Column Generation Bounds for the Capacitated Arc Routing Problem

Rafael Martinelli
Diego Pecin
Marcus Poggi de Aragão

Departamento de Informática
PUC-Rio, Rio de Janeiro–RJ, Brazil
{ rmartinelli, dpecin, poggi }@inf.puc-rio.br

Humberto Longo

Instituto de Informática
Universidade Federal de Goiás, Goiás–GO, 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. This work presents new lower bounds for this problem. They are based on formulations on variables representing routes. These are solved by column generation. Valid inequalities from formulations on arc variables are added to the route based formulations. Issues regarding the use of elementary and nonelementary routes are explored. Extensive experiments are presented.

Keywords: CARP, Column Generation, Dual Ascent. Main Area: Mathematical Programming.

Resumo

Problemas de roteamento sobre arcos estão entre os mais desafiantes na área de otimização combinatória. Neste trabalho, o Problema de Roteamento de Veículos sobre Arcos é abordado, onde as demandas estão distribuídas sobre um subconjunto de arestas de um grafo dado, chamado de conjunto de arestas obrigatórias. Os custos de percorrer as arestas, as demandas das obrigatórias e a capacidade dos veículos idênticos disponíveis em um vértice de depósito são dados. Procura-se por rotas que atendam todas as demandas com um custo mínimo. Este trabalho apresenta novos limites inferiores para este problema. Eles são baseados em formulações sobre variáveis que representam rotas. Estas são resolvidas por geração de colunas. Desigualdades válidas a partir de formulações em variáveis de arestas são adicionadas às formulações baseadas em rotas. Questões relativas à utilização de rotas elementares e não-elementares são exploradas. Experiementos são apresentados.

Palavras-Chave: CARP, Geração de Colunas, Dual Ascent. Área Principal: Programação Matemática.

1 Introduction

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

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

We want to find a solution minimizing the sum of the costs of the walks. It can be noted that $\sum_{e \in E_R} c(e)$ is a trivial lower bound on the cost of an optimal solution, the remaining costs in a solution are the costs of the deadheaded edges.

This problem was first presented by Golden and Wong in 1981 Golden and Wong (1981) and has been used to model many situations, including street garbage collection, postal delivery and routing of electric meter readers. The reader may refer to Dror (2000) for further applications.

The *CARP* is strongly *NP*-hard (Golden and Wong (1981)). Several heuristics have been proposed for it. Among them we can cite Golden et al. (1983), Chapleau et al. (1984), Ulusoy (1985), Pearn (1989, 1991), Hertz et al. (2000), Hertz and Mittaz (2001), Brandão and Eglese (2008), Santos et al. (2009) and Mei et al. (2009a,b). Many other heuristics are described in Assad and Golden (1995), Eiselt et al. (1995) and Dror (2000). Algorithms yielding lower bounds for the *CARP* are presented in Golden and Wong (1981), Assad et al. (1987), Pearn (1988), Benavent et al. (1992), Amberg and Voß (2002), Belenguer and Benavent (2003) and Wøhlk (2006).

As previously mentioned, *CARP* is a very difficult problem to be solved exactly. For this reason, few works that use an exact approach to this problem were made. We can cite the most recent and successful. In Gómez-Cabrero et al. (2005), a set covering formulation with additional cuts was proposed and a column generation, which allowed nonelementary routes (routes servicing the same edge more than once), together with a cutting plane were used to try to solve this formulation. In Longo et al. (2006), the algorithm first transforms the instance into a Capacitated Vehicle Routing Problem instance (*CVRP*) and, after that, the instance is solved using the algorithm proposed in Fukasawa et al. (2006). In Letchford and Oukil (2009), two column generation algorithms were proposed over the same set covering formulation from Gómez-Cabrero et al. (2005). The first algorithm was similar to the one proposed in Gómez-Cabrero et al. (2005), but simpler. The second allowed only elementary routes and was implemented with different mixed-integer formulations.

Our present goal is to show the best results for the *CARP*, using algorithms specifically developed for this problem that generate columns allowing and not allowing nonelementary routes, with additional cuts. In section 2, we show the evolution of the formulations until the set partitioning formulation used in the rest of this work and the so-called capacity cuts. In section 3, we explain our pricing algorithm which allows only elementary routes and one formulation from Letchford and Oukil (2009) used just to compare our values. In section 4, we explain our pricing algorithm which allows nonelementary routes. In section 5, we define our Dual Ascent Heuristic, used to generate the capacity cuts. Computational experiments are in section 6 and our conclusions in section 7.

