# Solving Capacitated Arc Routing Problems using a transformation to the CVRP

Humberto Longo
Instituto de Informática,
Universidade Federal de Goiás, Brazil,
longo@inf.ufg.br

Marcus Poggi de Aragão
Departamento de Informática,
Pontifícia Universidade Católica do Rio de Janeiro, Brazil,
poggi@inf.puc-rio.br

Eduardo Uchoa
Departamento de Engenharia de Produção,
Universidade Federal Fluminense, Brazil,
uchoa@producao.uff.br

PUC-RioInf. MCC10/04 April, 2004.

## Abstract

A well known transformation by Pearn, Assad and Golden reduces a Capacitated Arc Routing Problem (CARP) into an equivalent Capacitated Vehicle Routing Problem (CVRP). However, that transformation is regarded as unpractical, since an original instance with $r$ required edges is turned into a CVRP over a complete graph with $3r + 1$ vertices. We propose a similar transformation that reduces this graph to $2r + 1$ vertices, with the additional restriction that $r$ edges are already fixed to 1. Using a recent branch-and-cut-and-price algorithm for the CVRP, we observed that it yields an effective way of attacking the CARP, being significantly better than the exact methods created specifically for that problem. Computational experiments obtained improved lower bounds for almost all open instances from the literature. Several such instances could be solved to optimality.

**Keywords**: CARP, CVRP, Routing, Mixed-Integer Programming.

## Resumo

Problemas de roteamento de veículos podem ter demandas nos vértices ou nas arestas. No primeiro caso o problema é conhecido como o problema de roteamento de veículos com restrição de capacidade (CVRP - Capacitated Vehicle Routing Problem) e corresponde na realidade ao caso em que o veículo é carregado em sua parada no vértice (cliente). No segundo caso, as demandas estão ao longo das arestas da rota, não necessariamente de todas. Este segundo problema é conhecido como de roteamento de veículos sobre os arcos com restrição de capacidade (CARP - Capacitated Arc Routing Problem). Usando-se a transformação proposta por Pearn, Assad e Golden, o CARP pode ser transformado no CVRP. Contudo, esta abordagem é praticamente inviável, uma vez que uma instância do CARP com $r$ arestas com demanda gera um CVRP com $3r + 1$ vértices. Este artigo apresenta uma transformação similar que reduz a instância resultante do CVRP a um grafo de $2r + 1$ vértices, onde as soluções válidas para o CARP serão apenas os conjuntos de rotas que utilizarem $r$ arestas previamente determinadas. Usando-se um algoritmo recente de *branch-and-cut-and-price* para o CVRP, essa nova transformação mostrou-se eficaz na resolução do CARP, sendo significativamente melhor do que os métodos exatos criados especificamente para esse problema. As experiências computacionais melhoraram os limites inferiores para o valor da solução ótima em quase todas as instâncias testadas. Muitas dessas instâncias da literatura puderam ser resolvidas até a otimalidade pela primeira vez.

**Palavras Chaves**: CARP, CVRP, Roteamento de veículos, Programação Linear Inteira.

# 1 Introduction

The Capacitated Arc Routing Problem (CARP) can be defined as follows. Suppose a connected undirected graph $G = (V, E)$, costs $c : E \rightarrow Z^+$, demands $w : E \rightarrow Z^+$, vehicle capacity $Q$ and a distinguished depot vertex labelled 0. Define $R = \{e \in E \mid w(e) > 0\}$ 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*. 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 $\sum_{e \in R} c(e)$ 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. In the remaining of this article, let $r$ denote the number of edges in $R$.

This problem was first presented by Golden and Wong in 1981 ([14]) and has been used to model many situations, including street garbage collection, postal delivery, routing of electric meter readers, etc [11].

The CARP is stronlgy NP-hard. Several heuristics have been proposed for it. Among them we can cite Golden et al. [4], Chapleau et al. [9], Ulusoy [27], Pearn [23, 24] and Hertz et al. [15]. Many other heuristics are described in [2] [12] and [11].

On the other hand, as far as we know, the only exact algorithms for it are the branch-and-bound algorithms by Hirabayashi, Saruwatari and Nishida [16], Kiuchi et al. [17], Welz [28], and Belenguer and Benavent [5]. Even using the fast machines available today, those algorithms can only solve small instances, with less than 30 required edges.

