A Branch-Cut-and-Price Algorithm for the Capacitated Arc Routing Problem

Rafael Martinelli[1], Diego Pecin[1], Marcus Poggi[1], and Humberto Longo[2]

[1] PUC-Rio – Departamento de Informática
Rua Marquês de São Vicente, 225 RDC – Rio de Janeiro, RJ 22453-900, Brazil
{rmartinelli,dpecin,poggi}@inf.puc-rio.br
[2] UFG – Instituto de Informática
Campus Samambaia, Goiânia, GO 74001-970, Brazil
longo@inf.ufg.br

Abstract. Arc routing problems are among the most challenging combinatorial optimization problems. We tackle the Capacitated Arc Routing Problem where demands are spread over a subset of the edges of a given graph, called the required edge set. Costs for traversing edges, demands on the required ones and the capacity of the available identical vehicles at a vertex depot are given. Routes that collect all the demands at minimum cost are sought. In this work, we devise a Branch-Cut-and-Price algorithm for the Capacitated Arc Routing problem using a column generation which generates non-elementary routes (usually called *q-routes*) and exact separation of odd edge cutset and capacity cuts. Computational experiments report one new optimal and twelve new lower bounds.

Keywords: Arc Routing, Branch-Cut-and-Price, Integer Programming.

1 Introduction

The Capacitated Arc Routing Problem (CARP) is a problem which has several applications in the real life. One can think of this problem as a garbage collection problem. Suppose a company is responsible for collecting the garbage of a neighborhood. This company has a set of identical vehicles available in its base and knows a priori the amount of garbage generated on each street. As a company, its objective is to minimize the operational costs. So, the company needs to define the route for each vehicle which gives the lowest distance traveled. These routes must begin and end at the company's base, none of the vehicles can have its capacity violated, the garbage of a given street may be collected only by a single vehicle and all the garbage must be collected. Other possible applications for this problem are street sweeping, winter gritting, electric meter reading and airline scheduling [35].

This problem is strongly NP-Hard as shown by Golden and Wong in 1981 [16], where it was also proposed. Since then, several works were made with different approaches. We can cite some specific primal heuristics like Augment-Merge [16,15], Path-Scanning [15], Parallel-Insert [8], Construct-Strike [32] and Augment-Insert

P.M. Pardalos and S. Rebennack (Eds.): SEA 2011, LNCS 6630, pp. 315–326, 2011.

[33]. There are several heuristics for obtaining lower bounds, like Matching [16], Node Scanning [3], Matching-Node Scanning [31], Node Duplication [19], Hierarchical Relaxations [2] and Multiple Cuts Node Duplication [36].

For the last years, most of the approximate results for the CARP were found using metaheuristics. All kinds of metaheuristics have already been proposed for the CARP, some of them are Simulated Annealing [12], the CARPET Tabu Search [18], Genetic Algorithm [22], Memetic Algorithm [23], Ant Colony [10], Guided Local Search [6] and Deterministic Tabu Search [7].

As being a hard problem, it is very difficult to solve the CARP to optimality. The works which tried to achieve this usually use integer programming. The first integer programming formulation was proposed by Golden and Wong [16] and since then some other formulations were proposed by Belenguer and Benavent [4] and Letchford [26], who also proposed valid inequalities for the problem. These integer formulations were solved using techniques such Branch-and-Bound [20], Cutting Plane [5,1], Column Generation [17,25] and Branch-and-Cut [24].

The objective of this work is to devise a Branch-Cut-and-Price algorithm for the CARP using a column generation algorithm with columns associated to non-elementary routes (also called *q-routes*) where 2-cycles are eliminated and exactly separating the odd edge cutset and capacity cuts. As far as we know, no Branch-Cut-and-Price algorithm which generates routes on the original graph was tailored for this problem. However, a Branch-Cut-and-Price was applied to the CARP by a transformation from the Capacitated Vehicle Routing Problem (CVRP) [29].

This work is divided in six sections. On section 2, we present mathematical formulations for the problem. On section 3, we detail the column generation with the non-elementary route pricing. On section 4, the Branch-Cut-and-Price is devised. On section 5, we show the computational experiments. On section 6, we conclude our work and point out future directions.

