A Generic Exact Solver for Vehicle Routing and Related Problems

Artur Alves Pessoa, Ruslan Sadykov, Eduardo Uchoa, François Vanderbeck

A Generic Exact Solver for Vehicle Routing and Related Problems

Artur Pessoa[3], Ruslan Sadykov[1,2], Eduardo Uchoa[3], and François Vanderbeck[2,1]

[1] INRIA Bordeaux — Sud-Ouest, France
[2] University of Bordeaux, France
[3] Universidade Federal Fluminense, Brazil

Abstract. Major advances were recently obtained in the exact solution of Vehicle Routing Problems (VRPs). Sophisticated Branch-Cut-and-Price (BCP) algorithms for some of the most classical VRP variants now solve many instances with up to a few hundreds of customers. However, adapting and reimplementing those successful algorithms for other variants can be a very demanding task. This work proposes a BCP solver for a generic model that encompasses a wide class of VRPs. It incorporates the key elements found in the best recent VRP algorithms: ng-path relaxation, rank-1 cuts with limited memory, and route enumeration; all generalized through the new concept of "packing set". This concept is also used to derive a new branch rule based on accumulated resource consumption and to generalize the Ryan and Foster branch rule. Extensive experiments on several variants show that the generic solver has an excellent overall performance, in many problems being better than the best existing specific algorithms. Even some non-VRPs, like bin packing, vector packing and generalized assignment, can be modeled and effectively solved.

Keywords: Integer Programming · Column Generation · Routing.

1 Introduction

Since its introduction by Dantzig and Ramser [25], the Vehicle Routing Problem (VRP) has been one of the most widely studied in combinatorial optimization. Google Scholar indicates that 728 works containing both words "vehicle" and "routing" in the title were published only in 2017. VRP relevance stems from its direct use in the real systems that distribute goods and provide services, vital to the modern economies. Reflecting the large variety of conditions in those systems, the VRP literature is spread into dozens, perhaps hundreds, of variants. For example, there are variants that consider capacities, time windows, heterogeneous fleets, pickups and deliveries, optional customer visits, arc routing, etc.

In recent years, big advances in the exact solution of VRPs had been accomplished. A milestone was certainly the Branch-Cut-and-Price (BCP) algorithm of [45, 47], that could solve Capacitated VRP (CVRP) instances with up to 360

customers, a large improvement upon the previous record of 150 customers. That algorithm exploits many elements introduced by several authors, combining and enhancing them. In particular, the new concept of *limited memory cut* proved to be pivotal. Improvements of the same magnitude were later obtained for a number of classical variants like VRP with Time Windows (VRPTW) [46], Heterogeneous Fleet VRP (HFVRP) and Multi Depot VRP (MDVRP) [50], and Capacitated Arc Routing (CARP) [49]. For all those variants, instances with about 200 customers are now likely to be solved, perhaps in hours or even days. However, there is something even more interesting: typical instances with about 100 customers, that a few years ago would take hours, are solved in less than 1 minute. This means that many more real world instances can now be tackled by exact algorithms in reasonable times.

Unhappily, designing and coding each one of those complex and sophisticated BCPs has been a highly demanding task, measured on several work-months of a skilled team. In effect, this prevents the practical use of those algorithms in real world problems, that actually, seldom correspond exactly to one of the most classical variants. This work presents a framework that can handle most VRP variants found in the literature and can be used to model and solve many other new variants. In order to obtain state-of-the-art BCP performance, some key elements found in the best specific VRP algorithms had to be generalized. The new concept of *packing set* was instrumental for that.

The quest for general exact VRP algorithms can be traced back to Balinski and Quandt [8], where a set partitioning formulation valid for many variants was proposed. That formulation had only turned practical in the 1980's and 1990's, when the Branch-and-Price (BP) method was developed. At that time, it was recognized that the pricing subproblems could often be modeled as Resource Constrained Shortest Path (RCSP) problems and solved by labeling algorithms, leading to quite generic methods (for example, [27]). However, those BP algorithms only worked well on problems with "tightly constrained" routes, like VRPTW with narrow time windows. Many variants, including CVRP, were much better handled by Branch-and-Cut (BC) algorithms using problem-specific cuts (for example, [43]). In the late 2010's, after works like [33, 5, 38, 28, 56, 4, 14], it became clear that the combination of cut and column generation performs better than pure BP or pure BC on almost all problems. Until today, BCP remains the dominant VRP approach. A first attempt of a generic BCP was presented in [6], where 7 variants, all of them particular cases of the HFVRP, could be solved. Recently, [58] proposed a BCP for several particular cases of the HFVRP with time windows. The framework now proposed is far more generical than that.

2 The Basic Model

2.1 Graphs for RCSP Generation

Define directed graphs $G^k = (V^k, A^k)$, $k \in K$. Let $V = \cup_{k \in K} V^k$ and $A = \cup_{k \in K} A^k$. The graphs are not necessarily simple and may even have loops. Vertices and arcs in all graphs are distinct and carry the information about which

graph they belong: a vertex $v \in V$ belongs to $G^{k(v)}$ and an arc $a \in A$ belongs to $G^{k(a)}$. Each graph has special source and sink vertices: v^k_{source} and v^k_{sink}. Define a set R of resources, divided into *main resources* R^M and *secondary resources* R^N. For each r in R and $a \in A$, $q_{a,r} \in \mathbb{R}$ is the consumption of resource r in arc a. If $r \in R^N$, consumptions are unrestricted in sign. However, for $r \in R^M$, consumptions should be non-negative. Moreover, for any $k \in K$ there should not exist a cycle in G^k where the consumption of all main resources are zero. Finally, there are finite accumulated resource consumption intervals $[l_{a,r}, u_{a,r}]$, $a \in A$. Since in most applications these intervals are more naturally defined on vertices, we may define intervals $[l_{v,r}, u_{v,r}]$, $v \in V$, meaning that $[l_{a,r}, u_{a,r}] = [l_{v,r}, u_{v,r}]$ for every arc $a \in \delta^-(v)$ (i.e., entering v).

