Production, Manufacturing and Logistics

The pyramidal capacitated vehicle routing problem

Jens Lysgaard*

CORAL, Department of Business Studies, Aarhus School of Business, University of Aarhus, Fuglesangs Allé 4, DK-8210 Aarhus V, Denmark

ARTICLE INFO

Article history:
Received 29 April 2008
Accepted 21 November 2009
Available online 3 December 2009

Keywords:
Routing
Pyramidal traveling salesman
Branch-and-cut-and-price

ABSTRACT

This paper introduces the pyramidal capacitated vehicle routing problem (PCVRP) as a restricted version of the capacitated vehicle routing problem (CVRP). In the PCVRP each route is required to be pyramidal in a sense generalized from the pyramidal traveling salesman problem (PTSP). A pyramidal route is defined as a route on which the vehicle first visits customers in increasing order of customer index, and on the remaining part of the route visits customers in decreasing order of customer index. Moreover, this paper develops an exact branch-and-cut-and-price (BCP) algorithm for the PCVRP. A main feature of the algorithm is that exact pricing over elementary routes are done in pseudo-polynomial time.

Computational results suggest that PCVRP solutions are highly useful for obtaining near-optimal solutions to the CVRP. Furthermore, pricing of pyramidal routes may prove to be very useful in column generation for the CVRP.

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

This paper introduces the pyramidal capacitated vehicle routing problem (PCVRP) as a restricted version of the capacitated vehicle routing problem (CVRP). In the PCVRP each route is required to be pyramidal in a sense generalized from the pyramidal traveling salesman problem (PTSP). For a formal definition of the PCVRP, it is useful first to consider the CVRP and the PTSP.

The CVRP can be defined as follows (Fukasawa et al., 2006; Gold
den et al., 2007; Lysgaard et al., 2004; Toth and Vigo, 2002). Let $G = (V, E)$ be a complete undirected graph, with $V = \{0, \ldots, n\}$. Vertex 0 represents a depot, whereas each of the vertices in $V_c = \{1, \ldots, n\}$ represents a customer. The symmetric cost of travel between vertices $i$ and $j$ is denoted by $c_{ij}$. A number $K$ of identical vehicles, each of capacity $Q > 0$, is available. Each customer $i$ has an integer demand $q_i$, with $0 < q_i \leq Q$. Each customer must be served by a single vehicle and no vehicle can serve a set of customers whose demand exceeds its capacity. The task is to find a set of $K' \leq K$ vehicle routes of minimum cost, where each vehicle used leaves from and returns to the depot. The number of routes may or may not be predetermined. In particular, $K$ may be set equal to $n$ if the number of routes is unrestricted, but in many cases, $K$ is set equal to the minimum possible, as in Lysgaard et al. (2004).

The material in this paper does not require a particular value of $K$, but in our computational experiments we shall fix $K$ to the minimum possible for the purpose of comparison with known optimal solutions to certain CVRP instances.

The PTSP can be defined as follows (Gilmore et al., 1985, 2002). Let $G'$ be a directed graph with vertex set $\{1, \ldots, n\}$ and a set of arcs with given lengths. A pyramidal traveling salesman route in $G'$ is a hamiltonian circuit $(1, i_1, \ldots, i_k, 1, j_1, \ldots, j_p, 1)$ such that $i_1 < \cdots < i_k$ and $j_1 < \cdots < j_p$. The PTSP is to determine a shortest pyramidal traveling salesman route in $G'$.

In relation to the PTSP, the concepts of valleys and peaks are useful. In a given hamiltonian circuit containing the subsequence $(\ldots, i, j, k, \ldots)$, vertex $j$ is called a valley if and only if $i > j$ and $k > j$, and $j$ is called a peak if and only if $i < j$ and $k < j$. In a feasible solution to the PTSP, vertex 1 is the only valley, and vertex $n$ is the only peak. The requirement that a traveling salesman route in $G'$ must be pyramidal is equivalent to requiring that vertex 1 (vertex $n$) must be the only valley (peak).

We now consider the definition of the PCVRP. First, for the CVRP we define a pyramidal route as given by Definition 1.