2 Mathematical Formulations

The CARP can be defined as follows. Consider a connected undirected graph $G = (V, E)$, with vertex set V and edge set E, costs $c : E \to \mathbb{Z}_0^+$, demands $d : E \to \mathbb{Z}_0^+$, a set I containing k identical vehicles with capacity Q and a distinguished depot vertex labeled 0. Define $E_R = \{e \in E | d_e > 0\}$ as the set of required edges. Let F be a set of closed routes which start and end at the depot, where the edges in a route can be either *serviced* or *deadheaded* (when the vehicle traverses the edge without servicing it). The set F is a feasible CARP solution if:

- Each required edge is serviced by exactly one route in F and
- The sum of demands of the serviced edges in each route F does not exceed the vehicle capacity.

We want to find a solution minimizing the sum of the costs of the routes. It corresponds to minimize the sum of the deadheaded edges' cost in the routes.

2.1 Two-Index Formulation

The most intuitive formulation for the CARP is to create a binary variable, x_e^k, for each required edge and each vehicle, an integer variable, z_e^k, for each deadheaded edge and each vehicle and use them in flow constraints. This formulation, known as two-index formulation [25], can be written as follows.

$$\text{MIN} \sum_{p \in I} \left(\sum_{e \in E_R} c_e x_e^p + \sum_{e \in E} c_e z_e^p \right) \tag{1}$$

$$\text{s.t.} \quad \sum_{p \in I} x_e^p = 1 \qquad \forall e \in E_R \tag{2}$$

$$\sum_{e \in E_R} d_e x_e^p \leq Q \qquad \forall p \in I \tag{3}$$

$$\sum_{e \in \delta_R(S)} x_e^p + \sum_{e \in \delta(S)} z_e^p \geq 2x_f^p \qquad \forall S \subseteq V \backslash \{0\}, f \in E_R(S), p \in I \tag{4}$$

$$\sum_{e \in \delta_R(S)} x_e^p + \sum_{e \in \delta(S)} z_e^p \equiv 0 \bmod (2) \; \forall S \subseteq V \backslash \{0\}, p \in I \tag{5}$$

$$x_e^p \in \{0, 1\} \qquad \forall e \in E_R, p \in I \tag{6}$$

$$z_e^p \in \mathbb{Z}_0^+ \qquad \forall e \in E, p \in I \; . \tag{7}$$

The objective function (1) minimizes the cost of each edge, being serviced or deadheaded. Constraints (2) assure that all required edges are serviced. Constraints (3) limit the total demand serviced by each vehicle to the capacity Q. Given S a vertex set, $E_R(S) = \{(i,j) \in E_R | i \in S, j \in S\}$, $\delta(S) = \{(i,j) \in E | i \in S, j \notin S\}$ and $\delta_R(S) = \{(i,j) \in E_R | i \in S, j \notin S\}$, constraints (4) assure that every route is connected and constraints (5) force every route in the solution to induce an Eulerian graph.

This formulation can be used to generate complete solutions for the CARP, but there is an exponential number of constraints (5). The separation of these constraints turns this formulation prohibitive when one is trying to solve large instances of the problem.

2.2 One-Index Formulation

A relaxation for the CARP is given by the one-index formulation, defined in [5]. This formulation considers only the deadheaded edges and aggregates every vehicle in just one variable. So, there is an integer variable z_e, which indicates the number of times an edge e is deadheaded by *any* vehicle. This formulation uses valid inequalities as constraints, since none of the two-index formulation's constraints can be used with these variables.