A resource constrained path $p = (v^k_{\text{source}} = v_0, a_1, v_1, \ldots, a_{n-1}, v_{n-1}, a_n, v_n = v^k_{\text{sink}})$ over a graph G^k, having $n \geq 1$ arcs, is feasible if: for every $r \in R$, the accumulated resource consumption $S_{j,r}$ at visit j, $0 \leq j \leq n$, where $S_{0,r} = 0$ and $S_{j,r} = \max\{l_{a_j,r}, S_{j-1,r} + q_{a_j,r}\}$, does not exceed $u_{a_j,r}$. Note that some feasible paths may not be elementary, some vertices or arcs being visited more than once. For each $k \in K$, let P^k denote the set of all feasible resource constrained paths in G^k. We will assume that each set P^k is finite, either because G^k is acyclic or because at least one main resource is defined on it. Define $P = \cup_{k \in K} P^k$. Again, a general path $p \in P$ carries the information of its graph, $G^{k(p)}$.

2.2 Formulation

The problem should be formulated as follows. There are variables x_j, $1 \leq j \leq n_1$, and variables y_s, $1 \leq s \leq n_2$. The first $\bar{n}_1$ x variables and the first $\bar{n}_2$ y variables are defined to be integer. Equations (1a) and (1b) define a general objective function and m general constraints over those variables, respectively. Constraints (1b) may even contain exponentially large families of cuts, provided that suitable procedures are given for their separation. However, by simplicity, we continue the presentation as if all the m constraints are explicitly defined. For each variable x_j, $1 \leq j \leq n_1$, $M(x_j) \subseteq A$ defines its *mapping* into a non-empty subset of the arcs. We remark that mappings do not need to be disjoint, the same arc can mapped to more than one variable x_j. Define $M^{-1}(a)$ as $\{j | 1 \leq j \leq n_1; a \in M(x_j)\}$. As not all arcs need to belong to some mapping, some M^{-1} sets may be empty. For each path $p \in P$, let λ_p be a non-negative integer variable; coefficient h^p_a indicates how many times a appears in p. The relation between variables x and λ is given by (1c). For each $k \in K$, L^k and U^k are given lower and upper bounds on number of paths in a solution.

$$\text{Min} \qquad \sum_{j=1}^{n_1} c_j x_j + \sum_{s=1}^{n_2} f_s y_s \tag{1a}$$

$$\text{S.t.} \qquad \sum_{j=1}^{n_1} \alpha_{ij} x_j + \sum_{s=1}^{n_2} \beta_{is} y_s \geq d_i, \qquad i = 1, \ldots, m, \tag{1b}$$

$$x_j = \sum_{k \in K} \sum_{p \in P^k} \left(\sum_{a \in M(x_j)} h^p_a \right) \lambda_p, \quad j = 1 \ldots, n_1, \tag{1c}$$

$$L^k \leq \sum_{p \in P^k} \lambda_p \leq U^k, \qquad k \in K, \tag{1d}$$

$$\lambda_p \in \mathbb{Z}_+, \qquad p \in P, \tag{1e}$$

$$x_j \in \mathbb{N},\, y_s \in \mathbb{N} \qquad j = 1, \ldots, \bar{n}_1,\, s = 1, \ldots, \bar{n}_2. \tag{1f}$$

Eliminating the x variables and relaxing the integrality constraints, the following LP is obtained:

$$\text{Min} \qquad \sum_{k \in K} \sum_{p \in P^k} \left(\sum_{j=1}^{n_1} c_j \sum_{a \in M(j)} h_a^p \right) \lambda_p + \sum_{s=1}^{n_2} f_s y_s \tag{2a}$$

$$\text{S.t.} \sum_{k \in K} \sum_{p \in P^k} \left(\sum_{j=1}^{n_1} \alpha_{ij} \sum_{a \in M(x_j)} h_a^p \right) \lambda_p + \sum_{s=1}^{n_2} \beta_{is} y_s \geq d_i, \quad i = 1, \ldots, m, \tag{2b}$$

$$L^k \leq \sum_{p \in P^k} \lambda_p \leq U^k, \qquad k \in K, \tag{2c}$$

$$\lambda_p \geq 0, \qquad p \in P. \tag{2d}$$

Master LP (2) is solved by column generation. Let π_i, $1 \leq i \leq m$, denote the dual variables of Constraints (2b), ν_+^k and ν_-^k, $k \in K$, are the dual variables of Constraints (2c). The reduced cost of an arc $a \in A$ is defined as:

$$\bar{c}_a = \sum_{j \in M^{-1}(a)} c_j - \sum_{i=1}^{m} \sum_{j \in M^{-1}(a)} \alpha_{ij} \pi_i.$$

The reduced cost of a path $p = (v_0, a_1, v_1, \ldots, a_{n-1}, v_{n-1}, a_n, v_n) \in P^k$ is:

$$\bar{c}(p) = \sum_{j=1}^{n} \bar{c}_{a_j} - \nu_+^k - \nu_-^k.$$

So, the pricing subproblems correspond to finding, for each $k \in K$, a path $p \in P^k$ with minimum reduced cost.

3 Generalizing State-of-the-Art Elements: Packing Sets

Formulation (1) can be used to model most VRP variants (and also many other non-VRPs). It can be solved by a standard BP algorithm (or a standard robust BCP algorithm [52], if (1b) contains separated constraints), where the RCSP subproblems are handled by a labeling dynamic programming algorithm. However, its performance on the more classic VRP variants would be very poor when compared to the best existing specific algorithms. One of the main contributions of this work is a generalization of the key additional elements found in those state-of-the-art algorithms, leading to the construction of a powerful and still quite generic BCP algorithm.

