European Journal of Operational Research 236 (2014) 800–810

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

A branch-and-cut-and-price algorithm for the cumulative capacitated vehicle routing problem

Jens Lysgaard [*,1], Sanne Wøhlk [1]

CORAL, Department of Economics and Business, Aarhus University, Fuglesangs Allé 4, DK-8210 Århus, Denmark

ARTICLE INFO

Article history:
Available online 30 August 2013

Keywords:
Routing
Minimum latency
CCVRP
Branch-and-cut-and-price

ABSTRACT

In this paper we consider the Cumulative Capacitated Vehicle Routing Problem (CCVRP), which is a variation of the well-known Capacitated Vehicle Routing Problem (CVRP). In this problem, the traditional objective of minimizing total distance or time traveled by the vehicles is replaced by minimizing the sum of arrival times at the customers. We propose a branch-and-cut-and-price algorithm for obtaining optimal solutions to the problem. To the best of our knowledge, this is the first published exact algorithm for the CCVRP. We present computational results based on a set of standard CVRP benchmarks and investigate the effect of modifying the number of vehicles available.

1. Introduction

The Capacitated Vehicle Routing Problem (CVRP) is one of the most well-studied problems within the area of transportation optimization. Recently it has, however, become clear to the research community that the CVRP does not fully capture the essence of real life transportation problems. This has for instance led to the introduction of so-called Rich Vehicle Routing Problems (Hartl, Hasle, & Janssens, 2006), a family of problems which capture the complications of real life problems far better than the classical CVRP.

One aspect of rich vehicle routing problems is the consideration of objective functions that differ from the traditional one of minimizing the total distance or time traveled by the vehicles. These include minimizing the number of vehicles used and minimizing the length of the longest tour. The latter is known as a Min–Max objective and is studied by Golden, Laporte, and Taillard (1997) and Applegate, Cook, Dash, and Rohe (2002), among others. Other studies consider simultaneous optimization of multiple objectives (Bowerman, Hall, & Calamai, 1995; Corberán, Fernández, Laguna, & Martí, 2002).

In this paper we consider the variation of the CVRP where the objective is to minimize the sum of arrival times at the customers, for a fixed starting time of each route. This problem is known as the Cumulative Capacitated Vehicle Routing Problem (CCVRP). We note that minimizing the sum of arrival times is equivalent to minimizing the average arrival time.

The CCVRP occurs in several applications. It is relevant in distribution systems where it is desirable to provide early service measured across the whole set of customers. In school bus routing, for example, minimizing average arrival time is one fairness measure which may have priority over minimizing total distance traveled. A more detailed discussion of performance criteria in the context of school bus routing is provided in Bowerman et al. (1995). Furthermore, when natural disasters strike, it is essential that aid arrives quickly in order to save lives and provide emergency supplies, so the traditional goal of cost minimization must step aside for fast response and fairness. Several performance measures can be used in relation to providing aid to multiple locations quickly, and minimizing the latest arrival time or minimizing the average arrival time are among the commonly used. The effect on the quality of one objective function when optimizing another is investigated by Campbell, Vandenbussche, and Hermann (2008) in the context of relief effort.

The outline of our paper is as follows. We first consider related literature in Section 2. Our mathematical model formulation is presented in Section 3, our algorithm is described in Section 4, and computational results are given in Section 5. Finally, we present the conclusion and perspectives in Section 6.

2. Related literature

The CCVRP has recently been studied from a heuristic point of view in several papers. These studies include Iterated Local Search (Chen, Dong, & Niu, 2012), Adaptive Large Neighborhood Search (Ribeiro & Laporte, 2012), Memetic algorithms (Ngueveu, Prins, & Wolfler Calvo, 2010), and a two-phase heuristic (Ke & Feng, 2013). In Ke and Feng (2013), the performance of three algorithms

[*] Corresponding author. Tel.: +45 8716 4898.
E-mail addresses: lys@asb.dk (J. Lysgaard), sanw@asb.dk (S. Wøhlk).
[1] Supported by NordForsk Project No. 25900.

0377-2217/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2013.08.032

(Ke & Feng, 2013; Ngueveu et al., 2010; Ribeiro & Laporte, 2012) were compared. Based on that comparison, the two-phase algorithm (Ke & Feng, 2013) and the Adaptive Large Neighborhood Search (Ribeiro & Laporte, 2012) each provide the best known solution for about half of the test instances.

Kara, Kara, and Yetiş (2008) consider a variation of the CCVRP, where the objective function to be minimized is not the sum of arrival times, but rather the sum of arrival times multiplied by the demand of the node. They refer to this problem, which is an extension of the CCVRP, as CumVRP and study the relationship to various other problems. Flow based formulations of the problem for both the delivery and the collection case are presented and based on these, the authors are able to solve instances with up to 34 locations.

The uncapacitated version of the CCVRP is known as the k-traveling repairman problem, where k is the number of vehicles available. For this problem, Fakcharoenphol, Harrelson, and Rao (2007) present an 8.497-approximation algorithm which is partly based on a result due to Chaudhuri, Godfrey, Rao, and Talwar (2003). Jothi and Raghavachari (2007) study an extension of the k-traveling repairman problem in which there is a repair time in addition to the travel time. They present a $(\frac{3}{2}\beta + \frac{1}{2})$-approximation algorithm for this problem, where β is the approximation factor obtainable for the k-traveling repairman problem.

The related single vehicle problem, referred to as the Minimum Latency Problem (MLP), has attracted many researchers. It is well studied both from an approximation and an exact point of view. The current best approximation algorithm for the MLP is due to Chaudhuri et al. (2003) and achieves an approximation factor of 3.59. Several exact approaches have been proposed for the MLP, most of which are based on dynamic programming, branch-and-bound, or a combination of the two (Wu, Huang, & Zhan, 2004). We refer the reader to Silva, Subramanian, Vidal, and Ochi (2012) for an overview of research on the MLP.

Dewilde, Cattrysse, Coene, Spieksma, and Vansteenwegen (2013) study a single vehicle problem, where a profit is obtained the first time a location is visited and the visit of each location is optional. The objective is to maximize the sum of profits minus the sum of arrival times. This problem arises as the subproblem if the CCVRP is solved using column generation approaches, except from the exclusion of capacity constraints. The authors present a tabu seach algorithm for this problem.

Recently, some interesting problem variations, where the MLP is combined with another problem, have been studied. We point out a couple of such papers. Levin and Penn (2008) present a 16.31-approximation algorithm for a combination of MLP and machine scheduling where n jobs are to be processed on a single machine located at a plant and subsequently are to be delivered to n individual customer locations by a single vehicle. Processing times are given for the jobs and travel times are given among the customers and between the plant and the customers. The goal is to determine a production sequence for the jobs at the machine and determine the routing of the vehicle such that the sum of the delivery times of the jobs at the customers are minimized. We emphasize that it is fully allowed for the vehicle to pick up jobs from the plant several times. Li, Vairaktarakis, and Lee (2005) consider a variation of the same problem where several jobs can be associated with the same customer and the vehicle may or may not be capacity constrained.

Chakrabarty and Swamy (2011) study a combination of MLP and facility location. Given a central depot, a set of customers, and a set of possible facilities, the problem is to determine which facilities to activate. The objective function to be minimized is combined of three terms: a fixed cost for each activated facility, a cost of assigning each customer to a facility (this is not a routing cost), and a minimum latency cost of a tour connecting the depot to the facilities.

3. Model formulation

The CCVRP can be defined as follows. 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 travel time between vertices i and j is denoted by t_{ij}. A number K of identical vehicles, each of capacity $Q > 0$, is available. Each customer i has an integer demand q_i, with $0 < q_i \leqslant Q$. Each customer must be served by a single vehicle and no vehicle can serve a set of customers whose demand exceeds its capacity. Each vehicle used must leave the depot at time 0, visit one or more customers, and return to the depot. The objective is to minimize the sum of all n arrival times at the customers.

In the following subsections we first describe two individual formulations (a Set Partitioning and a Vehicle Flow formulation, respectively) which are then combined into the formulation that we solve by our algorithm.