Valid inequalities (cuts) are added to formulations in order to improve the solution. Two families of cuts are widely used in almost any formulation for the CARP. The first one was created knowing that it is easy to show that every cutset S must have an even degree on any feasible solution. Because of that, for any cutset S containing an odd number of required edges, i.e. odd $|\delta_R(S)|$, at

least one edge $e \in \delta(S)$ will be deadheaded. These are called *Odd Edge Cutset constraints*:

$$\sum_{e \in \delta(S)} z_e \geq 1 \quad \forall S \subseteq V \backslash \{0\}, |\delta_R(S)| \text{ odd} . \tag{8}$$

The second family of cuts comes from the knowledge of a lower bound for the number of vehicles needed to service the required edges of a cutset, called $k(S)$. This lower bound can be obtained solving a Bin Packing for each cutset S, but since this problem is NP-Hard [14], it is better to obtain an approximation for this value. A very good approximation is obtained dividing the sum of demands of each required edge $e \in E_R(S) \cup \delta_R(S)$ by the vehicle capacity, as shown in equation (9).

$$k(S) = \left\lceil \frac{\sum_{e \in E_R(S) \cup \delta_R(S)} d_e}{Q} \right\rceil . \tag{9}$$

Knowing that at least $2k(S)$ vehicles must cross any cutset S, we can conclude that at least $2k(S) - |\delta_R(S)|$ edges from the cutset will be deadheaded in any feasible solution. These are the *Capacity constraints*:

$$\sum_{e \in \delta(S)} z_e \geq 2k(S) - |\delta_R(S)| \quad \forall S \subseteq V \backslash \{0\} . \tag{10}$$

As the left hand side of both (8) and (10) constraints are the same, we can put them together in just one constraint with the right hand side shown in (11).

$$\alpha(S) = \begin{cases} \max\{2k(S) - |\delta_R(S)|, 1\} \text{ if } |\delta_R(S)| \text{ is odd,} \\ \max\{2k(S) - |\delta_R(S)|, 0\} \text{ if } |\delta_R(S)| \text{ is even.} \end{cases} . \tag{11}$$

With everything defined, we can state the one-index formulation:

$$\text{MIN} \sum_{e \in E} c_e z_e \tag{12}$$

$$\text{s.t.} \sum_{e \in \delta(S)} z_e \geq \alpha(S) \; \forall S \subseteq V \backslash \{0\} \tag{13}$$

$$z_e \in \mathbb{Z}_0^+ \qquad \forall e \in E . \tag{14}$$

The objective function (12) minimizes the deadheadeds cost. In order to obtain the total cost, one must sum this value with the required edges costs. Constraints (13) are the odd edge cutset and capacity constraints together.

This formulation is a relaxation for the CARP because it does not give a complete solution for the problem and sometimes the solution found does not correspond to any feasible solution. But in practice, it gives very good lower bounds. The difficulty which arises here is how to generate the cuts (8) and (10). We will discuss this later.

2.3 Set Partitioning Approach

Another way to formulate the problem is, given a set of every possible route Ω, create a binary variable λ_r for every route $r \in \Omega$ and use them within a set partitioning formulation. Let the binary constant a_r^e be 1 if route r services the required edge e, 0 otherwise, and the integer constant b_r^e be the number of times edge e is deadheaded in route r. The set partitioning formulation for the CARP is as follows.

$$\text{MIN} \sum_{r \in \Omega} c_r \lambda_r \tag{15}$$

$$\text{s.t.} \sum_{r \in \Omega} \lambda_r = k \tag{16}$$

$$\sum_{r \in \Omega} a_r^e \lambda_r = 1 \; \forall e \in E_R \tag{17}$$

$$\lambda_r \in \{0, 1\} \quad \forall r \in \Omega \; . \tag{18}$$

The objective function (15) minimizes the total cost of the used routes. Constraints (16) limit the number of routes used to the number of available vehicles and constraints (17) assure each required edge is serviced by only one route.

This formulation is obtained doing a Dantzig-Wolfe decomposition on constraints (3), (4) and (5) from the two-index formulation. The decomposition of these constraints defines the λ_r variables as routes. This is the reason why there is no vehicle index on the variables. Remark that this decomposition does not enforces the routes to be elementary.

Knowing how many times a route r traverses an edge e as deadheaded, we can create a direct mapping between the λ_r variables and the z_e variables from the one-index formulation, as shown in (19). This is enough to use the one-index constraints (13) to improve this formulation.

$$\sum_{r \in \Omega} b_r^e \lambda_r = z_e \quad \forall e \in E \; . \tag{19}$$

As the former two formulations, this one also has a problem. The number of possible routes is exponentially large, making it hard to generate all of them. To tackle this difficulty, we use a column generation algorithm.