Algorithms yielding lower bounds for the CARP are presented by Golden and Wong [14], Assad et al. [3], Pearn [22], Benavent et al.[7], Amberg and Voß[1] and Wøhlk [30]. The current best bounds are those obtained by a cutting plane algorithm by Belenguer and Benavent [6]. Those bounds are much better than previous ones and matched the best heuristic solutions on 47 out of 87 instances tested from the literature. The largest instance thus solved had 97 required edges. The algorithm uses cuts over $z$ variables, where $z(e)$ represents the number of times edge $e$ is deadheaded. However, those cuts are not enough to give a *formulation* for the CARP, they are only a *relaxation* since they allow some some integer solutions that do not correspond to feasible solutions. This rules out the use of those bounds in a consistent exact algorithm because *no one knows how to determine in polynomial time whether an integer solution $z$ is feasible or not.* In other words, that bound can at most prove that a given heuristic CARP solution is indeed optimal, and only on instances without duality gap.

The approach proposed in this work, reducing the CARP to a CVRP that is given as input to a branch-and-cut-and-price algorithm, not only gives lower bounds even better than those by [6], it is already an exact algorithm that can solve many open instances from the literature.

## 2 Pearn, Assad and Golden's Transformation

Pearn, Assad and Golden [25] proposed a transformation of the CARP to the CVRP that replaces each edge of $R$ by three vertices. An edge $(i,j)$ in $R$ is associated to vertices $s_{ij}$ and $s_{ji}$, referred as side vertices, and to $m_{ij}$, the middle vertex. A CVRP instance is defined on the complete undirected graph $H = (N, A)$, where

$$N = \bigcup_{(i,j)\in R} \{s_{ij}, s_{ji}, m_{ji}\} \bigcup \{0\}.$$

Vertex 0 is defined as the depot. The edge costs $d : A \rightarrow Z^+$ and the demands $q : N \rightarrow Z^+$ are defined as follows.

$$\begin{aligned}
d(s_{ij}, s_{kl}) &= \begin{cases} \frac{1}{4}(c_{ij} + c_{kl}) + dist(i,k) & \text{if } (i,j) \neq (k,l) \\ 0 & \text{if } (i,j) = (k,l) \end{cases} \\
d(0, s_{ij}) &= \tfrac{1}{4}c_{ij} + dist(0,i) \\
d(m_{ij}, v) &= \begin{cases} \frac{1}{4}c_{ij} & \text{if } v = s_{ij} \text{ or } s_{ji} \\ \infty & \text{otherwise,} \end{cases}
\end{aligned}$$

where $dist(i,j)$ is the value of the shortest path from vertex $i$ to vertex $j$ in $G$, calculated using the costs $c$. The new demands are

$$q(s_{ij}) = q(m_{ij}) = q(s_{ji}) = \frac{1}{3}.w(i,j).$$

The resulting instance has $3r + 1$ vertices. Since the current best CVRP codes [8, 20, 29, 26, 13] can only cope consistently with instances up to 100 vertices, the practical use of that transformation is limited.

## 3 New Transformation

We observed that, in the above transformation, the construction around the middle vertex ($m_{ij}$) has only the purpose of obliging a CVRP route to pass through the three vertices corresponding to $(i,j)$ in sequence (either $s_{ij} \rightarrow m_{ij} \rightarrow s_{ji}$ or $s_{ji} \rightarrow m_{ij} \rightarrow s_{ij}$). Our idea to avoid the middle vertices is simply to fix $r$ edges to 1 in the transformed CVRP instance. The new transformation is described as follows.

An edge $(i,j)$ in $R$ is now associated only to vertices $s_{ij}$ and $s_{ji}$. The resulting CVRP instance is defined on the complete undirected graph $H = (N, A)$, where

$$N = \bigcup_{(i,j)\in R} \{s_{ij}, s_{ji}\} \bigcup \{0\}.$$

Vertex 0 is defined as the depot. The edge costs $d : A \rightarrow Z^+$ and the demands $q : N \rightarrow Z^+$ are defined as follows.

$$\begin{aligned}
d(s_{ij}, s_{kl}) &= \begin{cases} 0 & \text{if } (i,j) = (k,l) \\ c(i,j) & \text{if } (i,j) = (l,k) \\ dist(i,k) & \text{if } (i,j) \neq (k,l), (i,j) \neq (l,k) \end{cases} \\
d(0, s_{ij}) &= dist(0,i)
\end{aligned}$$

where $dist(i,j)$ have the same meaning as in the previous section. The new demands are

$$q(s_{ij}) = q(s_{ji}) = \frac{1}{2}.w(i,j).$$

Finally, we fix all edges $\{(s_{ij}, s_{ji}) \,|\, (i,j) \in R\}$ to 1, meaning that we only accept CVRP solutions where $s_{ij}$ and $s_{ji}$ are visited in sequence, either $s_{ij} \to s_{ji}$ or $s_{ji} \to s_{ij}$.