3.1. A Set Partitioning formulation

We define a feasible *elementary* route as a path $(0, z_1, \ldots, z_k, 0)$, where $z_1, \ldots, z_k$ are k different customers whose total demand does not exceed the vehicle capacity Q. As such, any feasible elementary route starts and ends at the depot, and we use the convention that the starting time at the depot is zero for any route.

For any feasible elementary route r, we define its cost c_r as the sum of arrival times for all customers on the route. Further, let $\Re$ denote the set of all feasible elementary routes. Moreover, we let α_{ir} be a parameter of value 1 if route r visits customer i and 0 otherwise, and we let λ_r be a variable of value 1 if route r is chosen and 0 otherwise. This leads to the following Set Partitioning formulation:

(SPP)

$$\text{min}: \quad \sum_{r\in\Re} c_r \lambda_r \tag{1}$$

$$\text{s.t.}: \quad \sum_{r\in\Re} \alpha_{ir}\lambda_r = 1 \quad \forall i \in V_c \tag{2}$$

$$\sum_{r\in\Re} \lambda_r = K \tag{3}$$

$$\lambda_r \in \{0,1\} \quad \forall r \in \Re \tag{4}$$

The objective (1) minimizes the total cost of all routes. Constraints (2) ensure that each customer is contained in exactly one route, the constraint (3) specifies the required number of routes K, and (4) are the binary constraints on the decision variables.

This model can be solved by branch-and-bound (BB), where the Linear Programming (LP) relaxation in each subproblem is solved by column generation (CG). This would result in a branch-and-price (BP) algorithm where the CG subproblem is the problem of determining a feasible elementary route of minimum reduced cost, for a given set of dual prices associated with (2) and (3).

In this paper, however, we will develop a different formulation involving the same set of route variables, and apply branch-and-cut-and-price (BCP) on this formulation.

3.2. A vehicle flow formulation

While any feasible solution to the CCVRP can be described in terms of λ-variables as in (2)–(4), we also have the alternative of representing a solution as in the two-index solution space used in vehicle flow formulations of the CVRP (Laporte, 2009; Lysgaard, Letchford, & Eglese, 2004), which we reproduce here with the objective function intentionally omitted. Let x_{ij} denote the number of times a vehicle travels directly between vertices i and j. Moreover, for any $S \subset V$, let $\delta(S)$ denote the set of edges with exactly one end-vertex in S, where we for simplicity write $\delta(i)$ instead of

$\delta(\{i\})$. Further, for any customer set $S \subseteq V_c$, we let $k(S) = \lceil \sum_{i \in S} q_i / Q \rceil$ denote a lower bound on the number of vehicles required to service the customers in S. Finally, let $x(F)$ denote $\sum_{e \in F} x_e$ for any $F \subseteq E$. Then the constraints (5)–(9) describe a feasible solution to the CVRP.

$$x(\delta(i)) = 2 \quad \text{for } i = 1, \ldots, n \tag{5}$$

$$x(\delta(S)) \geqslant 2k(S) \quad \forall S \subseteq V_c, |S| \geqslant 2 \tag{6}$$

$$x(\delta(0)) = 2K \tag{7}$$

$$x_{ij} \in \{0,1\} \quad \text{for } i,j = 1, \ldots, n \tag{8}$$

$$x_{0j} \in \{0,1,2\} \quad \text{for } j = 1, \ldots, n \tag{9}$$

Constraints (5) are the degree equations, (6) are rounded capacity inequalities (RCIs), and (7) specifies the required number of routes K. Finally, (8) and (9) are the integrality constraints, where variables x_{0j} are permitted to take the value 2 in order to allow a vehicle to serve only a single customer. The formulation in (5)–(9) involves the requirement that each route begins and ends at the depot. In other words, in the CVRP each route is required to be *closed*, unlike an *open* route beginning at the depot and ending at a customer, as in the Capacitated Open Vehicle Routing Problem (COVRP) (Letchford, Lysgaard, & Eglese, 2007).

For the purpose of working only with closed routes in relation to the CCVRP, we defined the route set $\mathfrak{R}$ in Section 3.1 so that any route ends at the depot. Nonetheless, it is clear from the definition of route cost that the traversal back to the depot from the last customer does not influence the route cost at all. Indeed, our requirement for closed routes is entirely due to our interest in working with the same solution space as in the CVRP, as specified by (5)–(9).

3.3. A combined formulation

Following the modeling approach in Fukasawa et al. (2006), as also described in Laporte (2009), we can combine a formulation based on route variables, as in the SPP formulation, with the two-index vehicle flow formulation. Letting β_r^e denote the number of times route r traverses edge $e \in E$, we have the following coupling between flow variables and route variables:

$$x_e = \sum_{r \in \mathfrak{R}} \beta_r^e \lambda_r, \quad \forall e \in E. \tag{10}$$

The β-coefficients may take values as follows. For a feasible elementary route r with more than one customer we have that $\beta_r^e = 1$ for each edge e along the route, whereas $\beta_r^e = 2$ for $e = \{0, j\}$ if route r services only customer j.

We are then able to strengthen the SPP formulation by adding valid inequalities for the 2-index formulation. This leads to a formulation involving both a large number of rows and columns; the resulting formulation (SPP-VF) is given in (11)–(18).

(SPP – VF)

$$\min: \quad \sum_{r \in \mathfrak{R}} c_r \lambda_r \tag{11}$$

$$\text{s.t.}: \quad \sum_{r \in \mathfrak{R}} \sum_{e \in \delta(i)} \beta_r^e \lambda_r = 2 \quad \forall i \in V_c \tag{12}$$

$$\sum_{r \in \mathfrak{R}} \lambda_r = K \tag{13}$$

$$x(\delta(S)) \geqslant 2k(S) \quad \forall S \subseteq V_c, |S| \geqslant 2 \tag{14}$$

$$\sum_{r \in \mathfrak{R}} \beta_r^e \lambda_r - x_e = 0 \quad \forall e \in E \tag{15}$$

$$\lambda_r \in \{0,1\} \quad \forall r \in \mathfrak{R} \tag{16}$$

$$x_{ij} \in \{0,1\} \quad \text{for } i,j = 1, \ldots, n \tag{17}$$

$$x_{0j} \in \{0,1,2\} \quad \text{for } j = 1, \ldots, n \tag{18}$$

We do not attempt to solve (11)–(18) directly; instead, we use Eq. (15) to reformulate the model so that it contains only route variables. In this way the RCIs (14) are expressed in route variables, we avoid the $|E|$ constraints (15), and we obtain a much more compact formulation. The LP relaxation of the resulting formulation is given by (19)–(23), which we refer to as the *Dantzig-Wolfe Master* problem (DWM), as in Fukasawa et al. (2006).

(DWM)

$$\min: \quad \sum_{r \in \mathfrak{R}} c_r \lambda_r \tag{19}$$

$$\text{s.t.}: \quad \sum_{r \in \mathfrak{R}} \sum_{e \in \delta(i)} \beta_r^e \lambda_r = 2 \quad \forall i \in V_c \tag{20}$$

$$\sum_{r \in \mathfrak{R}} \sum_{e \in \delta(S)} \beta_r^e \lambda_r \geqslant 2k(S) \quad \forall S \subseteq V_c, |S| \geqslant 2 \tag{21}$$

$$\sum_{r \in \mathfrak{R}} 2\lambda_r = 2K \tag{22}$$

$$\lambda_r \geqslant 0 \quad \forall r \in \mathfrak{R} \tag{23}$$

The set of feasible solutions to the CCVRP is given by the set of integer feasible solutions to DWM. We adopt the general framework of BB for finding an optimal integer solution to DWM. However, the presence of both an exponential number of columns (λ-variables) and an exponential number of rows (due to the RCIs in (21)) implies that we resort to generating both columns and rows dynamically, i.e., by branch-and-cut-and-price.