3 Column Generation

Column generation is a technique for solving a linear program which has a prohibitive number of variables (columns). The algorithm starts with a small set of columns and solves the linear program (called here restricted master) to optimality. After that, the algorithm 'prices' the columns trying to find one with a reduced cost suitable to improve the solution. It then repeats the whole operation until no improving column is found.

In order to generate columns for the set partitioning formulation, first we have to define the reduced cost of a route. Given the dual variables γ, β_e and π_S, associated with constraints (16), (17) and (13), the reduced cost of a route r is:

$$\tilde{c}_r = c_r - \gamma - \sum_{e \in E_R} a_r^e \beta_e - \sum_{S \subseteq V \setminus \{0\}} \sum_{e \in \delta(S)} b_r^e \pi_S \tag{20}$$

$$= -\gamma + \sum_{e \in E_R} a_r^e \left(c_e - \beta_e\right) + \sum_{e \in E} b_r^e \left(c_e - \sum_{S \subseteq V \setminus \{0\} : e \in \delta(S)} \pi_S \right) . \tag{21}$$

The objective of the pricing subproblem is to find a route r (not necessarily elementary) with the smallest reduced cost $\tilde{c_r}$. This corresponds to solve a *Shortest Path Problem with Resource Constraints* (SPPRC), problem which can be solved in pseudopolynomial time with dynamic programming [9].

Our basic algorithm starts with a dynamic programming matrix $T(e, c, v)$, which indicates the reduced cost of the path from the depot to required edge e, with capacity c, ending at vertex v. It means that, for each edge e and capacity c, we have to store the value for both endpoints of e. Let $\tilde{c}_e = c_e - \beta_e$ and $\tilde{g}_e = c_e - \sum_{S \subseteq V \setminus \{0\} : e \in \delta(S)} \pi_S$ be the reduced costs associated with the required and deadheaded edges, respectively, $e = (w, v)$ and $f = (i, j)$. We initialize every cell with $+\infty$ and use the following recurrence to fill the dynamic programming matrix, which gives a complexity of $O(|E_R|^2 Q)$:

$$\begin{cases} T(e, c, v) = \min_{f \in E_R} \{T(f, c - d_e, i) + \tilde{c}_e + \tilde{g}_{iw}, T(f, c - d_e, j) + \tilde{c}_e + \tilde{g}_{jw}\} \\ T(e, d_e, v) = \tilde{c}_e + \tilde{g}_{0w} - \gamma \end{cases} . \tag{22}$$

In order to improve the lower bounds, we use a cycle elimination scheme, which forbids repeating edges within a given size. A 2-cycle elimination forbids cycles of size one $(e - e)$ and size two $(e - f - e)$. To do 1-cycle elimination, the algorithm avoids updating a required edge from the same edge. For 2-cycle elimination, it stores the two bests reduced costs from different edges in each cell of the dynamic programming matrix. To eliminate cycles greater than 2 is more complicated and was not used in this work. A complete description of k-cycle elimination can be found in [21].

Ideally, one would like to consider only elementary routes. With this modification the pricing subproblem turns into an *Elementary Shortest Path Problem with Resource Constraints* (ESPPRC), where the resource is the vehicle's capacity Q. This problem is strongly NP-Hard [11], but one can eventually solve it exactly when the graph is sparse [25].

4 Branch-Cut-and-Price

Since the column generation algorithm solves the linear relaxation of the set partitioning formulation, another technique is needed in order to obtain an integer

solution. In this work, we use Branch-and-Bound, a widely known technique, which enumerates all possible solutions through a search tree, discarding the branches with solutions greater than a known upper bound for the instance. This technique, when used together with column generation and cut separation is called *Branch-Cut-and-Price* (BCP).