2 Mathematical Formulation

2.1 Edge Formulation

The intuitive formulation for the *CARP* is to create one boolean variable associated with each serviced edge for each vehicle (x_e^p) and an integer variable to represent the number of times each edge is deadheaded by each available vehicle (z_e^p). This approach is usually called the *two-index formulation*, because both variables types have indexes e for edge and p for vehicle.

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

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

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

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

$$x_e^p \in \{0,1\} \qquad \forall e \in E_R, p \in I \tag{5}$$

$$z_e^p \in \mathbb{Z}^+ \qquad \forall e \in E, p \in I \tag{6}$$

Constraints (2) assure unique vehicle service for each required edge, constraints (3) limit the total demand serviced by each vehicle to the capacity Q and, given $E_R(S) = \{(i,j) \in E_R | i \in S, j \in S\}$, where S is a vertex set, constraints (4) are sub-tour elimination constraints, used to force a closed walk for every vehicle.

2.2 Set Partitioning Formulation

The structure of the *CARP* suggests to reformulate it as a Set Partitioning Problem (*SPP*) in order to apply column generation. A solution to the *CARP* consists of a set of routes (interpreted as columns), one for each vehicle, which can be calculated independently, since they cover the set of required edges to be serviced. In a column generation approach, the master problem is a *SPP* like the following formulation:

$$\text{MIN} \quad \sum_{r \in \Omega} c_r \lambda_r \tag{7}$$

$$s.t. \quad \sum_{r \in \Omega} \lambda_r = k \tag{8}$$

$$\sum_{r \in \Omega} a_e^r \lambda_r = 1 \quad \forall e \in E_R \tag{9}$$

$$\lambda_r \in \{0,1\} \quad \forall r \in \Omega \tag{10}$$

where Ω is the set of feasible routes, c_r is the cost of the route r and a_e^r is 1 if route r services the required edge $e \in E_R$ or is 0 otherwise. The constraints (8) specify the number of routes in a solution, as each vehicle is associated with a single route. Constraints (9) indicate that each required edge $e \in E_R$ is serviced by exactly one route r.

This formulation allows only *elementary routes*. An elementary route can service only once a required edge. A *nonelementary route* can service more than once a required edge. In order to allow these routes, one can relax the λ_r variables to be continuous or change the equal sign from (9) to a greater-equal sign. We chose the first approach.

2.3 Capacity Cuts

In Belenguer and Benavent (2003), a cutting plane algorithm was devised for the *CARP* which uses cuts over z_e variables, called *capacity cuts*. However, these cuts are not enough to give a *formulation* for the *CARP*, they are only a *relaxation* since they allow some integer solutions that do not correspond to feasible solutions. Instead, this relaxation give very good lower bounds for the *CARP*. Using these cuts together with others defined in Belenguer and Benavent (2003), the cutting plane algorithm matched the best heuristic solutions on 47 out of 87 instances tested from the literature.

Given a set $S \in V\backslash\{0\}$, $\delta(S) = \{(i,j) \in E | i \in S, j \notin S\}$, $\delta_R(S) = \{(i,j) \in E_R | i \in S, j \notin S\}$ and a lower bound $k(S)$ on the number of vehicles needed to service the required edges of this set S, the capacity cuts are defined as follows:

$$\sum_{e \in \delta(S)} z_e \geq 2k(S) - |\delta_R(S)| \quad \forall S \subseteq V\backslash\{0\} \tag{11}$$

To calculate exactly the value of $k(S)$, we need to solve a Bin Packing Problem, which is *NP*-hard, as described in Garey and Johnson (1979). Instead of it, a good relaxation is $k(S) = \lceil \sum_{e \in E_R(S) \cup \delta_R(S)} q_e / Q \rceil$.