We give as example of the transformation an instance with 4 vertices and 5 edges (all required), where $V = \{0,1,2,3\}$ and edge costs are $c(0,1) = 2$, $c(0,2) = 4$, $c(1,2) = 1$, $c(1,3) = 3$ and $c(2,3) = 5$. The transformed instance with 11 vertices is shown in Figure 1. The corresponding $d$ function is on Table 1.

The important point is that the fixing of $r$ edges to 1 is not an additional burden to most CVRP algorithms, quite to the contrary, they easy a lot the solution of a transformed instance. Although the transformation results in an instance where $|N| = 2.r + 1$, the mandatory fixing of edges can make an algorithm to perform almost as if on an instance with $|N| = r + 1$. We address this point in the next section.

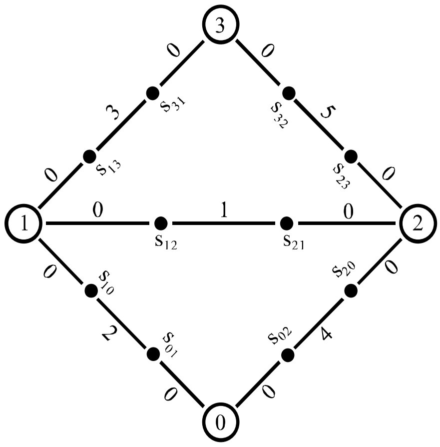

Figure 1: Transformation example.

| 0 | $s_{01}$ | $s_{10}$ | $s_{02}$ | $s_{20}$ | $s_{12}$ | $s_{21}$ | $s_{13}$ | $s_{31}$ | $s_{23}$ | $s_{32}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 2 | 0 | 3 | 2 | 3 | 2 | 5 | 3 | 5 |
| $s_{01}$ | - | 2 | 0 | 3 | 2 | 3 | 2 | 5 | 3 | 5 |
| $s_{10}$ | | - | 2 | 1 | 0 | 1 | 0 | 3 | 1 | 3 |
| $s_{02}$ | | | - | 4 | 2 | 3 | 2 | 5 | 3 | 5 |
| $s_{20}$ | | | | - | 1 | 0 | 1 | 4 | 0 | 4 |
| $s_{12}$ | | | | | - | 1 | 0 | 3 | 1 | 3 |
| $s_{21}$ | | | | | | - | 1 | 4 | 0 | 4 |
| $s_{13}$ | | | | | | | - | 3 | 1 | 3 |
| $s_{31}$ | | | | | | | | - | 4 | 0 |
| $s_{23}$ | | | | | | | | | - | 5 |
| $s_{32}$ | | | | | | | | | | - |

Table 1: Intervertices distances for the example.

# 4 Formulation and Algorithm

The goal of this section is to present how a CVRP formulation can be slightly specialized to instances coming from the CARP, in order to take advantage of having $r$ edges that must belong to every solution.

The Explicit Master formulation for the CVRP, as presented in Fukasawa et al. [13], combines a column generation formulation with the classical formulation on edge variables. Let $H = (N, A)$, $d$, $q$ and $Q$ define a CVRP instance having vertex 0 as the depot and the remaining vertices in $N$ as clients.

This formulation follows.

$$
EM-CVRP: \left\{
\begin{array}{llcll r}
\min & \sum\limits_{e=(u,v)\in A} d(e).x_e & & & & (0)\\
\text{subject to} & & & & & \\
 & \sum\limits_{e\in\delta(\{u\})} x_e & = & 2 & \forall\, u \in N\setminus\{0\} & (1)\\
 & \sum\limits_{e\in\delta(\{0\})} x_e & \geq & 2\cdot K^* & & (2)\\
 & \sum\limits_{e\in\delta(S)} x_e & \geq & 2\cdot k(S) & \forall\, S \subseteq N\setminus\{0\} & (3)\\
 & x_e & \leq & 1 & \forall\, e \in A\setminus\delta(\{0\}) & (4)\\
\sum\limits_{l=1}^{p} q_l^e\cdot\lambda_l \quad - & x_e & = & 0 & \forall e \in A & (5)\\
 & x_e & \in & \{0,1,2\} & \forall\, e\in A & \\
\lambda_l & & \geq & 0 & \forall\, l \in \{1,\ldots,p\}\ . &
\end{array}
\right.
$$

- Variable $x_e$ represents the number of times that edge $e$ is traversed by a vehicle. This variable can assume value 2 if $e$ is adjacent to the depot, corresponding to a route with a single client.
- A *q-route* is a walk that starts at the depot, traverses a sequence of clients with total demand at most $Q$, and returns to the depot. Each variable $\lambda_l$ is associated to one of the $p$ possible $q$-routes. Let $q_l^e$ be the number of times edge $e$ appears in the $l$-th $q$-route.
- Degree constraints (1) states that each client vertex is served by exactly one vehicle. Constraint (2) requires that at least $K^*$ vehicles leave and return to the depot. This number, representing the minimum number of vehicles to service all clients, is calculated by solving a Bin-Packing Problem. The *rounded capacity* constraints stated in (3) use $k(S) = \lceil \sum_{u\in S} q(u)/Q\rceil$ as a lower bound on the minimum number of vehicles necessary to service the clients in set $S \subset N$. Constraints (5) oblige $x$ to be a linear combination of $q$-routes. The integrality constraints complete the formulation.