The BCP algorithm starts on the root node of the search tree. The column generation algorithm is executed and a lower bound for the solution is found. If the solution is not integral, it applies a cut separation routine and re-execute the column generation algorithm. When no cuts are found, it branches. A variable with a continuous value is chosen and two branches are created, following a defined branching rule, and the nodes of these branches are put in a queue. There are some policies which can be used to choose the next node to explore. In our BCP algorithm, we put the nodes in a heap and always choose the one with the lowest value (lower bound). The whole procedure is then repeated for each node. The algorithm stops when the difference between the lower bound and the best integer solution found (upper bound) is less than one.

4.1 Branching Rule

The most intuitive branching rule is, given variable λ_r from the set partitioning formulation with a continuous solution $\bar{\lambda_r} \in [0, 1]$, to create two branches, the first one with $\lambda_r = 0$ and other with $\lambda_r = 1$. But this cannot be done due to the column generation algorithm. If a route r is fixed to zero, the pricing subproblem will find this route again on the next iteration and will return it to the restricted master.

There are some ways to cope with this difficulty. But instead of that, we prefer to branch on the deadheaded edges variables. When we need to branch, we obtain the values of the z_e variables using equation (19), search for the variable whose value $\bar{z}_e$ is closer to 0.5 and then create two branches, one with $z_e \leq \lfloor \bar{z}_e \rfloor$ and other with $z_e \geq \lceil \bar{z}_e \rceil$. Mapping these to λ_r variables, we get:

$$\sum_{r \in \Omega} b_r^e \lambda_r \leq \lfloor \bar{z}_e \rfloor \tag{23}$$

$$\sum_{r \in \Omega} b_r^e \lambda_r \geq \lceil \bar{z}_e \rceil \ . \tag{24}$$

These inequalities generate new bounds constraints on the restricted master problem:

$$lb_e \leq \sum_{r \in \Omega} b_r^e \lambda_r \leq ub_e \quad \forall e \in E \ . \tag{25}$$

Note that when an integer solution is found, it is integer just on z_e variables and it may not be integer on λ_r. For this reason, this integer solution may be lower than the optimal solution and may also be infeasible – it behaves like the

one-index formulation. Nevertheless, these integer solutions give very good lower bounds for the problem [5].

4.2 Pricing

With the introduction of the bounds constraints, the pricing subproblem does not change, but the reduced cost of a route (21) must consider the dual values associated with these constraints. Given ρ_e, the dual variable associated with constraints (25), the new equation for the reduced cost of a route is:

$$\tilde{c}_r = -\gamma + \sum_{e \in E_R} a_r^e \left(c_e - \beta_e\right) + \sum_{e \in E} b_r^e \left(c_e - \rho_e - \sum_{S \subseteq V \setminus \{0\} : e \in \delta(S)} \pi_S \right) . \quad (26)$$

4.3 Strong Branching

In order to obtain better lower bounds faster, the BCP algorithm does strong branching. When a branching variable needs to be chosen, it selects n candidates to branch (here $n = 3$), usually the first n closest to 0.5. Then, it runs the column generation algorithm for both branches of every candidate and, given $left_c$ and $right_c$, the values of left and right branches of candidate c, it chooses the one with largest $min\{left_c, right_c\}$. In the case of a tie, it chooses the one with largest $max\{left_c, right_c\}$. If it finds a candidate with at least one branch infeasible, this one is chosen immediately.

4.4 Cut Generation

Our BCP algorithm pre-generates cuts before starting to solve the root node of the search tree. We use the one-index formulation and separate (8) and (10) cuts using the algorithms described in [30] and [1] respectively. The first one runs several maximum flows, which can be done in polynomial time. The second one is more difficult and is solved using mixed integer programming. This cut generation runs iteratively until no new cut is found. After that, we gather the cuts and start to solve the root node of the BCP.

During the BCP algorithm, for each node open on the search tree, we run the separation algorithm for the (8) cuts. Only when an integer solution is found, we run the separation algorithm for the (10) cuts. It would be prohibitively costly to run this separation on every node of the BCP.