In addition to the capacity cuts, Belenguer and Benavent defined the so-called *odd edge cutset cuts*. For any set S, if $|\delta_R(S)|$ is odd, at least one edge from $\delta(S)$ must be deadheaded.

$$\sum_{e \in \delta(S)} z_e \geq 1 \quad \forall S \subseteq V\backslash\{0\} \text{ with } |\delta_R(S)| \text{ odd} \tag{12}$$

Using these two cuts together with only deadheaded variables z_e, the usually called *one-index formulation* can be created with the following objective function:

$$\text{MIN} \quad \sum_{e \in E} c_e z_e \tag{13}$$

For the cutting plane algorithm from Belenguer and Benavent (2003), several separation routines were used to identify violated (11) and (12) cuts. The separation for (12) cuts can be done in polynomial time using the Odd Minimum Cut algorithm from Padberg and Rao (1982). For the (11), the separation is more difficult and was solved by heuristics in Belenguer and Benavent (2003) and exactly in Ahr (2004).

3 Pricing with Elementary Routes

The pricing subproblem consists of finding an elementary route (column) of minimum reduced cost within a given maximum load capacity. This subproblem is a special case of the Elementary Shortest Path Problem with Resource Constraints (*ESPPRC*) where the only resource constraint is the vehicle's capacity. The *ESPPRC* is strongly *NP*-hard Dror (1994), but one can eventually solve this problem exactly when the graph is sparse (Letchford and Oukil (2009)).

3.1 Dynamic Programming Approach

In the dynamic programming approach, our algorithm follows the label correcting approach from Feillet et al. (2004). The data structure is a $Q \times |V|$ matrix M. Each entry

$M(q, v)$ represents a bucket that consists of a set of partial elementary (meaning that a required edge must be serviced at most once) walks that start at the depot and end at vertex v with a total demand exactly q on the required edges serviced (we assign a label to each partial walk). Each label belonging to the bucket $M(q, v)$ records the cost, the reduced cost and the set of required edges serviced (not necessarily the order in which these edges are serviced) of the partial walk represented. We use dynamic programming to fill the matrix M starting with $q = 0$ and going up to $q = Q$. When extending a label we rely upon the possibility of fathoming labels that cannot be part of an optimal solution following the dominance tests criterion described in Feillet et al. (2004) (i.e., we only record non-dominated labels).

To further reduce the number of labels to be handled by the dynamic programming algorithm, we apply the relaxation technique proposed in Righini and Salani (2008) to the *ESPPRC*. For this purpose, we identify some required edges as *critical*, according to the structure of an optimal solution obtained when nonelementary walks are allowed. The algorithm works iteratively: let Θ be the set of critical required edges of the current iteration. In the subsequent iteration, the dynamic programming algorithm prevents multiple visits to the edges in Θ, still allowing multiple visits to the others. At the final of this iteration we update the critical edge set as $\Theta' = \Psi \cup \Theta$, where Ψ denotes the set of required edges serviced more than once on the optimal solution found at the final of the current iteration. Obviously, the algorithm eventually provides an optimal solution that corresponds to an elementary route.

3.2 Mixed-Integer Programming Approach

In Letchford and Oukil (2009), the *ESPPRC* was solved using different MIP formulations, all of them exploiting the sparsity of G. Oddly, the results found using the dynamic programming approach were different from the ones found with the MIP formulations from Letchford and Oukil (2009). To investigate this problem, we decided to implement one of the MIP formulations. More exactly, we implemented the *directed* MIP formulation, which defines binary variables x_{ij} and x_{ji} for each edge $(i, j) \in E_R$, representing if edge (i, j) is serviced. It also defines binary variables z_{ij} and z_{ji} for each edge $(i, j) \in E$, representing if edge (i, j) is *traversed* (serviced or not). Finally, it defines continuous variables f_{ij} and f_{ji}, representing the remaining load on the vehicle when it goes from i to j.

$$\text{MIN} \quad \sum_{\{i,j\}\in E} c_{ij}\,(z_{ij} + z_{ji}) - \sum_{\{i,j\}\in E_R} c_{ij}\,(x_{ij} + x_{ji}) \tag{14}$$

$$s.t. \quad x_{ij} + x_{ji} \leq 1 \qquad \forall (i,j) \in E_R \tag{15}$$

$$z_{ij} \geq x_{ij}, z_{ji} \geq x_{ji} \qquad \forall (i,j) \in E_R \tag{16}$$

$$\sum_{\{j,i\}\in\delta^-(i)} z_{ji} - \sum_{(i,j)\in\delta^+(i)} z_{ij} = 0 \qquad \forall i \in V\backslash\{0\} \tag{17}$$

$$\sum_{\{j,i\}\in\delta^-(i)} f_{ji} - \sum_{(i,j)\in\delta^+(i)} f_{ij} = \sum_{(i,j)\in\delta_R^+(i)} q_{ij}x_{ij} \qquad \forall i \in V\backslash\{0\} \tag{18}$$