In order to do that, we introduce a new concept. Let $\mathcal{B} \subset 2^A$ be a collection of mutually disjoint subsets of A such that the constraints:

$$\sum_{a \in B} \sum_{p \in P} h_a^p \lambda_p \leq 1, \quad B \in \mathcal{B}, \tag{3}$$

are satisfied by at least one optimal solution (x^*, y^*, λ^*) of Formulation (1). In those conditions, we say that $\mathcal{B}$ defines a collection of *packing sets*. Note that a packing set can contain arcs from different graphs and not all arcs in A need to belong to some packing set. The definition of a proper $\mathcal{B}$ is part of the modeling. It does not follow automatically from the analysis of (1).

3.1 *ng*-paths

One of the weaknesses of linear relaxation (2) when modeling classical VRPs is the existence of non-elementary paths in P. In those cases, one would like to eliminate those paths. However, this would make the pricing subproblems much harder, intractable in many cases. A good compromise between formulation strength and pricing difficulty can be obtained by the so-called *ng*-paths, introduced in Baldacci et al. [7].

In our more general context, ideally, we would like to keep only routes that do not use more than one arc in the same packing set. Instead, we settle for generalized *ng*-paths defined as follows. For each arc $a \in A$, let $NG(a) \subseteq \mathcal{B}$ denote the *ng*-set of a. An *ng*-path may use two arcs belonging to the same packing set B, but only if the subpath between those two arcs passes by an arc a such that $B \notin NG(a)$. The *ng*-sets may be determined a priori or dynamically, like in [54] and [17].

3.2 Limited Memory Rank-1 Cuts

The Rank-1 Cuts (R1Cs) [48] are a generalization of the Subset Row Cuts proposed by Jepsen et al. [38]. Here, they are further generalized as follows. Consider a collection of packing sets $\mathcal{B}$ and non-negative multipliers ρ_B for each $B \in \mathcal{B}$. A Chvátal-Gomory rounding of Constraints (3) yields:

$$\sum_{p \in P} \left\lfloor \sum_{B \in \mathcal{B}} \rho_B \sum_{a \in B} h_a^p \right\rfloor \lambda_p \leq \left\lfloor \sum_{B \in \mathcal{B}} \rho_B \right\rfloor. \tag{4}$$

R1Cs are potentially strong, but each added cut makes the pricing subproblems significantly harder. The limited memory technique [45] is essential for mitigating that negative impact. In our context, a R1C characterized by its multipliers ρ is associated to a memory set $A(\rho) \subseteq A$. Variables λ_p corresponding to paths p passing by arcs $a \notin A(\rho)$ may have their coefficients decreased in (4). However, if the memory sets are adjusted in such a way that variables λ_p with positive values in the current linear relaxation have their best possible coefficients, the resulting limited memory R1C (lm-R1C) is as effective as the original R1C.

3.3 Path Enumeration

The path enumeration technique was proposed by Baldacci et. al. [5], and later improved by Contardo and Martinelli [23]. It consists in trying to enumerate into a table all paths in a certain set P^k that can possibly be part of an improving solution. After a successful enumeration, the corresponding pricing subproblem k can be solved by inspection, saving time. If the enumeration has already succeeded for all $k \in K$ and the total number of paths in the tables is not too large (say, less than 10,000) the overall problem may be even finished by a MIP solver.

In our context, we try to enumerate paths p without more than one arc in the same packing set, and with $\bar{c}(p) < UB - LB$, where UB is the best known integer solution cost, and LB the value of the current linear relaxation. Moreover, if two paths p and p' lead to variables λ_p and λ'_p with identical coefficients in (2b)–(2c), we drop the one with a larger cost.

3.4 Branching

Branching constraints over x and y variables do not change the structure of the pricing subproblems. In many models they suffice for correctness. However, there are models where Constraints (1e) need to be explicitly enforced. Branching over individual λ variables is permitted, but should be avoided due to a big negative impact in the pricing and also due to highly unbalanced branch trees [64].

The model has the option of branching using a generalization of the Ryan and Foster rule [57]. Choose distinct sets B and B' in $\mathcal{B}$. Let $P(B, B') \subseteq P$ be the subset of the paths that contain arcs in both B and B'. The branch is over the value of $\sum_{p \in P(B,B')} \lambda_p$. The pricing still becomes harder, but branch trees are more balanced.

We included in the model a new way of branching that does not increase the pricing difficulty. For chosen $B \in \mathcal{B}$, $r \in R^M$ and for a certain threshold value t^*: in the left child make $u_{a,r} = t^*$, for all $a \in B$; in the right child make $l_{a,r} = t^*$. In other words, the branch is over the accumulated consumption of resource r on arcs in B. In principle, this branching it is not complete: some fractional λ solutions can not be eliminated by it. However, it may work very well in practice.

4 Model Examples

We selected 4 problems to exemplify how problems are modeled in our solver. First, a simple didactic model; then a case where branching over the λ variables is necessary; the third model illustrates the use of secondary resources; the fourth model relies on a non-trivial transformation of the original problem.

4.1 Generalized Assignment Problem (GAP)

Data: Set T of tasks; set K of machines; capacity Q^k, $k \in K$; assignment cost c_t^k and machine load w_t^k, $t \in T$, $k \in K$.
Goal: Find an assignment of tasks to machines such that the total load in each machine does not exceed its capacity, with minimum total cost.

Model: Graph $G^k = (V^k, A^k)$ for each $k \in K$, $V^k = \{v_t^k : t = 0, \ldots, |T|\}$, $A^k = \{a_{t+}^k = (v_{t-1}^k, v_t^k), a_{t-}^k = (v_{t-1}^k, v_t^k) : t = 1, \ldots, |T|\}$, $v_{\text{source}}^k = v_0^k$, $v_{\text{sink}}^k = v_{|T|}^k$ (see Fig. 4). $R = R^M = \{1\}$; $q_{a_{t+}^k,1} = w_t^k, q_{a_{t-}^k,1} = 0, t \in T$; $[l_{v_t^k,1}, u_{v_t^k,1}] = [0, Q^k]$, $t \in T \cup \{0\}$. Binary variables x_t^k, $t \in T$, $k \in K$. The formulation is:

$$\text{Min} \sum_{t \in T} \sum_{k \in K} c_t^k x_t^k \tag{5a}$$

$$\text{S.t.} \sum_{k \in K} x_t^k = 1, \quad t \in T; \tag{5b}$$

$L^k = 0$, $U^k = 1$, $k \in K$; $M(x_t^k) = \{a_{t+}^k\}, t \in T, k \in K$ (by abuse of notation we use the variable itself instead of its linear index as the argument of mapping M). $\mathcal{B} = \cup_{t \in T}\{\{a_{t+}^k : k \in K\}\}$. Branching is over the x variables.