Now we proceed with the simplifications that come from the fact that $H = (N, A)$, $d$, $q$ and $Q$ comes from a CARP instance, as defined in the previous section, and that all edges $\{(s_{ij}, s_{ji}) \mid (i,j) \in R\}$ are fixed to 1.

- Since all vehicles appearing in a CARP solution must service at least one edge in $R$ and that each edge is represented as two vertices in $H$, we can restrict all $x$ variables to be less or equal to 1.

- We can restrict the $q$-paths to those that visit vertices $s_{ij}$ and $s_{ji}$ in sequence, either $s_{ij} \to s_{ji}$ or $s_{ji} \to s_{ij}$. This eliminates many $\lambda$ variables. Pricing $q$-paths over a complete graph with $n$ vertices takes $O(n^2.Q)$ time. The above restriction does not change that complexity.

- Half of the constraints (1) can be eliminated. The redefinition of the $q$-paths and the constraints (5) guarantee that if the degree constraint corresponding to $s_{ij}$ is satisfied, the degree constraint corresponding to $s_{ji}$ will also be satisfied. This observation have a crucial impact on the performance of the resulting branch-and-cut-and-price, because the size of LPs and the convergence of the column generation part of the algorithm depends a lot on the number of such constraints. In other words, in those aspects the algorithm behaves as if it were solving an instance with $r$ clients and not $2r$.

- It is well-known (see, for instance [20, 21]) that when separating capacity cuts, edges $(u, v)$ having value 1 in the fractional solution should be contracted. This happens because if a capacity cut over a set $S$ having $(u, v) \in \delta(S)$ is violated, a capacity cut over set $S \cup \{u, v\}$ is also violated. This means that the corresponding separation algorithm always work on a graph with at most $r + 1$ vertices.

This specialized formulation, called $EM-CARP$, is now presented.

$$
EM-CARP: \left\{
\begin{array}{lcclll}
\min & & \sum\limits_{e=(u,v)\in A} d(e).x_e & & & (0)\\
\text{subject to} & & & & & \\
& & \sum\limits_{e\in\delta(\{s_{ij}\})} x_e & = & 2 \quad \forall\,(i,j)\in R & (1')\\
& & \sum\limits_{e\in\delta(\{0\})} x_e & \geq & 2\cdot K^* & (2)\\
& & \sum\limits_{e\in\delta(S)} x_e & \geq & 2\cdot k(S) \quad \forall\,S \subseteq N\setminus\{0\} & (3)\\
& & x_e & \leq & 1 \quad \forall\,e\in A & (4')\\
\sum\limits_{l=1}^{p} q_l^e\cdot\lambda_l & - & x_e & = & 0 \quad \forall e\in A & (5)\\
& & x_e & \in & \{0,1\} \quad \forall\,e\in A & \\
\lambda_l & & & \geq & 0 \quad \forall\,l\in\{1,\ldots,p\}\ . &
\end{array}
\right.
$$

A more compact formulation, the one actually used in the algorithm, is obtained if every occurrence of $x_e$ in (0), (1'), (2), (3) is replaced by its equivalent given by (5). Relaxing the integrality constraints, a Linear Program, referred $DWM-CARP$, is obtained. The solution of that LP gives a valid lower bound.

$$\mathrm{DWM-CARP} \quad = \quad \begin{cases} \min & \sum_{l=1}^{p} \sum_{e \in E} d(e) \cdot q_l^e \cdot \lambda_l & & & (7) \\ \text{s.t.} & \sum_{l=1}^{p} \sum_{e \in \delta(\{s_{ij}\})} q_l^e \cdot \lambda_l & = & 2 & \forall\, (i,j) \in R & (8) \\ & \sum_{l=1}^{p} \sum_{e \in \delta(\{0\})} q_l^e \cdot \lambda_l & \geq & 2 \cdot K^* & & (9) \\ & \sum_{l=1}^{p} \sum_{e \in \delta(S)} q_l^e \cdot \lambda_l & \geq & 2 \cdot k(S) & \forall\, S \subseteq N \setminus \{0\} & (10) \\ & \sum_{l=1}^{p} q_l^e \cdot \lambda_l & \leq & 1 & \forall\, e \in A & (11) \\ & \lambda_l & \geq & 0 & \forall\, l \in \{1, \ldots, p\} \ . & \end{cases}$$