5 Computational Experiments

All algorithms were implemented in C++, using Windows Vista Business 32-bits, Visual C++ 2008 Express Edition and IBM Cplex 12.2. Tests were conducted on an Intel Core 2 Duo 2.8 GHz, using just one core, with 4GB RAM. We applied

Table 1. Results for eglese instances

							Cut Generation			Branch-Cut-and-Price				
Name	$\|V\|$	$\|E_R\|$	$\|E\|$	k	UB	LB	Cost	Cuts	Time	Root	Cost	Cuts	Nodes	Time
e1-a	77	51	98	5	3548	3548	3527	55	44	3543	**3548**	3	2	151
e1-b	77	51	98	7	4498	4498	4464	78	51	4465	4487	9	70	436
e1-c	77	51	98	10	5595	5566	5513	70	52	5528	5537	0	49	158
e2-a	77	72	98	7	5018	5018	4995	79	34	5002	5012	73	49	1167
e2-b	77	72	98	10	6317	6305	6271	78	38	6280	6291	31	143	996
e2-c	77	72	98	14	8335	8243	8161	80	49	8228	**8274**	58	4179	21600
e3-a	77	87	98	8	5898	5898	5894	73	31	5895	**5898**	1	2	746
e3-b	77	87	98	12	7775	7704	7649	75	45	7687	**7715**	116	1755	18986
e3-c	77	87	98	17	10292	10163	10125	84	51	10176	**10207**	91	3505	21600
e4-a	77	98	98	9	6444	6408	6378	71	22	6389	6395	120	63	3025
e4-b	77	98	98	14	8962	8884	8838	70	37	8874	**8893**	223	1947	21600
e4-c	77	98	98	19	11550	11427	11376	76	50	11439	**11471**	133	2719	21600
s1-a	140	75	190	7	5018	5018	5010	143	331	5013	5014	6	3	318
s1-b	140	75	190	10	6388	6384	6368	154	451	6376	**6388**	62	64	3557
s1-c	140	75	190	14	8518	8493	8404	153	806	8457	**8494**	134	1157	21600
s2-a	140	147	190	14	9884	9824	9737	131	449	9796	9807	537	147	21600
s2-b	140	147	190	20	13100	12968	12901	145	718	12950	**12970**	333	387	21600
s2-c	140	147	190	27	16425	16353	16248	154	1683	16331	**16357**	327	743	21600
s3-a	140	159	190	15	10220	10143	10083	129	303	10131	**10146**	344	157	21600
s3-b	140	159	190	22	13682	13616	13568	127	574	13608	**13623**	372	319	21600
s3-c	140	159	190	29	17194	17100	17007	135	1931	17088	**17115**	420	615	21600
s4-a	140	190	190	19	12268	12143	12026	121	343	12121	12140	270	159	21600
s4-b	140	190	190	27	16283	16093	15984	135	604	16049	16082	498	294	21600
s4-c	140	190	190	35	20517	20375	20236	140	1165	20362	**20380**	252	581	21600
mean					9738.7	9673.8	9615.1	106.5	410.9	9657.8	9676.8	183.9	795.2	13830.8
opt					—	5	0	—	—	0	3	—	—	—
gap					—	0.564%	1.152%	—	—	0.767%	0.561%	—	—	—

our algorithms to the instances of the dataset *eglese*, which was originally used in [27] and [28]. These instances were constructed using as underlying graph regions of the road network of the county of Lancashire (UK). They used cost and demands proportional to the length of the edges and most of the instances have non-required edges. From the classic sets of instances (*kshs, gdb, val* and *eglese*), this one is the only one which still has open instances.

Results are shown in table 1. Column *UB* lists the best known upper bounds, reported by Santos et al. [34] and Fu et al. [13]. Column *LB* lists the best known lower bounds, reported by Longo et al. [29] and Brandão and Eglese [7]. This latter work is a primal work and reported some lower bounds which we could not find in any paper, even in the ones referred by them. These lower bounds have been widely used, with rare exceptions, for instance in Santos et al. [34]. The next 3 columns show results for the cut generation done before solving the root node. Columns *Cost*, *Cuts* and *Time* show the solution cost, the number of cuts sent to BCP and the solution time in seconds. The next 5 columns show results for the BCP algorithm. Columns *Root*, *Cost*, *Cuts*, *Nodes* and *Time* show the solution cost at the root node, the BCP solution cost, the number of cuts generated, the nodes opened and the total time in seconds.