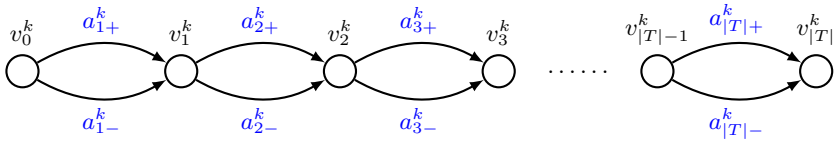

Fig. 1. GAP model graph, RCSPs correspond to binary knapsack solutions.

4.2 Vector Packing (VPP) / Bin Packing (BPP)

Data: Set T of items; set D of dimensions; bin capacities Q^d, $d \in D$; item weight w_t^d, $t \in T$, $d \in D$. (Bin packing is the case where $|D| = 1$).
Goal: Find a packing using the minimum number of bins, such that, for each dimension, the total weight of the items in a bin does not exceed its capacity.
Model: A single graph $G = (V, A)$ (we omit the index k in such cases), $V = \{v_t : t = 0, \ldots, |T|\}$, $A = \{a_{t+} = (v_{t-1}, v_t), a_{t-} = (v_{t-1}, v_t) : t = 1, \ldots, |T|\}$, $v_{\text{source}} = v_0$, $v_{\text{sink}} = v_{|T|}$. $R = R^M = D$; $q_{a_{t+},d} = w_t^d, q_{a_{t-},d} = 0, t \in T, d \in D$; $[l_{v_t,d}, u_{v_t,d}] = [0, Q^d]$, $t \in T \cup \{0\}, d \in D$. Binary variables x_t, $t \in T \cup \{0\}$. The formulation is:

$$\text{Min} \quad x_0 \tag{6a}$$

$$\text{S.t. } x_t = 1, \quad t \in T; \tag{6b}$$

$L = 0$, $U = \infty$; $M(x_0) = \{a_{1+}, a_{1-}\}$, $M(x_t) = \{a_{t+}\}, t \in T$. $\mathcal{B} = \cup_{t \in T}\{\{a_{t+}\}\}$. Branching on λ variables; first over accumulated resource consumption and, if still needed, by Ryan and Foster rule.

4.3 Pickup and Delivery VRPTW (PDPTW)

Data: Directed graph $G' = (V', A')$, where $V' = \{0\} \cup P' \cup D'$, $P' = \{1, \ldots, n\}$ is the set of pickup vertices and $D' = \{n+1, \ldots, 2n\}$ the set of corresponding deliveries (a pickup at i correspond to a delivery at $i+n$); vehicle capacities Q;

traveling cost c_a and time (traveling time plus service time) t_a , $a \in A'$; positive demands d_v, $v \in P'$ ($d_v = -d_{v-n}$, $v \in D'$); and time windows $[l'_v, u'_v]$, $v \in V'$.
Goal: Find a set of routes such that each pickup or delivery vertex is visited exactly once, any visit to a pickup vertex implies that the corresponding delivery vertex is visited later by the same route, the accumulated demand of visited nodes never exceed the capacity along a route, and the accumulated sum of traversal and waiting times until reaching each node falls within its time window (waiting times are added to meet lower bounds), minimizing the total sum of traversal costs.
Model: A single graph $G = (V, A)$, $V = V' \cup \{2n+1\}$, $A = (A' \setminus \{(v,0) : v \in D'\}) \cup \{(v, 2n+1) : v \in D'\}$ (assume $c_{(v,2n+1)} = c_{(v,0)}$ and $t_{(v,2n+1)} = t_{(v,0)}$), $v_{\text{source}} = v_0$, $v_{\text{sink}} = v_{2n+1}$. $R^M = \{n+2\}$; $R^N = \{1, \ldots, n+1\}$; $q_{(v,v'),v'} = 1$, if $v' \in P'$, $q_{(v,v'),v'-n} = -1$, if $v' \in D'$, and $q_{(v,v'),n+1} = d_{v'}$, $(v, v') \in A$; $q_{a,n+2} = t_a$, $a \in A$; all other resource consumptions are zero; $u_{v,r} = 1$, $r = 1, \ldots, n$, $u_{v,n+1} = u_{2n+1,n+1} = Q$ and $(l_{v,n+2}, u_{v,n+2}) = (l'_v, u'_v)$, $v \in P' \cup D'$; all other resource bounds are zero. Binary variables x_a, $a \in A$. The formulation is:

$$\text{Min} \quad \textstyle\sum_{a \in A} c_a x_a \tag{7a}$$
$$\text{S.t.} \textstyle\sum_{a \in \delta^-(v)} x_a = 1, \quad v \in P'; \tag{7b}$$

$L = 0$, $U = \infty$; $M(x_a) = \{a\}$, $a \in A$. $\mathcal{B} = \cup_{v \in V} \{\delta^-(v)\}$. Branching on x variables.

4.4 Capacitated Arc Routing (CARP)

Data: Undirected graph $G' = (V', E)$, $V' = \{0, \ldots, n\}$, 0 is the depot vertex; positive cost c_e and non-negative demand d_e, $e \in E$, set of required edges $S = \{e \in E \,|\, d_e > 0\}$; vehicle capacity Q.
Goal: Find a minimum cost set of routes, closed walks starting and ending at the depot, that serve the demands in all required edges. Edges in a route can be traversed either serving or deadheading (not servicing). The sum of the demands of the served edges in a route can not exceed capacity.
Model: For $i, j \in V'$, let $D(i,j) \subseteq E$ be the set of edges in a chosen cheapest path from i to j, with cost $C(i,j) = \sum_{e \in D(i,j)} c_e$. Define a dummy required edge $r_0 = (0, 0')$ and $S_0 = S \cup \{r_0\}$. For each $r = (w_1, w_2) \in S_0$, define $o(r, w_1) = w_2$ and $o(r, w_2) = w_1$.