# 5 Computational Results

Our CARP code is basically an adaptation of the implementation of a robust branch-and-cut-and-price algorithm for the CVRP described in [13]. Apart from a new pricing algorithm to restrict the $q$-paths to those visiting vertices $s_{ij}$ and $s_{ji}$ in sequence, we only changed the linear programming part to be like indicated in DWM-CARP. Tests were conducted on a Pentium IV, 2.4 GHz, with 1GB of RAM.

In the computational experiment we applied our algorithm to the instances of the datasets *kshs*, *gdb*, *bccm* and *egl*, available at `http://www.uv.es/ ~ belengue/ carp.html`. These datasets were originally used in Kiuchi et al. [17] (*kshs*), DeArmon [10] and Golden et al. [4](*gdb*), Belenguer et al. [7](*bccm*), and Li [18] and Li and Eglese [19](*egl*), respectively. Except for the *bccm* instances, that are named $1.A$, $1.B$, ..., $10D$, the remaining three sets have their names starting with the names of the sets.

Sets *kshs*, *gdb* and *bccm* were randomly generated following different construction patterns, varying the underlying graph, the vehicle capacity or the capacities itself. In these three sets of instances all edges are required, i.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). Costs and demands are proportional to the length of the edges, except for non-required edges that have zero demand.

In the first two tables we compare the lower bound given by the root node of our branch-and-cut-and-price algorithm (the solution of $DWM - CARP$) with the best known lower bounds of all instances from sets *egl* (Table 2) and *bccm* (Table 3). The columns of both tables list the name of the instance and its characteristics, the number of vertices $|V|$, the number of required edges $r$, the minimum number of vehicles $K^*$ to cover the demands, the best known upper bound $UB$, the previous best lower bound (all of them obtained in [6]), the lower bound given by our transformation $Our\ LB$, and the corresponding CPU time $RootTime$ in seconds. Lower bounds in bold indicate a matching with the best upper bound.

The results show that our lower bounds are equal or better than the previous best on almost all instances. On only two instances, $2.B$ and $5.B$, our bound was one unit smaller. It can be observed that our bounds are strictly better on all instances where $K^* > 5$, indicating that its quality is less sensitive to the use of many vehicles. The new bounds are also strictly better on all *egl* instances.

The last table presents the instances where we could run the complete branch-and-cut-and-price algorithm, solving them to optimality. The columns of this table have the same first four columns of the previous two tables. They are followed by columns with the optimal value of the instance $OPT$, the CPU time of our algorithm at the root node $ROOT\ Time$, the number of nodes in the branch-and-bound tree $Tree\ Nodes$, and the CPU time spent in the column generation algorithm $CG\ Time$, the cut generation procedures $Cut\ Time$ and total CPU time $Total\ Time$. The optimal values that are in bold indicate that optimality was proven for the first time. This was the case for 10 instances, the last one open from set *kshs*, the last three from set *gdb*, four from *bccm* and two from set *egl*. On one instance (gdb13) the optimal solution found improved upon the previous best heuristic solution.

Overall, the CPU times where reasonably small, which gives hope to closing some more of the remaining open instances on sets *bccm* and *egl* instances in the near future. Moreover, we believe that the lower bounds can be further improved with the addition of cuts that take into account the specific structure of the CARP, for example, the ones proposed in Belenguer and Benavent [6].

# References


[1] A. Amberg and S. Voß. A hierarquical relaxations lower bound for the capacitated arc routing problem. In *Proceedings of the 35th Anual Hawaii International Conference on System Sciences*, 2002.

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

[3] A. A. Assad, B. L. Golden, and W. Pearn. The Capacitated Chinese Postman Problem: Lower Bounds and Solvable Cases. *Am. J. Math. Management Sci*, 7(1,2):63–88, 1987.

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

[5] J. M. Belenguer and E. Benavent. The capacitated arc routing problem: Valid inequalities and facets. *Computational Optimization & Applications*, 10(2):165–187, 1998.

[6] J.M. Belenguer and E. Benavent. A Cutting Plane Algorithm for the Capacitated Arc Routing Problem. *Computers & Operations Research*, pages 705–728, 2003.

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

[8] U. Blasum and W. Hochstättler. Application of the branch and cut method to the vehicle routing problem. Technical Report ZPR2000-386, Zentrum fur Angewandte Informatik Köln, 2000.