Definition 1. In the CVRP, a pyramidal route is a sequence of vertices $(0, i_1, \ldots, i_k, j_1, \ldots, j_p, 1)$ which contains $\alpha + \beta$ distinct customers, such that $i_1 < \cdots < i_k$ and $j_1 < \cdots < j_p$.

We will use the terms valley and peak in relation to the CVRP. Their definitions are equivalent to those for the TSP. As such, in a feasible solution to the CVRP, the depot is a valley on each route, and customer $n$ is a peak on one of the routes. Moreover, no customer is a valley, and each route contains exactly one peak. By definition, a pyramidal route is elementary.

Given Definition 1, we can now define the PCVRP.

Definition 2. The PCVRP is the CVRP with the additional restriction that each route must be pyramidal.
Regarding the complexity of the PCVRP, it is useful to consider its relation to other combinatorial problems. Just as the CVRP may be viewed as conceptually lying in the intersection of the bin packing problem (BPP) and the TSP (Ralphs et al., 2003), the PCVRP may be viewed as conceptually lying in the intersection of the BPP and the PTSP.

The PCVRP with $c_{ij} = 0$ for $i, j = 1, \ldots, n$, and $c_{ij} = \frac{1}{2}$ for $j = 1, \ldots, n$, is equivalent to the BPP with item sizes given by the customer demands, and bin capacity $Q$. As such, the PCVRP generalizes the BPP, which is strongly NP-hard (Garey and Johnson, 1979; Martello and Toth, 1990). The PCVRP is therefore strongly NP-hard.

However, the routing aspect is much easier for the PCVRP than for the CVRP. For a given set of customers to be visited on a route, an optimal pyramidal route can be computed in polynomial time. In the general case, the PTSP on $n$ vertices can be solved to optimality in $O(n^2)$ time (Gilmore et al., 1985).

Based on the relative ease of solving PTSPs, this paper develops specialized column generation algorithms for a set partitioning problem (SPP) formulation of the PCVRP, in which variables (columns) represent pyramidal routes. These algorithms are then embedded in branch-and-price (BP) and branch-and-cut-and-price (BCP) algorithms.

Provided that customers are indexed in nondecreasing order of distance from the depot, the shape of a pyramidal route is indeed found in many optimal solutions to CVRP instances. Moreover, when neglecting the order of visit and only focusing on the assignment of customers to routes, a PCVRP solution may be viewed simply as representing a clustering of customers, from which a CVRP solution may be produced by solving a TSP for each cluster. Computational results suggest that PCVRP solutions are highly useful for obtaining near-optimal solutions to the CVRP.

As an illustration, Fig. 1 shows the optimal solution to a PCVRP instance which is obtained from the CVRP instance $E-51-5$, but in which the customers have been indexed in nondecreasing order of distance from the depot. The solid square represents the depot, circles represent customers, and the numbers are the customer indices. Edges between the depot and customers are omitted. (We note that different indexing rules may produce different PCVRP instances from the same CVRP instance.)

In comparison, Fig. 2 shows the optimal solution to the CVRP instance $E-51-5$ itself. In Fig. 2, the edges in bold are those that are not present in Fig. 1. It is remarkable that the two solutions have two routes in common, and that the assignment of customers to routes is the same in the two solutions. As such, in this example, the optimal CVRP solution can be obtained simply by resequencing the customers on each route in the PCVRP solution (i.e., by solving a TSP for each route in the PCVRP solution).

The paper is organized as follows. Section 2 introduces the Capacitated Prize-Collecting PTSP which arises as the column generation subproblem for the SPP formulation, along with an exact algorithm as well as a heuristic for solving this problem. Section 3 presents a specialized assignment heuristic for solving the PCVRP. Section 4 presents our BP and BCP algorithms. Computational results are presented in Section 5, and conclusions are made in Section 6.

2. The capacitated prize-collecting PTSP

The Capacitated Prize-Collecting PTSP (CPPTSP) can be defined as a restricted version of the Capacitated Prize-Collecting TSP (Bixby, 1999; Fischetti et al., 2002), the restriction being that the route must be pyramidal.