$$\sum_{\{0,j\}\in\delta^+(0)} f_{0j} - \sum_{\{j,0\}\in\delta^-(0)} f_{j0} + \sum_{\{0,j\}\in\delta_R^+(0)} q_{0j}x_{0j} \leq Q \tag{19}$$

$$f_{ij} \leq Qz_{ij}, f_{ji} \leq Qz_{ji} \qquad \forall (i,j) \in E\backslash E_R \tag{20}$$

$$f_{ij} \leq Qz_{ij} - q_{ij}x_{ij}, f_{ji} \leq Qz_{ji} - q_{ji}x_{ji} \qquad \forall (i,j) \in E_R \tag{21}$$

$$f_{ij}, f_{ji} \in \mathbb{R} \qquad \forall (i,j) \in E \tag{22}$$

$$x_{ij}, x_{ji} \in \{0,1\} \qquad \forall (i,j) \in E_R \tag{23}$$

$$z_{ij}, z_{ji} \in \{0,1\} \qquad \forall (i,j) \in E \tag{24}$$

4 Pricing with Nonelementary Routes

We can relax pricing with elementary routes to nonelementary which in the *CARP*'s case has the meaning that a required edge can be serviced more than once in a single route. What motivates us to do pricing with nonelementary routes is that this can be solved in pseudopolynomial time, as shown in Christofides et al. (1981).

Our basic algorithm for nonelementary pricing is based on dynamic programming. We use a $Q \times |E_R|$ matrix M where each entry $M(q,(i,j))$ (respectively, a $Q \times |E_R|$ matrix M' where each entry $M'(q,(j,i))$) represent the least reduced costly walk that reaches the required edge $(i,j) \in E_R$ servicing it from vertex i to j (respectively, from vertex j to i) using total demand exactly q. Initially, the only known label represents an empty walk and has reduced cost zero. All other entries of the matrices M and M' are initialized with empty labels that represent partial walks with infinite cost. Then both matrices M and M' are filled with dynamic programming, starting with $q = 1$ and going up to $q = Q$. For each entry $M(q,(i,j))$ (respectively, $M'(q,(j,i))$, the algorithm goes through each required edge $(k,l) \in E_R$ and evaluates the reduced cost of the walk represented by $\min\{M(q,(k,l)) + dist(l,i), M'(q,(l,k)) + dist(k,i)\} + \bar{c}_{ij}$ (respectively, $\min\{M(q,(k,l)) + dist(l,j), M'(q,(l,k)) + dist(k,j)\} + \bar{c}_{ji}$) to update the reduced cost represented by the entry $M(q,(i,j))$ (respectively, $M'(q,(j,i))$) (where $dist(i,j)$ is the shortest path from vertex i to j and $\bar{c}_{ij}$ is the reduced cost of the required edge $(i,j) \in E_R$). Figure 1 shows the dynamic programming matrix M and how we update the entry $M(q,(i,j))$.

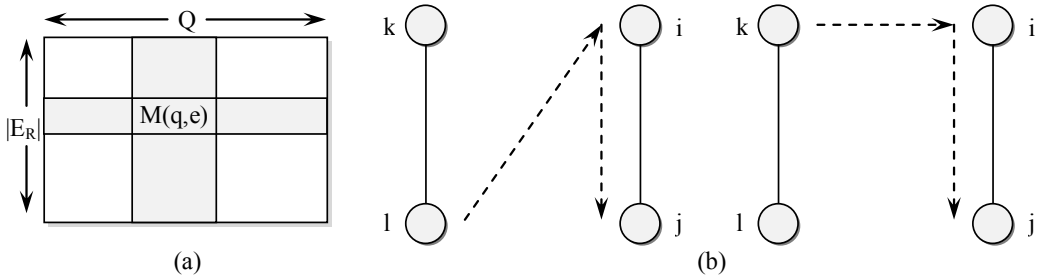

Figure 1: (a) The dynamic programming matrix; (b) Updating an entry of the matrix.

To strengthen the formulation, we look for walks that do not have any sequence $e' \rightarrow e \rightarrow e'$ of edges serviced, following the dominance criterion described in Christofides et al. (1981). It is a 2-cycle elimination on the required edges.