In order to run our BCP algorithm for all instances, we set the time limit of 6 hours (21600 seconds). At this point, the algorithm is stopped and the solution reported is the best found so far. All continuous values, costs or times, were rounded up to the next integer.

The improving lower bounds found are shown in bold font and optimal values are underlined. The BCP algorithm found 3 optimal values, *e1-a* and *e3-a* were already known and *s1-b* is a new optimal value.

6 Conclusions and Future Research

We have created the first Branch-Cut-and-Price algorithm for the Capacitated Arc Routing Problem which generates routes on the original graph. The algorithm could find one optimal value and twelve new lower bounds for the eglese instances in reasonable time.

As future research, there are some techniques that may be further explored. We can cite some examples. A column generation algorithm which generates only elementary routes can be tested with the Branch-Cut-and-Price. An enumeration routine which generates all routes between the lower and upper bound can be used to find the optimum on some instances. Besides that, some techniques may be used to speed up the Branch-Cut-and-Price algorithm, like active reduced cost fixing.

Acknowledgements. The contribution by Rafael Martinelli, Diego Pecin and Marcus Poggi de Aragão has been partially supported by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), processes numbers 140849 / 2008-4, 141538 / 2010-4 and 309337 / 2009-7.

References

1. Ahr, D.: Contributions to Multiple Postmen Problems. Ph.D. dissertation, Department of Computer Science, Heidelberg University (2004)
2. Amberg, A., Voß, S.: A Hierarchical Relaxations Lower Bound for the Capacitated Arc Routing Problem. In: Proceedings of the 35th Anual Hawaii International Conference on System Sciences (2002)
3. Assad, A.A., Pearn, W.L., Golden, B.L.: The Capacitated Chinese Postman Problem: Lower Bounds and Solvable Cases. American Journal of Mathematical Management Sciences 7(1,2), 63–88 (1987)
4. Belenguer, J.M., Benavent, E.: Polyhedral Results on the Capacitated Arc Routing Problem, Departamento de Estadística e Investigación Operativa, Universitat de València, Tech. Rep. TR 1-92 (1992)
5. Belenguer, J.M., Benavent, E.: A Cutting Plane Algorithm for the Capacitated Arc Routing Problem. Computers & Operations Research 30, 705–728 (2003)
6. Beullens, P., Muyldermans, L., Cattrysse, D., Oudheusden, D.V.: A Guided Local Search Heuristic for the Capacitated Arc Routing Problem. European Journal of Operational Research 147(3), 629–643 (2003)
7. Brandão, J., Eglese, R.: A Deterministic Tabu Search Algorithm for the Capacitated Arc Routing Problem. Computers & Operations Research 35, 1112–1126 (2008)
8. Chapleau, L., Ferland, J.A., Lapalme, G., Rousseau, J.M.: A Parallel Insert Method for the Capacitated Arc Routing Problem. Operations Research Letters 3(2), 95–99 (1984)
9. Christofides, N., Mingozzi, A., Toth, P.: Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations. Mathematical Programming 20, 255–282 (1981)
10. Doerner, K., Hartl, R., Maniezzo, V., Reimann, M.: An Ant System Metaheuristic for the Capacitated Arc Routing Problem. In: Proceedings of the 5th Metaheuristics International Conference, Tokyo, Japan (2003)
11. Dror, M.: Note on the Complexity of the Shortest Path Models for Column Generation in VRPTW. Operations Research 42(5), 977–978 (1994)
12. Eglese, R.W.: Routeing Winter Gritting Vehicles. Discrete Applied Mathematics 48(3), 231–244 (1994)
13. Fu, H., Mei, Y., Tang, K., Zhu, Y.: Memetic algorithm with heuristic candidate list strategy for capacitated arc routing problem. In: IEEE Congress on Evolutionary Computation, pp. 1–8 (2010)
14. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1979)
15. Golden, B.L., DeArmon, J.S., Baker, E.K.: Computational Experiments with Algorithms for a Class of Routing Problems. Computers and Operations Research 10(1), 47–59 (1983)
16. Golden, B.L., Wong, R.T.: Capacitated Arc Routing Problems. Networks 11, 305–315 (1981)
17. Gómez-Cabrero, D., Belenguer, J.M., Benavent, E.: Cutting Plane and Column Generation for the Capacitated Arc Routing Problem. In: ORP3, Valencia (2005)
18. Hertz, A., Laporte, G., Mittaz, M.: A Tabu Search Heuristic for the Capacitated Arc Routing Problem. Operations Research 48(1), 129–135 (2000)
19. Hirabayashi, R., Nishida, N., Saruwatari, Y.: Node Dublication Lower Bounds for the Capacitated Arc Routing Problems. Journal of the Operations Research Society of Japan 35(2), 119–133 (1992)