The model has a single graph $G = (V, A)$, $V = \{v_r^w : r \in S_0, w \in r\}$, $A = \{(v_{r_1}^{w_1}, v_{r_2}^{z_1}), (v_{r_1}^{w_1}, v_{r_2}^{z_2}), (v_{r_1}^{w_2}, v_{r_2}^{z_1}), (v_{r_1}^{w_2}, v_{r_2}^{z_2}) : r_1 = (w_1, w_2), r_2 = (z_1, z_2) \in S_0\}$, $v_{\text{source}} = v_{r_0}^0$, $v_{\text{sink}} = v_{r_0}^{0'}$; $R = R^M = \{1\}$; for $a = (v_{r_1}^w, v_{r_2}^z) \in A$, $q_{a,1} = d_{r_2}$; $l_{v,1} = 0, u_{v,1} = Q, v \in V$. Binary variables x_a, $a \in A$. For $a = (v_{r_1}^w, v_{r_2}^z) \in A$, $c_a = C(w, o(r_2, z)) + c_{r_2}$. The formulation is:

$$\text{Min} \quad \textstyle\sum_{a \in A} c_a x_a \tag{8a}$$
$$\text{S.t.} \textstyle\sum_{a \in \delta^-(\{v_r^{w_1}, v_r^{w_2}\})} x_a = 1, \quad r = (w_1, w_2) \in S, \tag{8b}$$

plus Rounded Capacity Cuts [43] and Lifted Odd-Cutsets [11, 9]; $L = 0$, $U = \infty$; $M(x_a) = \{a\}$, $a \in A$. $\mathcal{B} = \cup_{r=(w_1,w_2) \in S} \{\delta^-(\{v_r^{w_1}, v_r^{w_2}\})\}$. Branching on aggregation of x variables (see Appendix A).

5 Computational Experiments

The generic BCP solver optimization algorithms were coded in C++ over the BaPCod package [63]. IBM CPLEX Optimizer version 12.8.0 was used as the LP solver in column generation and as the solver for the enumerated MIPs. The experiments were run on a 2 Deca-core Ivy-Bridge Haswell Intel Xeon E5-2680 v3 server running at 2.50 GHz. The 128 GB of available RAM was shared between 8 copies of the algorithm running in parallel on the server. Each instance is solved by one copy of the algorithm using a single thread. The models are defined using either a C++ interface or a Julia–JuMP [29] based interface.

A description of the main algorithms used in the BCP solver can not be presented here by lack of space. However, they are generalizations and enhancements of already published algorithms. In some cases the original algorithms had to be significantly revised, to avoid that the generalizations introduce excessive performance overheads. Pricing problems are solved by a bucket graph based labeling algorithm [58], including a bucket arc elimination procedure based on reduced costs. Automatic dual price smoothing [51] is employed to stabilize the column generation convergence. Path enumeration is performed using an extension of the algorithm from [5, 47]. Multi-phase strong branching [55, 47] is used to reduce the search tree size. Restricted master and diving heuristics [59] are built-in to improve the primal solution during the search.

In Table 1, we show computational results for 13 problems. The first column is the problem acronym, second column refers to data sets, the third indicates the number of instances. Next is the time limit per instance. The last three columns show the results obtained by our generic solver, as well as by two other algorithms, those with the best (to our knowledge) published results for the data set. For each algorithm, we give the number of instances solved within the time limit, the average time in brackets (geometric mean time if the time limit is 10 hours or more), and its reference. For instances not solved, the time limit is considered as the solution time. Best results are marked in bold. Note that the generic solver uses a single parameterization per problem, not per data set. Additional information about experiments is available in Appendix A.

The results presented in Table 1 show that the generic BCP significantly outperforms the state-of-the-art for VRPTW, TOP, CTOP, CPTP, VRPSL, and VPP. A noticeably better performance is achieved for CVRP and HFVRP. For MDVRP, GAP, BPP and CARP, the generic BCP is comparable to the best performing algorithms in the literature. Results are mixed for PDPTW. Worse performance for LiLim instances can be explained by the fact that the generic BCP does not incorporate some labeling algorithm acceleration techniques specific to PDPTW. For the RopkeCordeau instances however, generic state-of-the-art BCP elements mitigate the effect of lacking ad-hoc enhancements.

6 Conclusions

We proposed a new generic way of modeling VRPs and related problems, so that they can be solved by an algorithm that already includes many state-of-the-art

Problem	Data set	#	T.L.	Gen. BCP	Best Publ.	2nd Best
CVRP	E-M [20, 21]	12	10h	12 (61s)	**12 (49s)** [47]	10 (432s) [23]
	X [62]	58	60h	**36 (147m)**	34 (209m) [62]	—
VRPTW	Solomon Hardest [61]	14	1h	**14 (5m)**	13 (17m) [46]	9 (39m) [7]
	Homberger 200 [34]	60	30h	**56 (21m)**	50 (70m) [46]	7 (-) [39]
HFVRP	BaldacciMingozzi [6]	40	1h	**40 (144s)**	39 (287s) [50]	34 (855s) [6]
MDVRP	Cordeau [24]	11	1h	**11 (6m)**	11 (7m) [50]	9 (25m) [23]
PDPTW	RopkeCordeau [56]	40	1h	**40 (5m)**	33 (17m) [35]	32 (14m) [4]
	LiLim [41]	30	1h	3 (56m)	**23 (20m)** [4]	18 (27m) [35]
TOP	Chao class 4 [19]	60	1h	**55 (8m)**	39 (15m) [13]	30 (-) [31]
CTOP	Archetti [2]	14	1h	**13 (7m)**	6 (35m) [1]	7 (34m) [2]
CPTP	Archetti open [2]	28	1h	**24 (9m)**	0 (1h) [16]	0 (1h) [1]
VRPSL	Bulhoes [16]	180	2h	**159 (16m)**	49 (90m) [16]	—
GAP	OR-Lib, type D [10]	6	2h	5 (40m)	**5 (30m)** [53]	5 (46m) [3]
	Nauss [44]	30	1h	**25 (23m)**	1 (58m) [36]	0 (1h) [44]
VPP	Classes 1,4,5,9 [18]	40	1h	**38 (8m)**	13 (50m) [37]	10 (53m) [15]
BPP	Falkenauer T [32]	80	10m	80 (16s)	**80 (1s)** [15]	80 (24s) [12]
	Hard28 [60]	28	10m	28 (17s)	**28 (7s)** [12]	26 (14s) [15]
	AI [26]	250	1h	**160 (25m)**	116 (35m) [12]	100 (40m) [15]
	ANI [26]	250	1h	103 (35m)	**164 (35m)** [22]	51 (48m) [12]
CARP	Eglese [30]	24	30h	**22 (36m)**	22 (43m) [49]	10 (237m) [9]