In addition, we include in this pricing routine a modification of the cuts defined in section 2.3. We just converted the *capacity cuts* (11) and the *odd edge cutset cuts* (12) into λ_r variables. We also define b_e^r, the number of times route r deadheads edge e.

$$\sum_{r\in\Omega}\sum_{e\in\delta(S)} b_e^r \lambda_r \geq 2k(S) - |\delta_R(S)| \quad \forall S \subseteq V\backslash\{0\} \tag{25}$$

$$\sum_{r\in\Omega}\sum_{e\in\delta(S)} b_e^r \lambda_r \geq 1 \quad \forall S \subseteq V\backslash\{0\} \text{ with } |\delta_R(S)| \text{ odd} \tag{26}$$

These cuts affect the reduced cost of the deadheadeds edges, which in turn affects the values of the shortest path between each pair of vertices ($dist(i,j)$). In each iteration of the pricing routine, these values must be recalculated in order to reflect the reduced cost variation.

5 The Dual Ascent Heuristic

Aiming to accelerate the discovery of cuts, a heuristic that works in the dual formulation of the one-index formulation was created, called the dual ascent heuristic (*DAH*). The *DAH* works on a modified graph $G' = (V', E)$, which can have united vertices. At each iteration, the heuristic generates several cuts by following the four strategies described below:

- *Simple Cut Set*: Any single vertex in the current graph;
- *Complement Cut Set*: The complement of any single vertex in the current graph;
- *Random Connected Component Cut Set*: Randomly choose the size of the cutset ($1 \leq |S| \leq |V'|$), randomly choose a start vertex and go on adding adjacent vertices. First add all adjacent to the first vertex, then all to the second one and so on, until S has the required size. Repeat this operation $2|E'|$ times;
- *Minimum Spanning Tree (MST) Cut Set*: Calculate the *MST* of the graph and create all the possible cuts having only one edge from the *MST* in $\delta(S)$.

After that, the heuristic chooses the cut with the biggest right hand side (*rhs*) and subtracts the value $min(c_e : e \in \delta(S))$ from the costs of all edges in $\delta(S)$ and adds $min(c_e : e \in \delta(S)) * rhs$ to the value of the solution. Then, the heuristics unite the vertices adjacent to all edges having $c_e = 0$. This whole procedure is repeated until there is only one vertex in G'.

6 Computational Experiments

All algorithms for the *CARP* were implemented in C++ programming language, using CPLEX 12.1 as linear programming solver. Tests were conducted on a Intel Core 2 Duo 2.8 GHz, using just one core, with 4GB of RAM.

In the computational experiment, we applied our algorithms to the instances of the datasets *kshs*, *gdb*, *bccm* (usually known as *val*) and *egl*, available at `http://www.uv.es/~belengue/carp.html`. These datasets were originally used in Kiuchi et al. (1995) (*kshs*), DeArmon (1981) and Golden et al. (1983)(*gdb*), Benavent et al. (1992) (*bccm*), and Li (1992) and Li and Eglese (1996)(*egl*). Except for the *bccm* instances, that are named *1A*, *1B*, $\cdots$, *6C*, the remaining three sets have their names starting with the names of the sets.

From these four sets, three of them *kshs*, *gdb* and *bccm* were randomly generated following different construction patterns, where subsets were obtained by varying the underlying graph, the vehicle capacity or the capacities themselves. In these three sets of instances all edges are required, i.e., E_R equals E. The fourth set, *egl*, was 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.

Tables (1), (2), (3) and (4) show results for each of the instance sets mentioned. The first column is the instance name, the following two columns present the best known lower and upper bounds (*LB* and *UB*). The column *NER-L* shows the results reported by Letchford and Oukil (2009) for the pricing allowing nonelementary routes. The following columns presents our results using the pricing with elementary routes (*ER*), the pricing allowing nonelementary routes with 2-cycle elimination without capacity cuts (*NER*), the dual ascent heuristic (*DAH*) and the pricing allowing nonelementary routes with 2-cycle elimination with capacity cuts generated by the dual ascent heuristic (*NERC*). Values marked in **bold** show

when a specific algorithm found the optimal value for that instance and values in *italics* show when a specific algorithm found a new lower bound for that instance.

For all instances tested, the lower bounds obtained using the dynamic programming algorithm described in section 3.1 are the same obtained using our own implementation of the directed MIP pricing from section 3.2. This suggests that there may have been some problem in generating the results reported by Letchford and Oukil (2009).