More formally, let $G = (V', E')$ be an undirected graph with vertices $V' = \{0, 1, \ldots, n\}$ and edge set $E'$. Each edge $\{i,j\}$ has a given length $d_{ij}$. Each vertex $i$ is associated with a nonnegative weight $w_i$ and a prize $p_i$, where $w_0 = p_0 = 0$. Finally, a weight limit $W$ is given. We emphasize that by Definition 1, any pyramidal route in $G$ which visits at least two customers is a simple cycle containing vertex 0.

The cost $C(E)$ of any simple cycle $E \subseteq E'$ containing vertices $V(E)$ is given by the total length of the edges on the cycle, minus the sum of the collected prizes:

$$C(E) = \sum_{\{i,j\} \in E} d_{ij} - \sum_{i \in V(E)} p_i$$
\[ C(E) = \sum_{(i,j) \in E} d_{ij} - \sum_{k \in V(F)} p_k. \]  
\hspace{1cm} (1)

Moreover, let the total weight of a cycle be the sum of the weights of the vertices on the cycle.

For completeness, since a route visiting only one customer is not a simple cycle, we need to define the cost and weight of a single-customer route, respectively: a route of the form \( 0 - i - 0 \) has cost \( 2d_{0i} - p_i \) and weight \( w \). We can now define the CPCPTSP.

**Definition 3.** The CPCPTSP is the problem of finding a minimum-cost pyramidal route in \( G \), subject to the restriction that its total weight must be less than or equal to \( W \).

The importance of the CPCPTSP is due to its role as the column generation subproblem in the set partitioning problem formulation of the PCVRP. For the purpose of constructing pyramidal routes, a key concept is that of a pyramidal path.

**Definition 4.** A pyramidal path is a sequence of vertices \( (i_1, \ldots, i_x, 0, j_1, \ldots, j_y) \) which contains \( x + y \) distinct customers, such that \( i_1 < \cdots < i_x \) and \( j_1 < \cdots < j_y \).

It follows from Definitions 1 and 4 that any pyramidal route can be constructed by adding the edge between the two end vertices of a pyramidal path.

Moreover, any pyramidal path can be constructed by including customers in increasing order of customer index. This leads to an exact dynamic programming algorithm which is now described.

Let \( L(i, j, w) \) denote the length of the shortest among all pyramidal paths with vertices \( i \) and \( j \) as the two end vertices, and with a total weight of \( w \). The value of \( L(i, j, w) \), for all combinations of \( i, j \) (with \( i < j \)) and \( 0 < w \leq W \), can be found by dynamic programming using the following recurrence relation:

\[
L(i, j, w) = \min \left\{ \min_{k < j} \left\{ L(i, k, w - w_j) + d_{ij} - p_i \right\}, \min_{k > i} \left\{ L(k, i, w - w_j) + d_{ij} - p_j \right\} \right\}. \hspace{1cm} (2)
\]

The recurrence in (2) considers all possible ways of linking customer \( j \) to each end vertex of a pyramidal path whose end vertices are both smaller-indexed than \( j \). The recurrence is initialized by setting \( L(0, 0, 0) = 0 \). The shortest among all pyramidal routes is given by \( i^*, j^* \) and \( w^* \) according to (3).

\[
L(i^*, j^*, w^*) + d_{i^*j^*} = \min_{i,j,w} L(i, j, w) + d_{i^*j^*}. \hspace{1cm} (3)
\]

The total number of elementary operations involved in (2) and (3) is \( O(n^2W) \). As such, we can solve the CPCPTSP by dynamic programming in \( O(n^2W) \) time.

Our implementation is a pulling algorithm in which we consider the indices \((i, j, w)\) in lexicographic order. For a given \((i, j)\), we calculate \(L(i, j, w)\) for \(w = 1, \ldots, W\), and then attempt to eliminate some of these labels before proceeding to the next \((i, j)\). The elimination is done based on the dominance relation that \(L(i, j, w_1)\) dominates \(L(i, j, w_2)\) if \(L(i, j, w_1) \leq L(i, j, w_2)\) and \(w_1 < w_2\). According to our experience, this may eliminate a significant part of the labels and may therefore also increase the overall computational speed of the dynamic programming algorithm.