Table 1. Generic solver vs best specific solvers on 13 problems.

elements. It combines old modeling concepts (like the use of RCSPs for defining the valid routes) with a new one, the packing sets. The experiments show that the generic solver has a performance either comparable or better than the specific algorithms for all VRP variants tested. The cases where the performance was much better can be explained by the fact that previous authors often did not use some advanced BCP elements due to the complexity of their implementation. However, if generic BCP solvers become publicly and/or commercially available, we believe that their use may become as standard as that of MIP solvers nowadays.

We plan to release the presented generic solver for academic use after additional testing and documentation. It will include the optimization algorithms in a pre-compiled library and a Julia–JuMP user interface. Modeling a typical VRP variant, like those in our tests, requires around 100 lines of Julia code. This means that a user can already have a good working algorithm in a day. More work on computational experiments for parameter tuning may be needed for an improved performance. Then separation routines for problem specific cuts can be added for top performance.

References

1. Archetti, C., Bianchessi, N., Speranza, M.: Optimal solutions for routing problems with profits. Discrete Applied Mathematics **161**(4–5), 547–557 (2013)
2. Archetti, C., Feillet, D., Hertz, A., Speranza, M.G.: The capacitated team orienteering and profitable tour problems. Journal of the Operational Research Society **60**(6), 831–842 (Jun 2009)
3. Avella, P., Boccia, M., Vasilyev, I.: A computational study of exact knapsack separation for the generalized assignment problem. Computational Optimization and Applications **45**(3), 543–555 (2010)
4. Baldacci, R., Bartolini, E., Mingozzi, A.: An exact algorithm for the pickup and delivery problem with time windows. Operations Research **59**(2), 414–426 (2011)
5. Baldacci, R., Christofides, N., Mingozzi, A.: An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. Mathematical Programming **115**, 351–385 (2008)
6. Baldacci, R., Mingozzi, A.: A unified exact method for solving different classes of vehicle routing problems. Mathematical Programming **120**(2), 347–380 (2009)
7. Baldacci, R., Mingozzi, A., Roberti, R.: New route relaxation and pricing strategies for the vehicle routing problem. Operations Research **59**(5), 1269–1283 (2011)
8. Balinski, M., Quandt, R.: On an integer program for a delivery problem. Operations Research **12**(2), 300–304 (1964)
9. Bartolini, E., Cordeau, J.F., Laporte, G.: Improved lower bounds and exact algorithm for the capacitated arc routing problem. Mathematical Programming **137**(1), 409–452 (Feb 2013)
10. Beasley, J.E.: OR-Library: Distributing test problems by electronic mail. The Journal of the Operational Research Society **41**(11), 1069–1072 (1990)
11. Belenguer, J., Benavent, E.: The capacitated arc routing problem: Valid inequalities and facets. Computational Optimization & Applications **10**(2), 165–187 (1998)
12. Belov, G., Scheithauer, G.: A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. European Journal of Operational Research **171**(1), 85 – 106 (2006)
13. Bianchessi, N., Mansini, R., Speranza, M.G.: A branchandcut algorithm for the team orienteering problem. International Transactions in Operational Research **25**(2), 627–635 (2018)
14. Bode, C., Irnich, S.: Cut-first branch-and-price-second for the capacitated arc-routing problem. Operations Research **60**(5), 1167–1182 (2012)
15. Brandão, F., Pedroso, J.a.P.: Bin packing and related problems: General arc-flow formulation with graph compression. Computers & Operations Research **69**, 56 – 67 (2016)
16. Bulhoes, T., Hà, M.H., Martinelli, R., Vidal, T.: The vehicle routing problem with service level constraints. European Journal of Operational Research **265**(2), 544 – 558 (2018)
17. Bulhoes, T., Sadykov, R., Uchoa, E.: A branch-and-price algorithm for the minimum latency problem. Computers & Operations Research **93**, 66–78 (May 2018)
18. Caprara, A., Toth, P.: Lower bounds and algorithms for the 2-dimensional vector packing problem. Discrete Applied Mathematics **111**(3), 231 – 262 (2001)
19. Chao, I.M., Golden, B.L., Wasil, E.A.: The team orienteering problem. European Journal of Operational Research **88**(3), 464 – 474 (1996)
20. Christofides, N., Eilon, S.: An algorithm for the vehicle-dispatching problem. Operational Research Quarterly **20**, 309–318 (1969)