4. The algorithm

We have developed a BCP algorithm for the exact solution of the CCVRP using the DWM formulation. As such, we implement a BB algorithm in which each node in the search tree consists of an LP problem which we solve by both column generation and cut separation.

In the following description of our algorithm, we focus on that part of the algorithm which in particular deals with the consequences of the particularities of the CCVRP relative to the CVRP. Essentially, the difference between CCVRP and CVRP lies in the definition of route cost, hence we will give priority to the description of how to solve the pricing subproblem, although we also briefly mention the other main ingredients of our BCP algorithm. For further details on BCP in relation to the CVRP we refer to (Fukasawa et al., 2006).

4.1. Initialization

We initialize the solution process by defining an LP containing only a small number of rows and columns.

The LP contains $n+1$ constraints, namely the n degree equations (20) together with the single constraint (22) specifying the number of routes.

Moreover, the LP contains a single-customer route for each customer. In addition, for the purpose of ensuring that the LP always has a feasible solution, we create a dummy column which, in any node of the BB tree, has a coefficient which is identical to the right-hand side in each of the rows in the LP. In this way, a feasible LP solution can be obtained simply by setting this dummy variable to one; obviously, its associated cost is given a prohibitive high value so that the column will be used only if necessary for the sake of feasibility.

Together with these $n+1$ columns, we add the columns produced by running a simple nearest neighbor (NN) heuristic, which works as follows. We start a route at the depot and iteratively extend the route from its current end point to the nearest yet unvisited customer subject to the vehicle capacity, where *nearest* is measured in terms of travel time. The route extension continues not necessarily as long as possible, but only until the route under construction has received an even share of the remaining demand. Specifically, the first route is completed as soon as its total demand is at least $\lceil \sum_{i \in V_c} q_i / K \rceil$. In general, if we let D^k denote the actual demand on the k'th route, then the required demand on route k is $\lceil (\sum_{i \in V_c} q_i - \sum_{i=1}^{k-1} D^i)/(K-k+1) \rceil$.

This heuristic rule for filling the individual vehicle is introduced as a very simple attempt to balance the routes in terms of route lengths and delivered quantities, which in broad terms appears to be useful in the CCVRP.

4.2. Pricing

Given an optimal solution to the current LP relaxation, we need to be able to determine whether or not there exists a feasible elementary route with negative reduced cost. If such a route exists, we also need to identify one or more such routes.

Under the conventional objective of minimizing route lengths, the reduced cost of a route can be calculated as the length of a path from source to sink in an appropriately defined graph, and so that the length of the path is simply the sum of the lengths of the individual arcs traversed along the path. This construction is applied extensively across a wide variety of problem types and algorithms, where the column generation subproblem is represented as a resource-constrained shortest path problem. For an extensive coverage we refer to (Desaulniers et al., 2005; Desrosiers, Dumas, Solomon, & Soumis, 1995, chap. 2).

However, under the objective in the CCVRP, the cost of traversing an edge is not defined independently from the route on which the edge is traversed. Basically, in order to evaluate the cost of traveling directly from i to j, we need to know not only the travel time from i to j, but also how many customers still remain to be visited on the route, as the time spent traveling from i to j adds to the arrival time at each of the remaining customers on the route.

Due to this untraditional cost structure we find it appropriate to give a detailed description of our pricing calculations in the following. We describe the pricing in two stages, where we first consider how the cost of a route can be calculated, and secondly we describe how we modify the route cost calculation so as to obtain the *reduced* cost of a route.

4.2.1. Route costs

The total cost c_r of a feasible elementary route $(0 = z_0, z_1, \ldots, z_k, 0)$ is the sum of the k arrival times at the customers along the route:

$$c_r = \sum_{j=1}^{j=k} T_{z_j} \tag{24}$$

where the arrival time T_{z_j} at customer z_j is calculated as follows:

$$T_{z_j} = T_{z_{j-1}} + t_{z_{j-1}z_j} \quad \text{for } 1 \leqslant j \leqslant k \tag{25}$$

with $T_0 = 0$. The recursive nature of (25) highlights the fact that the travel time from z_{j-1} to z_j is included not only in the arrival time at z_j but in all arrival times at the $k - j + 1$ last customers on the route.

This suggests an alternative way of calculating the sum of arrival times, namely as a sum of contributions, one for each edge traversed from the depot until the last customer on the route, and where the travel time from z_{j-1} to z_j is weighted by the number of remaining customers $(k - j + 1)$ on the route:

$$c_r = \sum_{j=1}^{j=k} (k-j+1) t_{z_{j-1}z_j} \tag{26}$$

If we would design a procedure where we would build up a route by repeatedly appending the edge $\{z_{j-1}z_j\}$ to a partial route $(0, z_1, \ldots, z_{j-1})$, then it is seen from (26) that it would be required to know the number of remaining customers on the route, in order to determine the cost contribution from traversing the edge $\{z_{j-1}$-$z_j\}$. Not only would this be impractical, but it also leads to our idea of basically taking the opposite approach, namely to build a route sequentially in reverse order, i.e., to begin with the end depot and then repeatedly add an edge before the current first vertex on the route.

Hence, let us again consider an elementary feasible route with k customers, but where we now let the subscript increase from the last to the first customer. As such, we can write the route as $(0 = v_{k+1}, v_k, \ldots, v_1, 0)$, i.e., v_j is the j'th last customer on the route. The route cost can now be calculated as follows, where the travel time from v_{j+1} to v_j is weighted by the remaining number (j) of customers:

$$c_r = \sum_{j=1}^{j=k} j t_{v_{j+1}v_j} \tag{27}$$

The important characteristic of (27) is that the cost contribution from each individual edge is determined without using information on the preceding part of the route. This allows us to calculate the cost contribution from each edge immediately while building up a route in reverse order.

Using the same fundamental idea of building paths in reverse order, we can define a convenient set of labels for our path calculations to be used in our pricing of routes. Specifically, we let $L(i, a, q)$ denote the cost of the minimum cost path from vertex i to the depot, containing a arcs, and having a total demand of q including the demand of vertex i. Throughout the calculations we use $q_0 = 0$.

We determine an upper bound $\bar{a}$ on a based on how many customer demands could possibly be loaded onto the same vehicle. Specifically, $\bar{a}$ is the largest integer for which the sum of the $\bar{a} - 1$ smallest demands does not exceed the vehicle capacity, considering that we have $a - 1$ customers on a route with a arcs, starting and ending at the depot.

For computing the L-values we propose the following steps; we prefix the step numbers by 'C' to represent cost calculations, as opposed to the prefix 'RC' which we use later to represent calculations of reduced costs.

Step (C1). Set $L(i, a, q) = \infty$ for $i = 0, \ldots, n$; $a = 0, \ldots, \bar{a}$; $q = 0, \ldots, Q$.
Step (C2). Set $L(i, 1, q_i) = 0$, for $i = 1, \ldots, n$.
Step (C3). Compute path costs as follows:

$$\begin{aligned} &L(i,a,q) = \min_{j \in V_c \setminus \{i\}} ((a-1)t_{ij} + L(j, a-1, q-q_i)), \\ &\text{for } i = 0, \ldots, n; \ a = 2, \ldots, \bar{a}; \ q = q_i, \ldots, Q \end{aligned} \tag{28}$$

Proposition 1. *The values of L(i, a, q) are determined correctly by steps C1–C3, for all combinations of i, a, and q.*

Proof. In step C1, all entries are set to infinity which effectively eliminates labels for non-existing paths. In step C2 we set for each customer i the cost to zero for the combination $(i, 1, q_i)$; this is obviously the only combination that represents the final arc from customer i back to the depot, and the cost is zero since there are no customers remaining to be visited. Hence, the values determined for $a = 1$ are correct. With respect to the correctness of step C3, we note that for $a \geqslant 2$ any path starting from vertex i must necessarily go immediately to some customer j and from there complete the path along $a - 1$ arcs to the depot. Along the first arc from i to j we have $a - 1$ customers remaining to be visited, and the travel time t_{ij} adds to the arrival time to each of these $a-1$ customers, hence we get the contribution of $(a-1)t_{ij}$ from traversing the arc (i, j). In addition, completing the path to the depot must be made from j along $a-1$ arcs with the remaining demand of $q-q_i$, which by definition is done at minimum cost given by $L(j, a-1, q-q_i)$. □

4.2.2. Reduced costs

In order to describe how the reduced cost of a route is determined, we let the dual prices associated with (20)–(22) be denoted by μ, π, and ν, respectively.