Instance	LB	UB	NER-L	ER	NER	DAH	NERC
kshs1	14661	14661	13363	13553	13876	**14661**	**14661**
kshs2	9863	9863	8195	8723	8929	**9863**	**9863**
kshs3	9320	9320	8401	8614	8538	**9320**	**9320**
kshs4	11498	11498	11442	11297	**11498**	11098	**11498**
kshs5	10957	10957	10215	10358	10370	**10957**	**10957**
kshs6	10197	10197	9080	9232	9345	**10197**	**10197**

Table 1: Computational results for *kshs* instances.

Instance	LB	UB	NER-L	ER	NER	DAH	NERC
gdb1	316	316	282	285	288	**316**	**316**
gdb2	339	339	313	314	318	**339**	**339**
gdb3	275	275	248	250	254	**275**	**275**
gdb4	287	287	266	272	270	**287**	**287**
gdb5	377	377	358	360	364	**377**	**377**
gdb6	298	298	282	285	287	**298**	**298**
gdb7	325	325	288	293	293	**325**	**325**
gdb8	348	348	319	331	330	344	347
gdb9	303	303	291	294	294	**303**	**303**
gdb10	275	275	254	254	256	**275**	**275**
gdb11	395	395	364	363	368	**395**	**395**
gdb12	458	458	422	445	445	450	453
gdb13	536	536	525	526	525	**536**	**536**
gdb14	100	100	98	99	**100**	**100**	**100**
gdb15	58	58	56	57	**58**	**58**	**58**
gdb16	127	127	122	122	122	**127**	**127**
gdb17	91	91	85	86	87	**91**	**91**
gdb18	164	164	159	159	**164**	**164**	**164**
gdb19	55	55	47	**55**	**55**	**55**	**55**
gdb20	121	121	107	114	114	**121**	**121**
gdb21	156	156	151	152	152	**156**	**156**
gdb22	200	200	196	197	197	**200**	**200**
gdb23	233	233	**233**	**233**	**233**	**233**	**233**

Table 2: Computational results for *gdb* instances.

7 Conclusions

In this work, we showed different pricing strategies for the *CARP* based on set partitioning formulation. First we showed a new dynammic algorithm for pricing with elementary routes, which had never been done specifically for the *CARP*. We compared our results using this approach with the only other similar work available (Letchford and Oukil (2009)) and found an inconsistency on the results reported. After implementing their approach, we could validate the results we showed in section 6.

Instance	LB	UB	NER-L	ER	NER	DAH	NERC
1A	173	173	146	146	146	**173**	**173**
1B	173	173	149	150	149	**173**	**173**
1C	245	245	220	226	235	233	242
2A	227	227	199	206	202	**227**	**227**
2B	259	259	229	234	233	257	258
2C	457	457	444	457	457	455	**457**
3A	81	81	68	69	69	79	80
3B	87	87	75	77	77	**87**	**87**
3C	138	138	129	131	131	135	136
4A	400	400	356	357	357	**400**	**400**
4B	412	412	368	370	370	**412**	**412**
4C	428	428	389	394	393	426	426
4D	530	530	493	500	498	514	522
5A	423	423	381	382	383	**423**	**423**
5B	446	446	402	405	405	443	444
5C	474	474	431	435	435	467	469
5D	577	577	539	547	547	569	572
6A	223	223	193	198	195	**223**	**223**
6B	233	233	202	208	204	229	229
6C	317	317	290	298	299	301	311

Table 3: Computational results for *bccm* instances.