21. Christofides, N., Mingozzi, A., Toth, P.: Combinatorial Optimization, chap. The vehicle routing problem, pp. 315–338. Wiley, Chichester (1979)
22. Clautiaux, F., Hanafi, S., Macedo, R., Émilie Voge, M., Alves, C.: Iterative aggregation and disaggregation algorithm for pseudo-polynomial network flow models with side constraints. European Journal of Operational Research **258**(2), 467 – 477 (2017)
23. Contardo, C., Martinelli, R.: A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. Discrete Optimization **12**, 129 – 146 (2014)
24. Cordeau, J.F., Gendreau, M., Laporte, G.: A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks **30**(2), 105–119 (1997)
25. Dantzig, G., Ramser, J.: The truck dispatching problem. Management science **6**(1), 80–91 (1959)
26. Delorme, M., Iori, M., Martello, S.: Bin packing and cutting stock problems: Mathematical models and exact algorithms. European Journal of Operational Research **255**(1), 1–20 (2016)
27. Desaulniers, G., Desrosiers, J., Solomon, M.M., Soumis, F., Villeneuve, D., et al.: A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In: Fleet management and logistics, pp. 57–93. Springer (1998)
28. Desaulniers, G., Lessard, F., Hadjar, A.: Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. Transportation Science **42**(3), 387–404 (2008)
29. Dunning, I., Huchette, J., Lubin, M.: JuMP: A modeling language for mathematical optimization. SIAM Review **59**(2), 295–320 (2017)
30. Eglese, R.W., Li, L.Y.O.: Efficient routeing for winter gritting. Journal of the Operational Research Society **43**(11), 1031–1034 (1992)
31. El-Hajj, R., Dang, D.C., Moukrim, A.: Solving the team orienteering problem with cutting planes. Computers & Operations Research **74**, 21 – 30 (2016)
32. Falkenauer, E.: A hybrid grouping genetic algorithm for bin packing. Journal of Heuristics **2**, 5–30 (1996)
33. Fukasawa, R., Longo, H., Lysgaard, J., Aragão, M.P.d., Reis, M., Uchoa, E., Werneck, R.F.: Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Mathematical Programming **106**(3), 491–511 (2006)
34. Gehring, H., Homberger, J.: Parallelization of a two-phase metaheuristic for routing problems with time windows. Journal of Heuristics **8**(3), 251–276 (2002)
35. Gschwind, T., Irnich, S., Rothenbächer, A.K., Tilk, C.: Bidirectional labeling in column-generation algorithms for pickup-and-delivery problems. European Journal of Operational Research **266**(2), 521 – 530 (2018)
36. Gurobi Optimization, L.: Gurobi optimizer reference manual, version 7.5 (2017), http://www.gurobi.com
37. Heßler, K., Gschwind, T., Irnich, S.: Stabilized branch-and-price algorithms for vector packing problems. European Journal of Operational Research **271**(2), 401 – 419 (2018)
38. Jepsen, M., Petersen, B., Spoorendonk, S., Pisinger, D.: Subset-row inequalities applied to the vehicle-routing problem with time windows. Operations Research **56**(2), 497–511 (2008)
39. Kallehauge, B., Larsen, J., Madsen, O.: Lagrangian duality applied to the vehicle routing problem with time windows **33**(5), 1464–1487 (2006)
40. Laporte, G., Nobert, Y.: A branch and bound algorithm for the capacitated vehicle routing problem. Operations-Research-Spektrum **5**(2), 77–85 (Jun 1983)

41. Li, H., Lim, A.: A metaheuristic for the pickup and delivery problem with time windows. International Journal on Artificial Intelligence Tools **12**(02), 173–186 (2003)
42. Lysgaard, J.: CVRPSEP: A package of separation routines for the capacitated vehicle routing problem. Aarhus School of Business, Department of Management Science and Logistics (2003)
43. Lysgaard, J., Letchford, A.N., Eglese, R.W.: A new branch-and-cut algorithm for the capacitated vehicle routing problem. Mathematical Programming **100**(2), 423–445 (Jun 2004)
44. Nauss, R.M.: Solving the generalized assignment problem: An optimizing and heuristic approach. INFORMS Journal on Computing **15**(3), 249–266 (2003)
45. Pecin, D., Pessoa, A., Poggi, M., Uchoa, E.: Improved branch-cut-and-price for capacitated vehicle routing. In: Proceedings of the XVII IPCO. Lecture Notes in Computer Science, vol. 8494, pp. 393–403. Springer (2014)
46. Pecin, D., Contardo, C., Desaulniers, G., Uchoa, E.: New enhancements for the exact solution of the vehicle routing problem with time windows. INFORMS Journal on Computing **29**(3), 489–502 (2017)
47. Pecin, D., Pessoa, A., Poggi, M., Uchoa, E.: Improved branch-cut-and-price for capacitated vehicle routing. Mathematical Programming Computation **9**(1), 61–100 (2017)
48. Pecin, D., Pessoa, A., Poggi, M., Uchoa, E., Santos, H.: Limited memory rank-1 cuts for vehicle routing problems. Operations Research Letters **45**(3), 206 – 209 (2017)
49. Pecin, D., Uchoa, E.: Comparative analysis of capacitated arc routing formulations for designing a new branch-cut-and-price algorithm. Transportation Science (Forthcoming) (2019)
50. Pessoa, A., Sadykov, R., Uchoa, E.: Enhanced branch-cut-and-price algorithm for heterogeneous fleet vehicle routing problems. European Journal of Operational Research **270**, 530–543 (2018)
51. Pessoa, A., Sadykov, R., Uchoa, E., Vanderbeck, F.: Automation and combination of linear-programming based stabilization techniques in column generation. INFORMS Journal on Computing **30**(2), 339–360 (2018)
52. Poggi de Aragão, M., Uchoa, E.: Integer program reformulation for robust branch-and-cut-and-price. In: Wolsey, L. (ed.) Annals of Mathematical Programming in Rio. pp. 56–61. Búzios, Brazil (2003)
53. Posta, M., Ferland, J.A., Michelon, P.: An exact method with variable fixing for solving the generalized assignment problem. Computational Optimization and Applications **52**, 629–644 (2012)
54. Roberti, R., Mingozzi, A.: Dynamic ng-path relaxation for the delivery man problem. Transportation Science **48**(3), 413–424 (2014)
55. Røpke, S.: Branching decisions in branch-and-cut-and-price algorithms for vehicle routing problems. Presentation in Column Generation 2012 (2012)
56. Ropke, S., Cordeau, J.F.: Branch and cut and price for the pickup and delivery problem with time windows. Transportation Science **43**(3), 267–286 (2009)
57. Ryan, D.M., Foster, B.A.: An integer programming approach to scheduling. In: Wren, A. (ed.) Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling, pp. 269–280. North-Holland (1981)
58. Sadykov, R., Uchoa, E., Pessoa, A.: A bucket graph based labeling algorithm with application to vehicle routing. Tech. Rep. L-2017-7, Cadernos do LOGIS-UFF, Niterói, Brazil (October 2017)