In the DWM, Eq. (20) may be viewed as constraints (2) of SPP where both the left hand side and the right hand side have been multiplied by two. That is, a feasible elementary route will have a coefficient of two in each constraint of (20) representing a visited customer on the route, as we on the left hand side of (20) count the number of traversals in $\delta(i)$, which equals two for each visit to customer i. As such, in the calculation of reduced costs, μ_i must be subtracted twice for each visited customer i on the route.

Concerning constraints (21), we note that the coefficient on the left hand side equals the number of times that the route traverses an edge in $\delta(S)$, hence we must subtract π_S for each traversal of an edge in $\delta(S)$.

Finally, in relation to (22) we note that ν must be subtracted twice for each route, equivalent to subtracting ν once for each traversal of an edge in $\delta(0)$.

In order to embed these cost reductions into our pricing algorithm, we define $\tilde{c}_e$ as the cost reduction induced by each traversal of edge e, for each $e \in E$, and we compute $\tilde{c}_e$ as follows:

$$\tilde{c}_e = \begin{cases} \mu_i + \mu_j + \sum\limits_{S|\delta(S)\ni e} \pi_S & \text{for } e = \{i,j\} \in E \setminus \delta(0) \\ \nu + \mu_j + \sum\limits_{S|\delta(S)\ni e} \pi_S & \text{for } e = \{0,j\} \in \delta(0) \end{cases} \tag{29}$$

We can now express the reduced cost $\bar{c}_r$ of route r in the following form:

$$\bar{c}_r = c_r - \sum_{e\in E} \beta_r^e \tilde{c}_e \tag{30}$$

We wish to calculate reduced costs of routes by modifying the calculations of route costs, i.e., by making modifications to steps C1–C3. As C1–C3 calculates c_r, what remains to be done is to subtract $\tilde{c}_e$ for each traversal of edge e, for each $e \in E$. Accordingly, we let $\tilde{L}(i,a,q)$ denote the reduced cost of the minimum reduced cost path from vertex i to the depot, containing a arcs, and having a total demand of q including the demand of vertex i, where the reduced cost of a path is its cost (as given by $L(i, a, q)$) minus the cost reductions for the edges along the path, as given by (29). Effectively, this leads to steps RC1-RC3, where we use the prefix 'RC' to represent calculations of *reduced* costs.

Step (RC1). Set $\tilde{L}(i,a,q) = \infty$ for $i = 0, \ldots, n$; $a = 0, \ldots, \bar{a}$; $q = 0, \ldots, Q$.
Step (RC2). Set $\tilde{L}(i,1,q_i) = -\tilde{c}_{i0}$, for $i = 1, \ldots, n$.
Step (RC3). Compute reduced path costs as follows:

$$\tilde{L}(i,a,q) = \min_{j\in V_c\setminus\{i\}} ((a-1)t_{ij} - \tilde{c}_{ij} + \tilde{L}(j,a-1,q-q_i)),$$
$$\text{for } i = 0,\ldots,n;\ a = 2,\ldots,\bar{a};\ q = q_i,\ldots,Q \tag{31}$$

In step RC2, the final edge back to the depot has a reduced cost equal to its cost reduction, as its original cost is zero. With respect to the recursion in RC3 we note that the calculation is essentially the same as in step C3, except that we now use $(a-1)t_{ij} - \tilde{c}_{ij}$ instead of $(a-1)t_{ij}$, i.e., the cost reduction of $\tilde{c}_{ij}$ is embedded in the recursion. Correctness of steps RC1-RC3 follows by the same line of reasoning as in the proof of Proposition 1.

We note that whole routes, i.e., paths that start at the depot, are associated with values of $\tilde{L}(i,a,q)$ for which $i = 0$.

The recursion in step RC3 does not guarantee the resulting paths to be elementary. Indeed, we have adopted the strategy of allowing paths to be non-elementary. However, we eliminate the occurrence of 2-cycles, as in many previous approaches to column generation in vehicle routing (see, e.g., Christofides, Mingozzi, & Toth, 1981; Irnich & Desaulniers, 2005, chap. 2). For the sake of brevity, we do not describe the details of the well-established technique of 2-cycle elimination.

Permitting non-elementary paths allows for a pseudo-polynomial running time. The complexity of our CG procedure is $O(\bar{a}n^2Q)$.

4.3. Cut separation

As observed in Section 3.2, in the space of the x-variables we may use any of the inequalities that are valid for the CVRP. However, we use only one class of cuts, namely the RCIs (Lysgaard et al., 2004), as this in a BCP framework seems to be by far the most effective class in practice for the CVRP (see Fukasawa et al., 2006). We use the CVRPSEP package (Lysgaard, 2003) for separating the RCIs.

4.4. Other ingredients

In our implementation, each BB node 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 no columns as well as no RCIs are produced for the given LP solution.

In the first call to column generation at each BB node, we run the NN heuristic in Section 4.1 using modified travel times, i.e., we replace each t_{ij} by $t_{ij} - \tilde{c}_{ij}$ in order to let the routes be influenced by the current dual prices. Only in the first call to column generation do we use the heuristic, in all subsequent iterations do we call our dynamic programming procedure described in Section 4.2.

In each call to our dynamic programming procedure we allow a maximum of $5n$ columns to be returned. These are selected as follows. Given that all $\tilde{L}(i,a,q)$ values have been calculated as described in Subsection 4.2.2, we loop through the a-values in decreasing order, and for each a value we loop through all customer indices i. For a given combination of a and i, we identify the route $(0, i, \ldots, 0)$ with minimum reduced cost among routes with $a + 1$ arcs (i.e., with a arcs from i to 0); as such, we consider only the best q-value for each pair of a and i. When all combinations of a and i have been examined, or as soon as $5n$ routes with negative reduced cost have been identified, the set of identified routes with negative reduced cost is returned from the dynamic programming procedure. All these routes are then added as columns to the LP.

With respect to the branching strategy, we choose to branch on sets as in (Fukasawa et al., 2006; Lysgaard et al., 2004), i.e., to branch on the disjunction $(x(\delta(S)) = 2 \vee x(\delta(S)) \geqslant 4)$ in the underlying flow network, where S is a set of customers. Given that we use this disjunction, we must use a customer set S so that $2 < x^*(\delta(S)) < 4$ in order to make the current LP solution infeasible in both subproblems. We use (Lysgaard, 2003) to identify a single customer set S. The procedure that we call takes as input the *target* value that we want $x^*(\delta(S))$ to be close to, and we simply use 3.0 as the target value of $x^*(\delta(S))$. It may well happen that $|S| = 2$, which is just the special case of branching on a single edge.

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

5. Computational results

We have applied our BCP algorithm on CVRP test instances all of which are available from www.branchandcut.org. More specifically, in Subsections 5.1 and 5.2 we give results for all A, B, E, M, and P instances. We use distances rounded to integer values as travel times in all these instances. For these experiments we have set a maximum computing time of one hour.

In Subsection 5.1 we report our results obtained by fixing the number of routes (i.e., the value of K in (3)) at the minimum possible, taking the vehicle capacity and customer demands into account; this number equals the optimal objective value for a Bin Packing Problem with bin capacity given by the vehicle capacity and item sizes given by customer demands.

In Subsection 5.2 we report our results obtained by increasing K by one, i.e., we insist on using one more than the minimum number of routes. Provided that travel times satisfy the triangle inequality, the CCVRP objective value will generally improve as a result of increasing the number of routes. While the rounding convention used in the test instances implies that the triangle

Table 1
Computational results for A instances.