20. Hirabayashi, R., Nishida, N., Saruwatari, Y.: Tour Construction Algorithm for the Capacitated Arc Routing Problem. AsiaPacific Journal of Operational Research 9, 155–175 (1992)
21. Irnich, S., Villeneuve, D.: The Shortest-Path Problem with Resource Constraints and k-Cycle Elimination for k $\geq$ 3. INFORMS Journal on Computing 18(3), 391–406 (2006)
22. Lacomme, P., Prins, C., Ramdane-Chérif, W.: A genetic algorithm for the capacitated arc routing problem and its extensions. In: Boers, E.J.W., Gottlieb, J., Lanzi, P.L., Smith, R.E., Cagnoni, S., Hart, E., Raidl, G.R., Tijink, H. (eds.) EvoIASP 2001, EvoWorkshops 2001, EvoFlight 2001, EvoSTIM 2001, EvoCOP 2001, and EvoLearn 2001. LNCS, vol. 2037, pp. 473–483. Springer, Heidelberg (2001)
23. Lacomme, P., Prins, C., Ramdane-Chérif, W.: Competitive memetic algorithms for arc routing problems. Annals of Operations Research 131, 159–185 (2004)
24. Laganá, D., Ghiani, G., Musmanno, R., Laporte, G.: A Branch-and-Cut Algorithm for the Undirected Capacitated Arc Routing Problem. The Journal of the Operations Research Society of America, 1–21 (2007)
25. Letchford, A., Oukil, A.: Exploiting Sparsity in Pricing Routines for the Capacitated Arc Routing Problem. Computers & Operations Research 36, 2320–2327 (2009)
26. Letchford, A.N.: Polyhedral Results for some Constrained Arc-Routing Problems, Ph.D. dissertation, Lancaster University (1996)
27. Li, L.Y.O.: Vehicle Routeing for Winter Gritting, Ph.D. dissertation, Department of Management Science, Lancaster University (1992)
28. Li, L.Y.O., Eglese, R.W.: An Interactive Algorithm for Vehicle Routeing for Winter-Gritting. Journal of the Operational Research Society 47, 217–228 (1996)
29. Longo, H., Poggi de Aragão, M., Uchoa, E.: Solving Capacitated Arc Routing Problems Using a Transformation to the CVRP. Computers & Operations Research 33, 1823–1827 (2006)
30. Padberg, M.W., Rao, M.R.: Odd minimum cut-sets and b-matchings. Mathematics of Operations Research 7, 67–80 (1982)
31. Pearn, W.L.: New Lower Bounds for the Capacitated Arc Routing Problem. Networks 18, 181–191 (1988)
32. Pearn, W.L.: Approximate Solutions for the Capacitated Arc Routing Problem. Computers & Operations Research 16(6), 589–600 (1989)
33. Pearn, W.L.: Augment-Insert Algorithms for the Capacitated Arc Routing Problem. Computers & Operations Research 18(2), 189–198 (1991)
34. Santos, L., Coutinho-Rodrigues, J., Current, J.: An Improved Ant Colony Optimization Based Algorithm for the Capacitated Arc Routing Problem. Transportation Research Part B 44, 246–266 (2010)
35. Wøhlk, S.: Contributions to Arc Routing, Ph.D. dissertation, Faculty of Social Sciences, University of Southern Denmark (2005)
36. Wøhlk, S.: New Lower Bound for the Capacitated Arc Routing Problem. Computers & Operations Research 33(12), 3458–3472 (2006)