As a heuristic alternative to the exact dynamic programming algorithm, the weight constraint may be taken into account in a Lagrangean fashion by redefining the edge lengths in \( G \). Given a multiplier \( \lambda \geq 0 \), we increase the length of any edge \((i, k)\) (where \( i < k \)) by the amount \( \lambda w_i \). Letting \( \Lambda(\lambda) \) denote the length of the shortest pyramidal route in \( G \) using the modified edge lengths, the Lagrangean dual is the problem of finding a multiplier \( \lambda \geq 0 \) that maximizes \( \Lambda(\lambda) - uW \). For a given \( u \), the value of \( \Lambda(\lambda) \) can be computed by dynamic programming in \( O(n^2) \) time, effectively by using the recurrence in (2) without the weight index.

This particular Lagrangean dual can be solved by specialized algorithms taking into account that we have dualized only a single constraint. We have implemented the algorithm by Handler and Zang (1980) for finding \( u^* \).

### 3. Assignment heuristic for the PCVRP

The special structure of the PCVRP, including the symmetry of costs, makes it possible to use a particular linear assignment problem (LAP) for generating columns which possess certain attractive characteristics. Our purpose of using LAP is to be able to generate promising columns at the early stage of a BP algorithm.

The idea is as follows. We heuristically choose a set of peak customers, and solve the PCVRP without the capacity constraint subject to that fixed set of peak customers. This can be done in polynomial time using the LAP. If the LAP solution contains a route with excess demand, we choose one of its customers as a new peak, and resolve the LAP with the expanded set of peak customers. When all routes are feasible, we use these as columns in the BP algorithm. An attractive characteristic of each route in the LAP solution is that its cost is minimized, i.e., the customers on each route are sequenced optimally under the pyramidal constraint.

In the following we firstly describe how the LAP is solved for a given set of peak customers, and secondly we describe how we choose the set of peak customers.

Let \( F = (f_1, \ldots, f_n) \), where \( \{n\} \in F \), denote a given set of peak customers. We seek an optimal solution to the PCVRP without the capacity constraint, containing \( |F| \) routes such that each route has one of the customers in \( F \) as its peak.

The corresponding LAP is given by the cost matrix \( \Gamma = (\gamma_{ij}) \) containing \( n + |F| \) rows and \( n + |F| \) columns. Each customer \( i \in V_c \setminus F \) is represented by row \( i \) and column \( i \), each peak customer \( f_i \) is represented by rows \( f_i \) and \( n + i \), and the depot is represented by columns \( f_1, f_2, \ldots, f_n \) and \( n + 1, \ldots, n + |F| \). The costs are defined as follows:

\[
\gamma_{ij} = \begin{cases} 
  c_{ij} & \text{for } j \in V_c \setminus F, j < i \leq n \\
  \infty & \text{for } j \in V_c \setminus F, i < j \\
  c_{f_0} & \text{for } i \leq n, j \in F \cup \{n + 1, \ldots, n + |F|\} \\
  c_{f_i, i} & \text{for } i > n, j \in V_c \setminus F, j < f_i, n \\
  \infty & \text{for } i > n, j \in V_c \setminus F, f_i, n \leq j \\
  c_{f_i, j} & \text{for } i > n, j \in F \cup \{n + 1, \ldots, n + |F|\}
\end{cases} \hspace{1cm} (4)
\]

By construction of the cost matrix, any LAP solution of finite cost contains two paths from each peak to the depot, all \( 2|F| \) paths being customer disjoint. Reversing the direction of one of the paths from each peak yields a collection of \( |F| \) pyramidal routes. This can be done without changing the total cost, as the PCVRP is defined as being undirected. Moreover, subtours are not possible in a finite cost solution, so the \( |F| \) routes collectively visit all customers.