[9] L. Chapleau, J. A. Ferland, G. Lapalme, and J.-M. Rousseau. A parallel insert method for the capacitated arc routing problem. *Operations Research Letters*, 3(2):95–99, 1984.

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

[11] M. Dror. *Arc Routing: Theory, Solutions and Applications*. Kluwer Academic Publishers, 2000.

[12] H. A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems II: The Rural Postman Problem. *Operations Research*, 43(3):339–414, 1995.

[13] R. Fukasawa, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa, and R. F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. In *Proceedings of the X IPCO*, Lecture Notes in Computer Science, New York, June 2004. To appear.

[14] B. L. Golden and R. T. Wong. Capacitated arc routing problems. *Networks*, 11:305–315, 1981.

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

[16] R. Hirabayashi, N. Nishida, and Y. Saruwatari. Node duplication lower bounds for the capacitated arc routing problems. *Journal of the Operations Research Society of Japan*, 35(2):119–133, 1992.

[17] M. Kiuchi, Y. Shinano, R. Hirabayashi, and Y. Saruwatari. 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, 1995. in Japanese.

[18] L. Y. O. Li. *Vehicle Routing for Winter Gritting*. PhD thesis, Dept. of Management Science, Lancaster University, 1992.

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

[20] J. Lysgaard, A. Letchford, and R. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 2003. To appear.

[21] D. Naddef and G. Rinaldi. Branch-and-cut algorithms for the capacitated VRP. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, chapter 3, pages 53–84. SIAM, 2002.

[22] W.-L Pearn. New lower bounds for the capacitated arc routing problem. *Networks*, 18:181–191, 1988.

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

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

[25] W. L. Pearn, A. Assad, and B. L. Golden. Transforming Arc Routing into Node Routing Problems. *Computers & Operations Research*, 14(4):285–288, 1987.

[26] T. Ralphs, L. Kopman, W. Pulleyblank, and L. Trotter Jr. On the capacitated vehicle routing problem. *Mathematical Programming*, 94:343–359, 2003.

[27] G. Ulusoy. The fleet size and mix problem for capacitated arc routing. *European Journal of Operational Research*, 22(3):329–337, 1985.

[28] S. A. Welz. *Optimal solutions for the capacitated arc routing problem using integer programming*. PhD thesis, Department of QT and OM, University of Cincinnati, 1994.

[29] K.M. Wenger. *Generic Cut Generation Methods for Routing Problems*. PhD thesis, Institute of Computer Science, University of Heidelberg, 2003.

[30] Sanne W$\phi$hlk. New Lower Bound for the Capacitated Arc Routing Problem. Preprint PP-2003-15, University of Southern Denmark, IMADA - Department of Mathematics and Computer Science, 2003.

| Instance | \|V\| | \|E\| | r | K* | UB | Prev. LB | Our LB | Root Time (s) |
|---|---|---|---|---|---|---|---|---|
| egl-e1-a | 77 | 98 | 51 | 5 | 3548 | 3515 | **3548** | 144.203 |
| egl-e1-b | 77 | 98 | 51 | 7 | 4498 | 4436 | 4468 | 52.594 |
| egl-e1-c | 77 | 98 | 51 | 10 | 5595 | 5453 | 5542 | 46.063 |
| egl-e2-a | 77 | 98 | 72 | 7 | 5018 | 4994 | 5011 | 521.219 |
| egl-e2-b | 77 | 98 | 72 | 10 | 6340 | 6249 | 6280 | 198.531 |
| egl-e2-c | 77 | 98 | 72 | 14 | 8415 | 8114 | 8234 | 66.875 |
| egl-e3-a | 77 | 98 | 87 | 8 | 5898 | 5869 | **5898** | 924.859 |
| egl-e3-b | 77 | 98 | 87 | 12 | 7822 | 7646 | 7697 | 375.609 |
| egl-e3-c | 77 | 98 | 87 | 17 | 10433 | 10019 | 10163 | 142.125 |
| egl-e4-a | 77 | 98 | 98 | 9 | 6461 | 6372 | 6395 | 1171.580 |
| egl-e4-b | 77 | 98 | 98 | 14 | 9021 | 8809 | 8884 | 418.984 |
| egl-e4-c | 77 | 98 | 98 | 19 | 11779 | 11276 | 11427 | 202.688 |
| egl-s1-a | 140 | 190 | 75 | 7 | 5018 | 4992 | 5014 | 750.391 |
| egl-s1-b | 140 | 190 | 75 | 10 | 6435 | 6201 | 6379 | 204.500 |
| egl-s1-c | 140 | 190 | 75 | 14 | 8518 | 8310 | 8480 | 66.984 |
| egl-s2-a | 140 | 190 | 147 | 14 | 9995 | 9780 | 9824 | 3260.250 |
| egl-s2-b | 140 | 190 | 147 | 20 | 13174 | 12886 | 12968 | 896.706 |
| egl-s2-c | 140 | 190 | 147 | 27 | 16795 | 16221 | 16353 | 408.878 |
| egl-s3-a | 140 | 190 | 159 | 15 | 10296 | 10025 | 10143 | 1680.371 |
| egl-s3-b | 140 | 190 | 159 | 22 | 14053 | 13554 | 13616 | 1639.530 |
| egl-s3-c | 140 | 190 | 159 | 29 | 17297 | 16969 | 17100 | 635.292 |
| egl-s4-a | 140 | 190 | 190 | 19 | 12442 | 12027 | 12143 | 14318.100 |
| egl-s4-b | 140 | 190 | 190 | 27 | 16531 | 15933 | 16093 | 2761.130 |
| egl-s4-c | 140 | 190 | 190 | 35 | 20832 | 20179 | 20375 | 1119.980 |
| Average gap % | | | | | | 2.40 | 1.45 | |
| Maximum gap % | | | | | | 4.27 | 3.11 | |