Instance	LB	UB	NER-L	ER	NER	DAH	NERC
egl-e1-a	3548	3548	2983	3426	3395	3527	3544
egl-e1-b	4498	4498	3791	4290	4246	4372	4430
egl-e1-c	5566	5595	4931	5473	5424	5427	5528
egl-e2-a	5018	5018	4221	4832	4707	4915	4971
egl-e2-b	6305	6317	5463	6106	5978	6193	6259
egl-e2-c	8243	8335	7679	8188	8116	7936	8196
egl-e3-a	5898	5898	5076	5706	5607	5809	5870
egl-e3-b	7704	7777	6882	7542	7476	7487	7613
egl-e3-c	10163	10305	9434	10086	9957	9971	10128
egl-e4-a	6408	6456	5634	6234	6118	6272	6345
egl-e4-b	8884	9000	8048	8678	8558	8784	8839
egl-e4-c	11427	11601	10770	11411	11281	11285	*11431*
egl-s1-a	5018	5018	4170	4986	4803	4887	4996
egl-s1-b	6384	6388	5542	6285	6077	6143	6287
egl-s1-c	8493	8518	7716	8424	8247	8230	8403
egl-s2-a	9824	9956	8867	9667	9520	9598	9789
egl-s2-b	12968	13165	12146	12802	12713	12697	12927
egl-s2-c	16353	16505	15618	16262	16190	16049	16292
egl-s3-a	10143	10260	9190	9925	9777	9833	10076
egl-s3-b	13616	13807	12752	13388	13309	13369	13570
egl-s3-c	17100	17234	16390	17014	16950	16797	17067
egl-s4-a	12143	12341	11314	11906	11810	11886	12074
egl-s4-b	16093	16442	15266	15754	15688	15826	16026
egl-s4-c	20375	20591	19651	20144	20090	20086	20346

Table 4: Computational results for *egl* instances.

Then, we showed a pricing algorithm which allows nonelementary routes using 2-cycle elimination on the required edges and additional cuts. Gómez-Cabrero et al. (2005) is the only work which used a similar approach, but it did not reported detailed results. Therefore, our results are the best found so far using an algorithm specifically devised for the *CARP*.

There are possibilities for future research. For the nonelementary pricing algorithm, one can implement the exact separation of the capacity cuts from Ahr (2004), a *k-cycle elimination* procedure as the one described in Irnich and Villeneuve (2006) or a complete branch-cut-and-price algorithm. For the elementary pricing algorithm, one can include in the formulation a modification of the lifting of the capacity cuts described on Baldacci et al. (2008).

8 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 311997/06-6.

References

Ahr, D. (2004). *Contributions to Multiple Postmen Problems*. PhD thesis, Department of Computer Science, Heidelberg University.

Amberg, A. and Voß, S. (2002). A Hierarquical Relaxations Lower Bound for the Capacitated Arc Routing Problem. In *Proceedings of the 35th Anual Hawaii International Conference on System Sciences*.

Assad, A. A. and Golden, B. L. (1995). Arc Routing Methods and Applications. In Ball, M. G., Magnanti, T. L., Monma, C. L., and Nemhauser, G. L., editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, chapter 5, pages 375–483. Elsevier.

Assad, A. A., Pearn, W. L., and Golden, B. L. (1987). The Capacitated Chinese Postman Problem: Lower Bounds and Solvable Cases. *American Journal of Mathematical Management Sciences*, 7(1,2):63–88.

Baldacci, R., Christofides, N., and Mingozzi, A. (2008). An Exact Algorithm for the Vehicle Routing Problem Based on the Set Partitioning Formulation with Additional Cuts. *Math. Program.*, 115(2):351–385.

Belenguer, J. M. and Benavent, E. (2003). A Cutting Plane Algorithm for the Capacitated Arc Routing Problem. *Computers & Operations Research*, 30:705–28.

Benavent, E., Campos, V., Corberan, A., and Mota, E. (1992). The Capacitated Arc Routing Problem: Lower Bounds. *Networks*, 22:669–690.

Brandão, J. and Eglese, R. (2008). A Deterministic Tabu Search Algorithm for the Capacitated Arc Routing Problem. *Computers & Operations Research*, 35:1112–1126.

Chapleau, L., Ferland, J. A., Lapalme, G., and Rousseau, J. M. (1984). A Parallel Insert Method for the Capacitated Arc Routing Problem. *Operations Research Letters*, 3(2):95––99.

Christofides, N., Mingozzi, A., and Toth, P. (1981). Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations. *Mathematical Programming*, 20:255–282.

DeArmon, J. S. (1981). A Comparison of Heuristics for the Capacitated Chinese Postman Problem. Master's thesis, University of Maryland, Colledge Park, MD.

Dror, M. (1994). Note on the Complexity of the Shortest Path Models for Column Generation in VRPTW. *Operations Research*, 42(5):977–978.

Dror, M. (2000). *Arc Routing: Complexity and Approximability*, chapter 4, pages 133–169. Kluwer Academic Publishers.

Eiselt, H. A., Gendreau, M., and Laporte, G. (1995). Arc Routing Problems, Part I: The Chinese Postman Problem. *Operations Research*, 43(2):231–242.