Name	UB	Root CG		Root CG and RCI		BCP		
		Obj.	Time	Obj.	Time	LB	Time	Nodes
A-n32-k5	2192*	2159.4	0.2	2184.8	0.3	2192	1	9
A-n33-k5	1725*	1687.1	0.1	1704.6	0.2	1725	2	79
A-n33-k6	1612*	1600.8	0.1	1610.4	0.2	1612	0	3
A-n34-k5	2104*	2060.1	0.2	2080.5	0.3	2104	2	35
A-n36-k5	2279*	2267.0	0.3	2270.7	0.3	2279	1	15
A-n37-k5	1970*	1912.8	0.4	1928.0	0.5	1970	14	169
A-n37-k6	2241*	2225.4	0.3	2228.0	0.4	2241	2	25
A-n38-k5	2084*	2028.1	0.3	2060.2	0.6	2084	7	75
A-n39-k5	2312*	2265.5	0.5	2285.9	0.8	2312	9	103
A-n39-k6	2216*	2181.0	0.4	2189.0	0.5	2216	3	41
A-n44-k6	2563*	2532.7	0.5	2536.1	0.6	2563	7	87
A-n45-k6	2848*	2739.9	4.3	2785.4	6.0	2848	171	609
A-n45-k7	2831*	2809.0	0.5	2812.2	0.6	2831	6	55
A-n46-k7	2373*	2354.0	0.5	2363.5	0.8	2373	4	27
A-n48-k7	3101*	3083.3	0.5	3101.0	0.7	3101	1	1
A-n53-k7	3115*	3064.8	1.2	3082.2	1.8	3115	31	97
A-n54-k7	3357*	3318.9	1.4	3326.3	1.8	3357	120	651
A-n55-k9	2588*	2539.7	0.6	2563.6	1.0	2588	40	385
A-n60-k9	3446*	3420.0	1.7	3424.4	2.5	3446	418	1557
A-n61-k9	3652	2786.4	1.4	2802.9	2.1	2847	–	2337
A-n62-k8	3925*	3902.1	2.0	3907.1	2.8	3925	34	105
A-n63-k10	3256*	3210.9	1.9	3238.7	2.9	3256	101	627
A-n63-k9	4630*	4585.4	1.9	4604.1	3.1	4630	663	1245
A-n64-k9	4135*	4105.7	2.1	4114.6	2.9	4135	242	679
A-n65-k9	3487*	3422.6	1.3	3448.0	2.4	3487	1683	1325
A-n69-k9	3528*	3489.3	2.4	3504.6	3.4	3528	80	291
A-n80-k10	7174	5872.8	7.5	5892.9	8.8	5922	–	3087

Table 2
Computational results for B instances.

Name	UB	Root CG		Root CG and RCI		BCP		
		Obj.	Time	Obj.	Time	LB	Time	Nodes
B-n31-k5	1830*	1815.5	0.1	1828.2	0.2	1830	0	5
B-n34-k5	2271*	2252.4	0.4	2263.7	0.6	2271	2	15
B-n35-k5	2846*	2760.6	0.2	2838.9	0.5	2846	3	41
B-n38-k6	2103*	2086.6	0.3	2102.0	0.5	2103	1	3
B-n39-k5	1968	1899.3	0.4	1934.7	1.0	1949	–	2503
B-n41-k6	2329*	2297.9	0.2	2313.8	0.6	2329	16	233
B-n43-k6	2123*	2099.3	0.3	2109.2	0.5	2123	18	183
B-n44-k7	2295*	2269.5	0.4	2292.5	0.8	2295	5	47
B-n45-k5	2386*	2298.9	1.5	2374.7	3.2	2386	16	65
B-n45-k6	–	1947.2	2.2	1998.0	3.5	2017	–	2325
B-n50-k7	2293	2179.3	1.1	2229.4	1.8	2257	–	6931
B-n50-k8	2953*	2941.8	0.7	2949.5	1.2	2953	7	63
B-n51-k7	3133*	2991.4	0.6	3115.9	1.9	3133	16	99
B-n52-k7	2573*	2521.5	2.3	2553.1	4.8	2573	3536	2837
B-n56-k7	2358*	2353.9	3.2	2358.0	3.8	2358	4	3
B-n57-k7	–	3786.7	31.6	3845.3	42.9	3863	–	2675
B-n57-k9	4500*	4489.0	0.8	4498.5	1.4	4500	5	25
B-n63-k10	4379*	4340.3	1.4	4366.5	3.1	4379	235	533
B-n64-k9	3222	2479.7	2.1	2579.6	4.2	2597	–	1871
B-n66-k9	4872	4075.6	3.6	4094.0	5.5	4109	–	3557
B-n67-k10	2871	2817.2	2.5	2842.9	4.5	2860	–	2859
B-n68-k9	4058*	4046.0	1.4	4054.8	2.5	4058	14	15
B-n78-k10	4809	3782.8	12.4	3813.1	21.8	3828	–	2477

inequality may just be violated by a single unit, the results are still expected to show a decrease in objective function value as a function of a larger number of vehicles.

Finally, in Subsection 5.3 we report our results from applying our BCP algorithm on the 'CMT' instances which have been used for testing recently published metaheuristics. Unlike the instances with integer distances in Subsections 5.1 and 5.2, the CMT instances have real distances. We allow a computing time of eight hours for the CMT instances.

All computations were done on a PC with a 2.53 GHz Intel Core 2 Duo P8700 processor and 1.86 GB RAM running under Microsoft Windows XP. We used the IBM ILOG CPLEX 12.2 callable library and the Microsoft Visual Studio 2005 C/C++ compiler. All computing times are reported in seconds.

5.1. Results with a minimum number of routes

Tables 1–4 show the results. Column 'Name' gives the name of the instance, where the two numbers in the instance name indicate the number of vertices including the depot, and the minimum possible number of routes, respectively. For example, the instance `A-n32-k5` has 31 customers and requires at least 5 routes.

Column 'UB' gives the upper bound, i.e., the objective value of the best known feasible solution, where a '*' means that the bound is proven optimal. A '–' represents that no feasible solution were found in our computations. In our algorithm, the only source of feasible solutions was the BCP algorithm itself, i.e., only when the LP solution in the individual BB node was integer and feasible did we record a feasible solution (possibly using routes produced by our NN heuristic).

The two columns under the heading 'Root CG' show the LP objective value and the associated computing time at the root node when it happens for the first time that no columns are produced for the given LP solution.

Similarly, the next two columns under the heading 'Root CG & RCI' show the LP objective value and the associated computing time at the root node when it happens for the first time that no columns as well as no RCIs are produced for the given LP solution.

Finally, the three columns under the heading 'BCP' show the overall results for the BCP algorithm, specifically the global lower bound, the total computing time (where a '–' represents that the

Table 3
Computational results for E and M instances.

Name	UB	Root CG		Root CG and RCI		BCP		
		Obj.	Time	Obj.	Time	LB	Time	Nodes
E-n22-k4	845*	839.5	0.0	841.0	0.0	845	0	3
E-n23-k3	1908*	1833.3	0.5	1861.8	1.0	1908	9	39
E-n30-k3	1987	1837.1	0.5	1919.5	0.8	1984	–	1711
E-n33-k4	2852*	2820.9	1.0	2830.7	1.3	2852	15	115
E-n51-k5	2213*	2190.9	2.4	2191.4	2.8	2213	26	103
E-n76-k7	3503	2870.5	10.4	2872.5	11.0	2905	–	2645
E-n76-k8	3674	2638.1	7.0	2640.1	7.7	2672	–	2469
E-n76-k10	3364	2325.4	4.5	2330.0	5.4	2357	–	2663
E-n76-k14	–	2052.3	5.8	2053.7	6.5	2075	–	2761
E-n101-k8	5166	3918.8	62.3	3925.0	68.6	3951	–	1289
E-n101-k14	4022	2919.3	34.2	2922.6	37.5	2941	–	3529
M-n101-k10	3566	3523.4	24.9	3530.2	27.6	3556	–	3267
M-n121-k7	8539	7031.1	189.3	7068.8	259.6	7089	–	727
M-n151-k12	6314	4884.5	260.9	4897.9	300.0	4912	–	669

Table 4
Computational results for P instances.