Table 2: Lower bound comparison for the egl instances.

| Instance | \|V\| | r | K* | UB | Prev. LB | Our LB | Root Time (s) |
|---|---|---|---|---|---|---|---|
| 1.A | 24 | 39 | 2 | 247 | **247** | **247** | 98.278 |
| 1.B | 24 | 39 | 3 | 247 | **247** | **247** | 54.561 |
| 1.C | 24 | 39 | 8 | 319 | 309 | 312 | 771.882 |
| 2.A | 24 | 34 | 2 | 298 | **298** | **298** | 79.404 |
| 2.B | 24 | 34 | 3 | 330 | **330** | 329 | 169.010 |
| 2.C | 24 | 34 | 8 | 528 | 526 | **528** | 0.969 |
| 3.A | 24 | 35 | 2 | 105 | **105** | **105** | 127.590 |
| 3.B | 24 | 35 | 3 | 111 | **111** | **111** | 134.339 |
| 3.C | 24 | 35 | 7 | 162 | 161 | 161 | 3.156 |
| 4.A | 41 | 69 | 3 | 522 | **522** | **522** | 2475.280 |
| 4.B | 41 | 69 | 4 | 534 | **534** | **534** | 1178.370 |
| 4.C | 41 | 69 | 5 | 550 | **550** | **550** | 824.599 |
| 4.D | 41 | 69 | 9 | 652 | 644 | 648 | 76.576 |
| 5.A | 34 | 65 | 3 | 566 | **566** | **566** | 629.417 |
| 5.B | 34 | 65 | 4 | 589 | **589** | 588 | 388.144 |
| 5.C | 34 | 65 | 5 | 617 | 612 | 613 | 274.772 |
| 5.D | 34 | 65 | 9 | 724 | 714 | 716 | 62.779 |
| 6.A | 31 | 50 | 3 | 330 | **330** | **330** | 158.667 |
| 6.B | 31 | 50 | 4 | 340 | 338 | 337 | 169.254 |
| 6.C | 31 | 50 | 10 | 424 | 418 | 420 | 119.371 |
| 7.A | 40 | 66 | 3 | 382 | **382** | **382** | 319.349 |
| 7.B | 40 | 66 | 4 | 386 | **386** | **386** | 163.754 |
| 7.C | 40 | 66 | 9 | 437 | 436 | 436 | 607.637 |
| 8.A | 30 | 63 | 3 | 522 | **522** | **522** | 359.426 |
| 8.B | 30 | 63 | 4 | 531 | **531** | **531** | 168.885 |
| 8.C | 30 | 63 | 9 | 663 | 653 | 654 | 376.863 |
| 9.A | 50 | 92 | 3 | 450 | **450** | **450** | 17722.600 |
| 9.B | 50 | 92 | 4 | 453 | **453** | **453** | 4520.030 |
| 9.C | 50 | 92 | 5 | 459 | **459** | **459** | 1460.890 |
| 9.D | 50 | 92 | 10 | 518 | 509 | 512 | 305.158 |
| 10.A | 50 | 97 | 3 | 637 | **637** | **637** | 13336.600 |
| 10.B | 50 | 97 | 4 | 645 | **645** | **645** | 13719.400 |
| 10.C | 50 | 97 | 5 | 655 | **655** | **655** | 5078.570 |
| 10.D | 50 | 97 | 10 | 739 | 732 | 734 | 473.632 |
| Average gap % | | | | | 0.41 | 0.32 | |
| Maximum gap % | | | | | 3.13 | 2.19 | |

Table 3: Lower bound comparison for the bccm instances.