Using a straightforward implementation of the Hungarian algorithm, for example the APC code by Carpaneto et al. (1988), the LAP can be solved in \( O(n^3) \) time. However, we have specialized the code for solving the LAP, taking into account the special structure of the cost matrix. The resulting algorithm, which is a specialized version of the APC code, runs in \( O(|F|n^2) \) time. Specifically, we first solve the LAP for \( F = \{n\} \) and then iteratively add the remaining peaks to \( F \).

To solve the LAP with \( F = \{ n \} \), we first identify the LAP solution for only one path from customer \( n \) to the depot. The solution to this LAP is trivially \( (1, n), (2, 1), (3, 2), \ldots, (n, n - 1) \). Letting \( u \) and \( v \) denote the dual variables associated with rows and columns, respectively, we have the following optimal dual values: \( u_{00} = \gamma_{n,n-1} \).
Let \( v_i = u_{i+1} + \gamma_{i+1} \) for \( i = n - 1, \ldots, 2; u_1 = u_2 + \gamma_2; v_n = 0; \) and \( v_j = -(u_{j+1} + \gamma_{j+1}) \) for \( j = 1, \ldots, n - 2 \). Then we add row \( n + 1 \) representing customer \( n \), and column \( n + 1 \) representing the depot, and perform a single iteration of the Hungarian algorithm in \( O(n^2) \) time. (We note that this is in fact an alternative \( O(n^2) \) algorithm for solving the PTSP with symmetric costs.)

Subsequently we repeatedly add a peak customer until the resulting assignment solution is feasible, i.e., such that the demand on each route does not exceed the vehicle capacity. This process contains two main operations, one for choosing a peak from a given set of candidates, and another for choosing the candidate sets. A description of these two operations follows.

A bit more notation is needed for the following. For any set of customers \( S \subseteq V_c \), let \( k(S) \) denote \( \sum_{i \in S} \theta_i \). Clearly, \( k(S) \) is a lower bound on the number of vehicles needed to service all customers in \( S \).

The choice of candidate sets is divided into two stages. In the first stage, we choose \( k(V_c) - 1 \) additional peaks as follows. For \( p = 2, \ldots, k(V_c) \), let \( \theta_p \) denote the largest customer index such that \( k(\{\theta_p, \ldots, n\}) = p \), and let \( \theta_p = \{\theta_p, \ldots, n\} \). Each of the \( p \) (or more) routes visiting at least one customer in \( \theta_p \) has its peak customer in \( \theta_p \). As such, \( \theta_p \) contains at least \( p \) peaks in any feasible PCVRP solution. As a consequence of this observation, we choose for \( p = 2, \ldots, k(V_c) \) a peak among those customers in \( \theta_p \) which have not already been chosen as a peak.

In the second stage, we repeatedly choose, among the routes in the latest LAP solution, the route with largest demand, as long as this demand exceeds the vehicle capacity. Suppose that the route has a demand which exceeds \( Q \), let \( R \) denote the set of customers on the route, and let \( p \) denote its peak. We begin with the set \( S = R \setminus \{p\} \) and then repeatedly remove the smallest indexed customer from \( R \) as long as the resulting \( S \) satisfies \( k(S \cup \{p\}) > 1 \). The final \( S \) is then our candidate set.

The second main operation is the choice of a peak from a given candidate set \( S \). We implemented a randomized choice of the peak, such that repeated use of the overall procedure may lead to different solutions. A central part of this operation is to assign probabilities to candidate nodes. For this, the following cost measure for each candidate node \( j \in S \) is used:

\[
Y_j = \min_{i \in \{1, \ldots, n\}} \{c_{ij} + c_{i0} - c_{0i}\}
\]  

(5)

The value of \( Y_j \) is taken as a measure of the cost of inserting customer \( j \) into a pyramidal route such that customer \( j \) is not the peak. As such, a large value of \( Y_j \) suggests that it would be sensible to choose customer \( j \) as a peak. For completeness, we define \( Y_j = \max(\epsilon, Y_j) \) in order to handle nonpositive insertion costs, where \( \epsilon \) is some small positive number. Finally, we let the probability of choosing node \( j \) be proportional to \( Y_j \).