Name	UB	Root CG		Root CG and RCI		BCP		
		Obj.	Time	Obj.	Time	LB	Time	Nodes
P-n16-k8	396*	396.0	0.1	396.0	0.1	396	0	1
P-n19-k2	849*	821.9	0.1	844.0	0.1	849	0	13
P-n20-k2	924*	905.0	0.1	916.6	0.2	924	1	9
P-n21-k2	928*	922.5	0.2	928.0	0.2	928	0	1
P-n22-k2	991*	988.1	0.2	991.0	0.3	991	0	1
P-n22-k8	681*	676.0	0.0	681.0	0.0	681	0	1
P-n23-k8	616*	600.6	0.0	610.0	0.0	616	0	7
P-n40-k5	1541*	1529.7	0.7	1530.6	0.9	1541	5	33
P-n45-k5	1894*	1885.6	1.1	1888.1	1.3	1894	4	11
P-n50-k7	1554*	1552.5	0.6	1554.0	0.7	1554	1	1
P-n50-k8	–	1485.2	1.1	1489.8	1.3	1528	–	2637
P-n50-k10	1347*	1330.9	0.1	1333.7	0.2	1347	6	155
P-n51-k10	1487*	1455.7	0.3	1458.2	0.4	1487	2113	2563
P-n55-k7	1764*	1749.9	1.0	1752.8	1.2	1764	13	85
P-n55-k8	1768*	1757.6	0.7	1760.7	1.1	1768	5	21
P-n55-k10	1463*	1449.4	0.3	1454.7	0.5	1463	3	39
P-n55-k15	1414*	1384.4	0.1	1390.4	0.2	1414	51	465
P-n60-k10	1704*	1688.4	0.6	1691.3	0.9	1704	13	101
P-n60-k15	1509*	1486.9	0.2	1495.6	0.3	1509	3	89
P-n65-k10	1948*	1924.6	0.8	1927.0	1.0	1948	70	405
P-n70-k10	2121*	2095.7	1.7	2100.7	2.4	2121	73	243
P-n76-k4	6191	4491.0	126.3	4508.1	138.8	4553	–	923
P-n76-k5	5463	3728.3	68.8	3738.2	74.5	3775	–	1579
P-n101-k4	8707	6787.9	1889.3	6806.3	1994.8	6807	–	5

one hour time limit was reached), and the number of nodes in the search tree.

For example, the instance A-n32-k5 has an optimal objective value of 2192, and this is concluded by the BCP in 1 second using a BB tree with 9 nodes. At the root node of the BB tree, the lower bound from using only column generation is 2159.4 (found in 0.2 second), whereas the addition of RCIs increases the lower bound to 2184.8 (found in a total of 0.3 second).

For many instances, the combination of CG and RCIs produces much tighter bounds than those obtained with CG only (such as for A-n32-k5), whereas the CG bound for other instances (such as for E-n51-k5) is not improved much by the addition of RCIs. Viewed across all instances, however, we believe that the results confirm the effectiveness of combining CG and RCIs as in our BCP.

In general, we do find it encouraging that our algorithm is able to solve the majority of instances with a limited computational effort.

For the sake of completeness we note that the instance M-n200-k16 is omitted from Table 3, as the one hour of computing time was exceeded without reaching the situation where column generation produced no columns; moreover, no feasible solution was obtained.

Table 5
Computational results for A instances with an extra vehicle.

Name	UB	Root CG		Root CG and RCI		BCP		
		Obj.	Time	Obj.	Time	LB	Time	Nodes
A-n32-k5+1	2021*	2021.0	0.2	2021.0	0.2	2021	0	1
A-n33-k5+1	1571*	1545.8	0.1	1553.9	0.2	1571	2	71
A-n33-k6+1	1461*	1452.0	0.1	1458.5	0.2	1461	0	7
A-n34-k5+1	1879*	1854.0	0.2	1863.0	0.3	1879	2	35
A-n36-k5+1	2133*	2127.7	0.3	2129.4	0.3	2133	1	17
A-n37-k5+1	1747*	1732.5	0.3	1735.2	0.4	1747	4	41
A-n37-k6+1	2103*	2093.1	0.5	2097.0	0.6	2103	2	21
A-n38-k5+1	1794*	1768.8	0.4	1794.0	0.5	1794	0	1
A-n39-k5+1	2116*	2080.5	0.9	2081.0	0.9	2116	17	305
A-n39-k6+1	2058*	2049.5	0.4	2049.5	0.4	2058	1	9
A-n44-k6+1	2377*	2369.3	0.5	2371.8	0.7	2377	1	11
A-n45-k6+1	2527*	2462.5	0.5	2493.7	1.0	2527	15	219
A-n45-k7+1	2731*	2715.4	0.3	2719.3	0.5	2731	9	147
A-n46-k7+1	2242*	2237.0	0.4	2240.0	0.5	2242	1	3
A-n48-k7+1	3001*	2975.2	0.5	2991.9	0.6	3001	4	29
A-n53-k7+1	2897*	2886.8	0.8	2890.8	0.9	2897	7	17
A-n54-k7+1	3188*	3159.7	1.1	3164.1	1.4	3188	73	659
A-n55-k9+1	2455*	2443.9	0.4	2450.7	0.6	2455	3	25
A-n60-k9+1	3356*	3347.4	1.2	3350.0	1.5	3356	8	35
A-n61-k9+1	2640*	2603.6	0.9	2614.6	1.3	2640	143	639
A-n62-k8+1	3787*	3783.5	1.7	3783.5	1.7	3787	7	11
A-n63-k10+1	3146*	3113.3	2.1	3131.8	2.7	3146	61	393
A-n63-k9+1	4509*	4504.6	1.0	4508.3	1.5	4509	5	3
A-n64-k9+1	4024*	4012.5	1.9	4015.9	2.7	4024	21	127
A-n65-k9+1	3306*	3268.0	1.0	3278.8	1.5	3306	66	399
A-n69-k9+1	3294*	3281.2	2.4	3287.0	3.0	3294	11	9
A-n80-k10+1	5802*	5772.9	3.8	5789.8	4.9	5802	70	199

Table 6
Computational results for B instances with an extra vehicle.

Name	UB	Root CG		Root CG and RCI		BCP		
		Obj.	Time	Obj.	Time	LB	Time	Nodes
B-n31-k5+1	1790*	1790.0	0.1	1790.0	0.1	1790	0	1
B-n34-k5+1	2136*	2135.3	0.3	2135.5	0.3	2136	1	3
B-n35-k5+1	2703*	2689.0	0.1	2702.5	0.3	2703	0	3
B-n38-k6+1	2053*	2052.8	0.2	2053.0	0.2	2053	0	1
B-n39-k5+1	1844*	1841.8	0.4	1843.5	0.5	1844	1	3
B-n41-k6+1	2234*	2204.4	0.1	2219.2	0.3	2234	82	713
B-n43-k6+1	2055*	2030.8	0.4	2037.8	0.5	2055	784	3083
B-n44-k7+1	2254*	2246.8	0.4	2250.1	0.6	2254	6	89
B-n45-k5+1	2197*	2140.9	1.7	2187.2	2.7	2197	15	115
B-n45-k6+1	1884*	1859.0	0.6	1878.6	1.0	1884	5	33
B-n50-k7+1	2135*	2108.4	1.2	2126.3	1.8	2135	14	155
B-n50-k8+1	2901*	2895.9	0.7	2900.0	1.0	2901	3	9
B-n51-k7+1	2895*	2886.3	0.6	2890.8	0.8	2895	3	9
B-n52-k7+1	2461*	2451.2	1.5	2460.5	1.8	2461	6	9
B-n56-k7+1	2316*	2309.3	2.0	2312.7	2.2	2316	11	51
B-n57-k7+1	3689*	3685.4	1.9	3686.6	2.1	3689	5	7
B-n57-k9+1	4473*	4464.8	0.5	4470.4	0.9	4473	22	265
B-n63-k10+1	4305*	4286.7	1.2	4300.4	2.0	4305	14	125
B-n64-k9+1	2444*	2409.0	1.5	2429.6	2.7	2444	446	1239
B-n66-k9+1	3956*	3946.1	4.1	3947.2	4.8	3956	139	771
B-n67-k10+1	2768*	2768.0	1.6	2768.0	1.6	2768	2	1
B-n68-k9+1	4015*	4009.7	1.2	4014.1	2.2	4015	5	5
B-n78-k10+1	3728*	3698.8	2.9	3723.0	6.5	3728	65	153

5.2. Results with an extra route

Tables 5–8 show the results (under the same headings as in Tables 1–4) for these instances. We have appended a '+1' to the instance names in order to emphasize that an extra vehicle has been added.