59. Sadykov, R., Vanderbeck, F., Pessoa, A., Tahiri, I., Uchoa, E.: Primal heuristics for branch-and-price: the assets of diving methods. INFORMS Journal on Computing (Forthcoming) (2018)
60. Schoenfield, J.E.: Fast, exact solution of open bin packing problems without linear programming. Technical report, US Army Space and Missile Defense Command (2002)
61. Solomon, M.M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research **35**(2), 254–265 (1987)
62. Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Subramanian, A., Vidal, T.: New benchmark instances for the capacitated vehicle routing problem. European Journal of Operational Research **257**(3), 845–858 (2017)
63. Vanderbeck, F., Sadykov, R., Tahiri, I.: BaPCod — a generic Branch-And-Price Code (2018), https://realopt.bordeaux.inria.fr/?page_id=2
64. Vanderbeck, F., Wolsey, L.A.: Reformulation and decomposition of integer programs. In: 50 Years of Integer Programming 1958-2008, pp. 431–502. Springer (2010)

A Additional Information on the Experiments

We provide additional information about the experiments reported in Table 1, including relevant modeling decisions, datasets and remarks.

Note that the performance of exact algorithms is sensitive to the initial primal bound value given by the user before execution. We tried to be as fair as possible in this regard. Unless stated otherwise for the problems below we use the same bounds (usually took from the heuristic literature) as in previous works.

CVRP: (Capacitated Vehicle Routing Problem) The model is defined over undirected edge variables and separates Rounded Capacity Cuts (RCCs) [40], using the procedure in CVRPSEP [42]. A packing set is defined for each customer and contains all incoming arcs to the corresponding node in the graph. Branching is done over edge variables. The considered E-M instances are the 12 hardest ones, those considered in [47]. The considered X instances are those with less than 400 customers.

VRPTW: (Vehicle Routing Problem with Time Windows) The same model as CVRP except that only time is defined as a graph resource, capacity is enforced by RCCs. The considered Solomon instances (all with 100 customers) are the hardest ones according to [46].

HFVRP: (Heterogeneous Fleet Vehicle Routing Problem) The model is defined over undirected edge variables. Each graph G^k (with capacity resource) corresponds to a vehicle type. Branching is on the number of paths in P^k, assignment of packing sets to graphs, and on edge variables. Instances with 50, 75, and 100 customers are considered.

MDVRP: (Multi-Depot Vehicle Routing problem) The model is defined over undirected edge variables. Each graph G^k corresponds to a depot. Branching is the same as for HFVRP. Only instances with one capacity resource are considered (without time constraints).

PDPTW: The model is precisely defined in Section 4.3.

TOP/CTOP: (Team Orienteering Problem) The model contains binary variables y that are not mapped to any RCSP graph, so they appear directly in Formulation 2. Those variables indicate which customers will be visited. The problem is to maximize the total profit of visited customers. For TOP, one (time) resource is defined. In CTOP, an additional capacity resource is considered. Branching is on y and edge variables. No initial upper bound is defined. Instances of class 4, the most difficult one according to [13], are considered for TOP. Only basic instances from [2] are considered for CTOP, as well as open ones.

CPTP: (Capacitated Profitable Tour Problem:) Similar to CTOP, except that there is no time resource. The objective is the difference between the total profit and the transportation cost. Only open instances from [2] are considered.

VRPSL: (VRP with Service Level constraints) Generalization of CVRP in which a service weight is defined for each customer. For each predefined group of customers, total service weight of visited customers should not be below a threshold. The model contains edge and y variables. For each group, a knapsack constraint over y variables is defined. Branching is both on y and edge variables.

GAP: The model is precisely defined in Section 4.1. Instances of the most difficult type D are considered. For OR-Library instances, we took best known solution values as initial upper bounds. We used Nauss instances with $|T| = 90, 100$ and $|K| = 25, 30$. Initial bounds for them were calculated by us using problem specific strong diving heuristic from [58]. Its time is included in the reported time.

BPP/VPP: The model is precisely defined in Section 4.2. The branching over the accumulated resource consumption showed to be effective so that Ryan and Foster branch rule was never needed. For VPP, we took only largest instances (200 items) with 2 resources of classes 1, 4, 5, and 9, the most difficult ones according to [37]. No initial bound is given for VPP. For BPP, we used the initial primal bound equal to the rounded up column generation dual bound plus one. Such solutions are easily obtainable by very simple heuristics.

CARP: The model is defined in Section 4.4. The branching is done on aggregation of x variables: 1) corresponding to node degrees in the original graph; 2) corresponding to whether required two edges are served immediately one after another by the same route or not. The Eglese dataset is used in all recent works on CARP.

B Open CVRP Instances Solved

According to CVRPLIB (`http://vrp.atd-lab.inf.puc-rio.br`) there were 52 open CVRP instances in the X set [62]. We started long runs of the generic solver on the most promising ones, using a specially calibrated parameterization. We could solve 5 instances to optimality for the first time, as indicated in Table 2. Improved best known solutions are underlined.

Instance	Prev. BKS	Root LB	Nodes	Total Time	OPT
X-n284-k15	20226	20168	940	11.0 days	**20215**
X-n322-k28	29834	29731	1197	5.6 days	**29834**
X-n393-k38	38260	38194	1331	5.8 days	**38260**
X-n469-k138	221909	221585	8964	15.2 days	**221824**
X-n548-k50	86701	86650	337	2.0 days	**86700**

Table 2. Detailed results on the open instances solved.