4. Branch-and-cut-and-price

The PCVRP can be formulated as an SPP, in which variables (columns) represent pyramidal routes. We call a route feasible if and only if the total demand on the route does not exceed the vehicle capacity. Let \( \mathcal{R} \) denote the set of all feasible pyramidal routes. Further, let \( \lambda_c \) be a parameter of value 1 if route \( c \) visits customer \( i \) and 0 otherwise, let \( \lambda_r \) be a variable of value 1 if route \( r \) is chosen and 0 otherwise, and let \( d_r \) be the cost of route \( r \). This leads to the following SPP formulation:

\[
\min \sum_{c \in \mathcal{R}} d_c \lambda_c
\]  

s.t. \[
\sum_{c \in \mathcal{R}} \lambda_c \lambda_r = 1 \quad \forall r \in V_c
\]  

(7) \[ \lambda_r \in \{0, 1\} \quad \forall r \in \mathcal{R} \]  

(8)

This model can be solved by branch-and-bound, where the LP relaxation in each subproblem is solved by column generation (CG). The resulting algorithm is a branch-and-price (BP) algorithm.

The column generation subproblem is the problem of determining a feasible pyramidal route of minimum reduced cost, for a given vector of dual prices associated with (7). This is exactly the CPCPTSP as presented in Section 2, where the dual prices of (7) are used as prizes, and customer demands are used as vertex weights.

The initial LP contains the \( n \) single-customer routes. Further, we initially make \( n \) repetitions of the LAP heuristic, and add all resulting columns (after removal of duplicates) to the LP.

Subsequently, for column generation after each LP reoptimization, we first call the Lagrangean relaxation heuristic, and only if this fails do we call the exact dynamic programming algorithm.

For branching we choose an edge \( \{i, j\} \) in the underlying flow network which minimizes the expression \( |x_j - \frac{1}{2}(i - j)| \), where \( x_j \) is the current flow along edge \( \{i, j\} \). The untraditional multiplier \( i - j \) is due to the structure of pyramidal routes. In particular, if \( i = j + 1 \), prohibiting this edge implies that \( i \) and \( j \) cannot be visited on the same route unless either (i) the peak of the route is visited in between \( i \) and \( j \), or (ii) \( i \) or \( j \) is the peak of the route. This makes the usually weaker 0-branch appear more tightly constrained than with ordinary routes.

Our node selection strategy is ‘best-bound’, i.e., when choosing a subproblem to be processed next, we choose the subproblem with the smallest lower bound.

When the BP algorithm has found an optimal PCVRP solution, we attempt to find an even better CVRP solution by solving a TSP for each route in the PCVRP solution. For this we have implemented a simple branch-and-cut algorithm for the symmetric TSP, in which only subtour elimination constraints are separated at each subproblem. The computational burden of this postprocessing step is negligible relative to the overall BP algorithm.

For our BCP algorithm we use a formulation of the PCVRP which is obtained by modifying the CVRP formulation used in Fukasawa et al. (2006). The key difference is our definition of routes which involves the pyramidality constraint. We use only one class of cuts, namely the rounded capacity inequalities which are separated using the CVRPSEP package (Lysgaard, 2003).

In our implementation, each subproblem is processed by first doing column generation until no more columns are generated, after which we call the cut separation routine. If this succeeds in finding at least one violated cut, we reoptimize the LP and repeat the process, again beginning with column generation. The process stops when both column generation and separation fail on the same LP solution.

The other ingredients, including the node selection strategy and the choice of branching, are the same as those in the BP algorithm. Although it would be possible to branch on sets as in (Lysgaard et al., 2004; Fukasawa et al., 2006), we preferred to use the same branching strategy as in the BP algorithm in order to isolate the effect of adding cuts.

5. Computational results

We have applied our BP and BCP algorithms on a set of test instances all of which are available from www.branchandcut.org. More specifically, we give results for all \( \lambda \) and \( \mathcal{R} \) instances and for two \( \mathcal{R} \) instances. The optimal CVRP solution is known for all these instances, which makes it possible to investigate the deviation from CVRP optimality that results from PCVRP solutions.