In addition to obtaining the smaller objective function value as expected, we also observe that the instances seem much easier to solve with an extra vehicle. For the B instances, this may very well be due to the structure of the location of customers. Indeed, in the B instances the customers are located in clusters, and the number of clusters exceed the minimum number of vehicles (see Augerat,

Table 7
Computational results for E and M instances with an extra vehicle.

Name	UB	Root CG		Root CG and RCI		BCP		
		Obj.	Time	Obj.	Time	LB	Time	Nodes
E-n22-k4+1	712*	712.0	0.1	712.0	0.1	712	0	1
E-n23-k3+1	1457*	1449.3	0.6	1453.1	0.8	1457	1	3
E-n30-k3+1	1638*	1619.3	0.4	1636.3	0.5	1638	1	3
E-n33-k4+1	2689*	2687.2	0.8	2689.0	0.9	2689	1	1
E-n51-k5+1	1899*	1888.2	2.0	1889.8	2.3	1899	10	31
E-n76-k7+1	2640*	2617.2	13.2	2619.2	13.7	2640	611	941
E-n76-k8+1	2457*	2428.8	6.3	2428.9	6.4	2457	1343	2023
E-n76-k10+1	2206*	2188.8	2.7	2192.1	3.3	2206	84	439
E-n76-k14+1	1976*	1968.5	1.0	1971.4	1.3	1976	4	17
E-n101-k8+1	3644*	3615.7	64.1	3622.2	69.7	3644	1832	767
E-n101-k14+1	3660	2820.3	6.4	2821.6	7.2	2839	–	4041
M-n101-k10+1	4591	3408.8	11.5	3413.3	12.5	3430	–	5151
M-n121-k7+1	9073	6785.8	393.6	6799.4	427.5	6814	–	773
M-n151-k12+1	6222	4648.2	783.3	4660.9	822.0	4674	–	367
M-n200-k16+1	7685	5638.6	1375.3	5662.0	1474.0	5665	–	17

Table 8
Computational results for P instances with an extra vehicle.

Name	UB	Root CG		Root CG and RCI		BCP		
		Obj.	Time	Obj.	Time	LB	Time	Nodes
P-n16-k8+1	388*	387.5	0.1	388.0	0.1	388	0	3
P-n19-k2+1	620*	618.5	0.0	620.0	0.0	620	0	1
P-n20-k2+1	693*	691.0	0.1	691.0	0.1	693	0	5
P-n21-k2+1	710*	710.0	0.1	710.0	0.1	710	0	1
P-n22-k2+1	763*	762.2	0.1	762.2	0.1	763	0	3
P-n22-k8+1	618*	618.0	0.0	618.0	0.0	618	0	1
P-n23-k8+1	567*	564.5	0.0	566.5	0.0	567	0	3
P-n40-k5+1	1360*	1350.0	0.5	1350.7	0.5	1360	2	37
P-n45-k5+1	1670*	1656.7	0.7	1656.7	0.7	1670	7	37
P-n50-k7+1	1439*	1434.5	0.4	1435.6	0.5	1439	1	9
P-n50-k8+1	1366*	1361.9	0.2	1362.5	0.3	1366	1	15
P-n50-k10+1	1264*	1256.8	0.1	1258.0	0.2	1264	1	21
P-n51-k10+1	1370*	1364.2	0.2	1367.0	0.2	1370	0	5
P-n55-k7+1	1609*	1605.1	0.7	1605.2	0.8	1609	2	7
P-n55-k8+1	1620*	1607.5	0.6	1607.7	0.7	1620	11	103
P-n55-k10+1	1386*	1383.5	0.3	1386.0	0.3	1386	0	1
P-n55-k15+1	1299*	1287.2	0.1	1295.1	0.2	1299	1	25
P-n60-k10+1	1595*	1595.0	0.4	1595.0	0.4	1595	0	1
P-n60-k15+1	1459*	1451.7	0.1	1457.6	0.2	1459	0	7
P-n65-k10+1	1834*	1823.3	0.6	1827.2	0.8	1834	5	43
P-n70-k10+1	1974*	1971.5	0.9	1971.7	0.9	1974	2	5
P-n76-k4+1	3767	3689.4	81.3	3701.4	90.7	3736	–	1435
P-n76-k5+1	4119	3203.2	20.1	3206.4	22.0	3243	–	2319
P-n101-k4+1	7501	5592.0	1695.7	5603.8	1776.7	5604	–	5

Table 9
Computational results for CMT instances.

Name	Other name	Our code		Metaheuristics		
		LB	Time	NPC	RL	KF
CMT1	E-n51-k5	2230.35*	27	2230.35*	2230.35*	2230.35*
CMT2	E-n76-k10	2391.63*	22,511	2426.12	2391.63*	2391.63*
CMT3	E-n101-k8	4045.42*	25,033	4045.42*	4045.42*	4045.42*
CMT4	M-n151-k12	4987.52*	7801	4987.52*	4987.52*	4987.52*
CMT5	M-n200-k16+1	5775.28	28,800	5817.75	5838.32	5809.59
CMT11	M-n121-k7	7197.69	28,800	7317.98	7315.87	7314.55
CMT12	M-n101-k10	3558.92*	841	3558.92*	3558.92*	3558.92*

1995), which necessitates some inter-cluster driving by one or more vehicles if K is fixed at the minimum possible. With an extra vehicle, there will be more flexibility towards having clusters serviced by separate vehicles.

However, also for the A and P instances do we find that the extra vehicle makes the problems more easily solved. Further, most of the E instances are also easily solved with the extra vehicle.

From a practical perspective, this seems to be a very promising result. Indeed, the nature of the objective function does in itself suggest that it will be beneficial for customers to allow for more routes. Our results seem to suggest that under such circumstances with a higher number of routes, many CCVRP instances are solvable with a limited computational effort.

5.3. Results for CMT instances

Finally, we have performed some further computational results to assess the quality of heuristic methods for the CCVRP. In particular, the CMT instances used by Christofides, Mingozzi, and Toth (1979, chap. 11) have been used for testing the metaheuristics for the CCVRP in Ke and Feng (2013), Ngueveu et al. (2010), Ribeiro and Laporte (2012), and we have accordingly made an attempt to solve them by our code. The first three of these instances were actually introduced in Christofides and Eilon (1969), whereas the remaining four instances were introduced in Christofides et al. (1979, chap. 11). In Table 9 we have also given the other name by which the individual instance is known, using our naming convention from Tables 1–8. As noted, all distances are real-valued in these instances.

In six of the seven instances in Table 9, the number of vehicles is fixed at the minimum possible, whereas an extra vehicle is available in `CMT5`. Following our naming convention, for this instance we have used the alternative name `M-n200-k16+1` in order to emphasize that the number of vehicles is actually one more than the minimum possible.

Table 9 shows the results. For each of the instances, the table shows our lower bound and the time spent by our code, respectively. Our LB is marked by a '*' if it is proven to be optimal, otherwise we report the LB obtained after eight hours. For each instance we also report the upper bounds obtained by three metaheuristics, namely those of Ngueveu, Prins and Wolfler Calvo (NPC) (Ngueveu et al., 2010), Ribeiro and Laporte (RL) (Ribeiro & Laporte, 2012), and Ke and Feng (KF) (Ke & Feng, 2013), respectively.

