (1981)
3. Dell'Amico, M., Righini, G., Salani, M.: A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. Transp. Sci. **40**(2), 235–247 (2006)
4. Dethloff, J.: Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up. OR Spektr. **23**, 79–96 (2001)
5. Fukasawa, R., Longo, H., Lysgaard, J., de Aragão, P.M., Reis, M., Uchoa, E., Werneck, R.F.: Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Math. Program. **106**, 491–511 (2006)

6. Gajpal, Y., Abad, P.: An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup. Comput. Oper. Res. **36**(12), 3215–3223 (2009)
7. Lysgaard, J.: A package of separation routines for the capacited vehicle routing problem. Technical report. www.hha.dk/lys/CVRPSEP.htm (2003)
8. Lysgaard, J., Letchford, A.N., Eglese, R.W.: A new branch-and-cut algorithm for the capacitated vehicle routing problem. Math. Program. **100**, 423–445 (2004)
9. Montané, F.A.T., Galvão, R.D.: A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. Comput. Oper. Res. **33**(3), 595–619 (2006)
10. Salhi, S., Nagy, G.: A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. J. Oper. Res. Soc. **50**, 1034–1042 (1999)
11. Subramanian, A., Drummond, L.M.A., Bentes, C., Ochi, L.S., Farias, R.: A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. Comput. Oper. Res. **37**(11), 1899–1911 (2010)
12. Subramanian, A., Uchoa, E., Ochi, L.S.: New lower bounds for the vehicle routing problem with simultaneous pickup and delivery. In: Proceedings of the 9th International Symposium on Experimental Algorithms (SEA). Lecture Notes in Computer Science, pp. 276–287 (2010)
13. Subramanian, A., Uchoa, E., Ochi, L.S.: New lower bounds for the vehicle routing problem with simultaneous pickup and delivery. Technical Report 01/10, UFF, Niterói, Brazil (2010)
14. Subramanian, A., Uchoa, E., Pessoa, A.A., Ochi, L.S.: Branch-and-cut with lazy separation for the vehicle routing problem with simultaneous pickup and delivery. Oper. Res. Lett. **39**(5), 338–341 (2011)
15. Zachariadis, E.E., Kiranoudis, C.T.: A local search metaheuristic algorithm for the vehicle routing problem with simultaneous pick-ups and deliveries. Expert Syst. Appl. **38**(3), 2717–2726 (2011)