When producing the PCVRP instance from the CVRP instance, we use the indexing rule as mentioned in Section 1, i.e., customers in the PCVRP instance are indexed so that \( c_{(i,j)} < c_{(i',j')} \) for \( 1 \leq i < j \leq n \). As a tie-breaking rule, for any two customers with the same distance to the depot, the smaller-indexed customer in the CVRP is also the smaller-indexed customer in the PCVRP.
In our experiments we fix the number of routes at the minimum possible by adding a constraint on the sum of all route variables to the SPP formulation.

The computations were done on a PC with a 2.0 GHz Intel Centrino Duo processor and 512 MB RAM running under Microsoft Windows XP. We used the ILOG CPLEX 9.1 callable library and the Microsoft Visual Studio 2005 C/C++ compiler.

Table 1 shows the results. Column ‘Name’ gives the name of the instance, and column ‘CVRP OPT’ gives the optimal CVRP objective value. Column ‘PCVRP OPT’ gives the optimal PCVRP objective value, and column ‘PCVRP OPT + TSP’ gives the objective value after resequencing the routes in the PCVRP solution, i.e., the value obtained by solving a TSP for the customer set (and the depot) of each route in the PCVRP solution. Column ‘Dev. (%)’ gives the deviation between ‘CVRP OPT’ and ‘PCVRP OPT + TSP’ in percent of the ‘CVRP OPT’ value, i.e., a measure of quality of the ‘PCVRP OPT + TSP’ solution when viewed as a heuristic solution to the CVRP (a blank entry represents a deviation of zero). A ‘–’ in columns ‘CVRP OPT’, ‘PCVRP OPT’ and ‘Dev. (%)’ represents that the problem was not solved to optimality.

The next five columns give the results for the BP algorithm. Column ‘Lagr.’ gives the objective value obtained at the root node of the search tree when the Lagrangean relaxation heuristic fails for the first time, i.e., without using the exact dynamic programming algorithm for column generation. Columns ‘Root LB’ and ‘Root time’ give the final lower bound, and the computing time in seconds, for the root node of the search tree. Columns ‘BB time’ and ‘BB tree size’ give the total computing time in seconds for solving the problem to optimality, and the size of the branch-and-bound tree. The final four columns, which show the results for the BCP algorithm, have the same meaning as the last four under the ‘Branch-and-price’ heading.

We draw two main conclusions from Table 1. First, the ‘PCVRP OPT + TSP’ solution does indeed come very close to the CVRP optimum, and in reasonable time. On many instances it coincides with the CVRP optimum, and the difference is very small on a considerable part of the remaining instances. This, in turn, suggests that the column generation procedures that are presented in this paper may also be useful as heuristics in column generation for the CVRP. Second, it fails for the first time, i.e., without using the exact dynamic programming algorithm for column generation.

As such, the addition of cuts has proven useful although only elementary routes are involved in the BP algorithm.
6. Conclusions

This paper has introduced the pyramidal capacitated vehicle routing problem (PCVRP) as a restricted version of the capacitated vehicle routing problem (CVRP). The added pyramidality restriction has led to pseudo-polynomial column generation of elementary routes. Computational results show that the optimal PCVRP solution tends to be very close to the optimal CVRP solution, in particular after resequencing the customers on each route in the PCVRP solution. The results also show that the BCP algorithm performs consistently better than the BP algorithm.

The encouraging results in this paper suggest two directions for further research. First, the introduced pseudo-polynomial algorithm for column generation of pyramidal routes could be used as a heuristic in an exact column generation of elementary routes for the CVRP. The results in this paper indicate that such a heuristic would indeed have a very good performance. Second, a similar restriction on route shape could be considered for other related routing problems. In particular, a similarly restricted version of the capacitated open vehicle routing problem (COVRP) (Letchford et al., 2007) could be obtained by requiring that customers are visited in nondecreasing order of distance from the depot. In that case it would be possible to obtain a dynamic programming algorithm for exact column generation that runs in \( O(n^2Q) \) time, i.e., a factor of \( n \) faster than for closed routes. We leave these and related issues for further research.

Acknowledgement

The author is grateful to an anonymous referee whose helpful comments led to an improved paper.

References