We find it very encouraging that our code was able to find the proven optimal solution to five of the CMT instances, including the instance `M-n151-k12` with as many as 150 customers.

The results in Table 9 also show that both RL and KF found the optimal solution to all five instances that we solved to optimality, whereas NPC reached the optimum to four of the five instances. As such, our results do indeed confirm the high quality of the three metaheuristics.

6. Conclusion and perspectives

In this paper we have proposed an exact algorithm for the CCVRP. Our algorithm is a BCP algorithm, which previously has shown to be an effective type of algorithm for the CVRP. Our computational experiments on well-known CVRP instances show that our algorithm is able to solve many CCVRP instances with a limited computational effort. Our algorithm has shown to be particularly effective in solving the CCVRP instances if we allow for one more than the minimum number of routes.

While our work is concerned with the CCVRP itself, we consider this work on the CCVRP also as a contribution towards being able to solve a wider spectrum of vehicle routing problems, taking into account issues of sustainability in transportation.

In particular, fuel consumption and CO2 emission do not depend solely on distance but are also influenced by factors such as type of vehicle and engine, speed, street surface, and load of the vehicle (we refer to Sbihi & Eglese (2006) for a survey).

The cost structure in the CCVRP may actually be viewed as one where the cost per distance unit along an arc is proportional to the weight of the vehicle when traversing the arc, provided that the number of customers on board the vehicle is used as the measure of weight of the vehicle.

More generally, a vehicle's fuel consumption might be modeled as a product of distance travelled and vehicle weight (see, e.g., Kara, Kara, & Yetiş, 2007), but where the vehicle weight includes both the weight of the empty vehicle and the load on board the vehicle when traversing the individual arc. We consider our model and algorithm for the CCVRP to be a step in the direction towards being able to handle such more general structures.

Acknowledgement

The authors thank two anonymous referees for their helpful comments which led to an improved paper.

References

Applegate, D., Cook, W., Dash, S., & Rohe, A. (2002). Solution of a min–max vehicle routing problem. *INFORMS Journal of Computing, 14*(2), 132–143.

Augerat, P. (1995). Approche polyédrale du Problème de Tournées de Véhicules. PhD thesis, Laboratoire ARTEMIS-IMAG, Institut National Polytechnique de Grenoble, France.

Bowerman, R., Hall, B., & Calamai, P. (1995). A multi-objective optimization approach to urban school bus routing: Formulation and solution method. *Transportation Research Part A, 29A*(2), 107–123.

Campbell, A. M., Vandenbussche, D., & Hermann, W. (2008). Routing for relief efforts. *Transportation Science, 42*(2), 127–145.

Chakrabarty, D., & Swamy, C. (2011). Facility location with client latencies: Linear programming based techniques for minimum latency problems. *Lecture Nodes in Computer Science, 6655*, 92–103.

Chaudhuri, K., Godfrey, B., Rao, S., & Talwar, K. (2003). Paths, trees, and minimum latency tours. In *Proceedings of the 44th annual IEEE symposium on the foundations of computer science* (pp. 36–45).

Chen, P., Dong, X., & Niu, Y. (2012). An iterated local search algorithm for the cumulative capacitated vehicle routing problem. In Honghua Tan (Ed.), *Technology for education and learning. Advances in intelligent and soft computing* (Vol. 136, pp. 575–581). Berlin/Heidelberg: Springer.

Christofides, N., & Eilon, S. (1969). An algorithm for the vehicle-dispatching problem. *Operations Research Quarterly, 20*(3), 309–318.

Christofides, N., Mingozzi, A., & Toth, P. (1979). The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, & C. Sandi (Eds.), *Combinatorial optimization*. Wiley & Sons.

Christofides, N., Mingozzi, A., & Toth, P. (1981). Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming, 20*(1), 255–282.

Corberán, Á., Fernández, E., Laguna, M., & Martí, R. (2002). Heuristic solutions to the problem of routing school buses with multiple objectives. *Journal of the Operational Research Society, 53*(4), 427–435.

Desaulniers, G., Desrosiers, J., & Solomon, M. M. (Eds.). (2005). *Column generation*. New York, NY, USA: Springer.

Desrosiers, J., Dumas, Y., Solomon, M. M., & Soumis, F. (1995). Time constrained routing and scheduling. In M. O. Ball, T. L. Magnanti, C. L. Monma, & G. L. Nemhauser (Eds.), *Network routing*. Elsevier.

Dewilde, T., Cattrysse, D., Coene, S., Spieksma, F. C. R., & Vansteenwegen, P. (2013). Heuristics for the traveling repairman problem with profits. *Computers & Operations Research, 40*(7), 1700–1707.

Fakcharoenphol, J., Harrelson, C., & Rao, S. (2007). The k-traveling repairman problem. *ACM Transactions on Algorithms, 3*(4).

Fukasawa, R., Longo, H., Lysgaard, J., Poggi de Aragão, M., Reis, M., Uchoa, E., et al. (2006). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming, 106*(3), 491–511.

Golden, B. L., Laporte, G., & Taillard, E. D. (1997). An adaptive memory heuristic for a class of vehicle routing problems with minmax objective. *Computers & Operations Research, 24*(5), 445–452.

Hartl, R. F., Hasle, G., & Janssens, G. K. (2006). Special issue on rich vehicle routing problems, editorial. *Central European Journal of Operations Research, 14*(2), 103–104.

Irnich, S., & Desaulniers, G. (2005). Shortest path problems with resource constraints. In Desaulniers et al. (2005).

Jothi, R., & Raghavachari, B. (2007). Approximating the k-traveling repairman problem with repairtimes. *Journal of Discrete Algorithms, 5*(2), 293–303.

Kara, İ., Kara, B. Y., & Yetiş, M. K. (2008). Cumulative vehicle routing problems. In T. Caric, & H. Gold (Eds.), *Vehicle routing problem*. InTech.

Kara, İ., Kara, B. Y., & Yetiş, M. K. (2007). Energy minimizing vehicle routing problem. In A. Dress, Y. Xu, & B. Zhu (Eds.), *Combinatorial optimization and applications. LNCS* (Vol. 4616, pp. 62–71).

Ke, L., & Feng, Z. (2013). A two-phase metaheuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research, 40*(2), 633–638.

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science, 43*(4), 408–416.

Letchford, A. N., Lysgaard, J., & Eglese, R. W. (2007). A branch-and-cut algorithm for the capacitated open vehicle routing problem. *Journal of the Operational Research Society, 58*(12), 1642–1651.

Levin, A., & Penn, M. (2008). Approximation algorithm for minimizing total latency in machine scheduling with deliveries. *Discrete Optimization, 5*(1), 97–107.

Li, C.-L., Vairaktarakis, G., & Lee, C.-Y. (2005). Machine scheduling with deliveries to multiple customer locations. *European Journal of Operational Research, 164*(1), 39–51.

Lysgaard, J. (2003). CVRPSEP: A package of separation routines for the capacitated vehicle routing problem. Working Paper 03-04, Department of Management Science and Logistics, Aarhus School of Business. <www.hha.dk/ ~ lys>.

Lysgaard, J., Letchford, A. N., & Eglese, R. W. (2004). A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming, 100*(2), 423–445.

Ngueveu, S. U., Prins, C., & Wolfler Calvo, R. (2010). An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Computers & Operations Research, 37*(11), 1877–1885.

Ribeiro, G. M., & Laporte, G. (2012). An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research, 39*(3), 728–735.

Sbihi, A., & Eglese, R. W. (2006). The relationship between vehicle routing and scheduling and green logistics – A literature survey. Management science working paper series, Lancaster University.

Silva, M. M., Subramanian, A., Vidal, T., & Ochi, L. S. (2012). A simple and effective metaheuristic for the minimum latency problem. *European Journal of Operational Research, 221*(3), 513–520.

Wu, B. Y., Huang, Z.-N., & Zhan, F.-J. (2004). Exact algorithm for the minimum latency problem. *Information Processing Letters, 92*(6), 303–309.