Feillet, D., Dejax, P., Gendreau, M., and Gueguen, C. (2004). An Exact Algorithm for the Elementary Shortest Path Problem with Resource Constraints: Application to some Vehicle Routing Problems. *Networks*, 44(3):216–229.

Fukasawa, R., Longo, H., Lysgaard, J., Poggi de Aragão, M., Reis, M., Uchoa, E., and Werneck, R. F. (2006). Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem. *Mathematical Programming*, 106(3):491–511.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

Gómez-Cabrero, D., Belenguer, J. M., and Benavent, E. (2005). Cutting Plane and Column Generation for the Capacitated Arc Routing Problem. In *ORP3, Valencia*.

Golden, B. L., DeArmon, J. S., and Baker, E. K. (1983). Computational Experiments with Algorithms for a Class of Routing Problems. *Computers and Operations Research*, 10(1):47–59.

Golden, B. L. and Wong, R. T. (1981). Capacitated Arc Routing Problems. *Networks*, 11:305–315.

Hertz, A., Laporte, G., and Mittaz, M. (2000). A Tabu Search Heuristic for the Capacitated Arc Routing Problem. *Operations Research*, 48(1):129–135.

Hertz, A. and Mittaz, M. (2001). A Variable Neighborhood Descent Algorithm for the Capacitated Arc Routing Problem. *Transportation Science*, 35(4):425–434.

Irnich, S. and Villeneuve, D. (2006). The Shortest-Path Problem with Resource Constraints and k-Cycle Elimination for $k \geq 3$. *INFORMS J. on Computing*, 18(3):391–406.

Kiuchi, M., Shinano, Y., Hirabayashi, R., and Saruwatari, Y. (1995). An exact algorithm for the Capacitated Arc Routing Problem Using Parallel Branch and Bound Method. In *Abstracts of the 1995 Spring National Conference of the Oper. Res. Soc. of Japan*, pages 28–29. in Japanese.

Letchford, A. and Oukil, A. (2009). Exploiting Sparsity in Pricing Routines for the Capacitated Arc Routing Problem. *Computers & Operations Research*, 36:2320–2327.

Li, L. Y. O. (1992). *Vehicle Routeing for Winter Gritting*. PhD thesis, Dept. of Management Science, Lancaster University.

Li, L. Y. O. and Eglese, R. W. (1996). An Interactive Algorithm for Vehicle Routeing for Winter-Gritting. *Journal of the Operational Research Society*, 47:217–228.

Longo, H., Poggi de Aragão, M., and Uchoa, E. (2006). Solving Capacitated Arc Routing Problems Using a Transformation to the CVRP. *Computers & Operations Research*, 33:1823–1827.

Mei, Y., Tang, K., and Yao, X. (2009a). A Global Repair Operator for Capacitated Arc Routing Problem. *IEEE Transactions on Systems, Man and Cybernetics*, 39(3):723–734.

Mei, Y., Tang, K., and Yao, X. (2009b). Improved Memetic Algorithm for Capacitated Arc Routing Problem. In *IEEE Congress on Evolutionary Computation*.

Padberg, M. W. and Rao, M. R. (1982). Odd minimum cut-sets and b-matchings. *Mathematics of Operations Research*, 7:67–80.

Pearn, W. L. (1988). New Lower Bounds for the Capacitated Arc Routing Problem. *Networks*, 18:181–191.

Pearn, W. L. (1989). Approximate Solutions for the Capacitated Arc Routing Problem. *Computers & Operations Research*, 16(6):589–600.

Pearn, W. L. (1991). Augment-Insert Algorithms for the Capacitated Arc Routing Problem. *Computers & Operations Research*, 18(2):189–198.

Righini, G. and Salani, M. (2008). New Dynamic Programming Algorithms for the Resource Constrained Elementary Shortest Path Problem. *Networks*, 51(3):155–170.

Santos, L., Coutinho-Rodrigues, J., and Current, J. (2009). An Improved Heuristic for the Capacitated Arc Routing Problem. *Computers & Operations Research*, 36:2632–2637.

Ulusoy, G. (1985). The Fleet Size and Mix Problem for Capacitated Arc Routing. *European Journal of Operational Research*, 22(3):329—337.

Wøhlk, S. (2006). New Lower Bound for the Capacitated Arc Routing Problem. *Computers & Operations Research*, 33(12):3458–3472.