| Instance | \|V\| | r | K* | Opt | Root Time (s) | Tree Nodes | GC Time (s) | Cut Time (s) | Total Time (s) |
|---|---|---|---|---|---|---|---|---|---|
| kshs1 | 8 | 15 | 4 | 14661 | 0.719 | 2 | 0.656 | 0.172 | 0.828 |
| kshs2 | 10 | 15 | 4 | 9863 | 0.375 | 2 | 0.391 | 0.047 | 0.453 |
| kshs3 | 6 | 15 | 4 | 9320 | 0.688 | 4 | 0.766 | 0.219 | 1.016 |
| kshs4 | 8 | 15 | 4 | **11498** | 0.828 | 3 | 0.531 | 0.391 | 0.953 |
| kshs5 | 8 | 15 | 3 | 10957 | 0.703 | 1 | 0.656 | 0.047 | 0.703 |
| kshs6 | 9 | 15 | 3 | 10197 | 0.734 | 3 | 1.391 | 0.078 | 1.484 |
| gdb1 | 12 | 22 | 5 | 316 | 1.859 | 10 | 1.359 | 1.672 | 3.125 |
| gdb2 | 12 | 26 | 6 | 339 | 0.531 | 5 | 1.125 | 0.141 | 1.359 |
| gdb3 | 12 | 22 | 5 | 275 | 0.406 | 5 | 0.531 | 0.234 | 0.844 |
| gdb4 | 11 | 19 | 4 | 287 | 0.281 | 8 | 0.625 | 0.125 | 0.813 |
| gdb5 | 13 | 26 | 6 | 377 | 0.891 | 2 | 0.422 | 0.578 | 1.016 |
| gdb6 | 12 | 22 | 5 | 298 | 0.250 | 1 | 0.203 | 0.047 | 0.250 |
| gdb7 | 12 | 22 | 5 | 325 | 0.625 | 10 | 1.203 | 0.422 | 1.750 |
| gdb8 | 27 | 46 | 10 | **348** | 2.750 | 28 | 30.922 | 0.859 | 32.469 |
| gdb9 | 27 | 51 | 10 | 303 | 12.016 | 7 | 22.141 | 8.313 | 30.969 |
| gdb10 | 12 | 25 | 4 | 275 | 1.063 | 10 | 5.500 | 0.656 | 6.344 |
| gdb11 | 22 | 45 | 5 | 395 | 19.438 | 29 | 1351.130 | 9.688 | 1364.630 |
| gdb12 | 13 | 23 | 7 | **458** | 0.703 | 27 | 4.422 | 0.406 | 5.219 |
| gdb13 | 10 | 28 | 6 | **536** | 1.125 | 145 | 88.063 | 1.484 | 93.031 |
| gdb14 | 7 | 21 | 5 | 100 | 0.156 | 1 | 0.156 | 0 | 0.156 |
| gdb15 | 7 | 21 | 4 | 58 | 0.781 | 9 | 3.094 | 0.531 | 3.750 |
| gdb16 | 8 | 28 | 5 | 127 | 2.313 | 19 | 24.406 | 1.563 | 26.422 |
| gdb17 | 8 | 28 | 5 | 91 | 0.953 | 23 | 33.328 | 0.703 | 34.828 |
| gdb18 | 9 | 36 | 5 | 164 | 5.578 | 16 | 53.641 | 2.172 | 56.391 |
| gdb19 | 8 | 11 | 3 | 55 | 0.063 | 1 | 0.047 | 0 | 0.063 |
| gdb20 | 11 | 22 | 4 | 121 | 1.375 | 7 | 3.719 | 0.656 | 4.500 |
| gdb21 | 11 | 33 | 6 | 156 | 3.969 | 13 | 17.781 | 2.734 | 20.859 |
| gdb22 | 11 | 44 | 8 | 200 | 6.406 | 29 | 105.109 | 5.156 | 112.078 |
| gdb23 | 11 | 55 | 10 | 233 | 9.547 | 30 | 136.438 | 6.625 | 145.453 |
| 1C | 24 | 39 | 8 | **319** | 774.906 | 6119 | 7802.220 | 830.156 | 8916.78 |
| 2B | 24 | 34 | 3 | 330 | 169.484 | 14 | 581.656 | 86.938 | 671.250 |
| 2C | 24 | 34 | 8 | **528** | 0.938 | 1 | 0.891 | 0.047 | 0.953 |
| 3C | 24 | 35 | 7 | **162** | 3.141 | 363 | 314.719 | 4.438 | 328.891 |
| 7C | 31 | 66 | 9 | **437** | 39.781 | 3 | 54.609 | 9.500 | 131.969 |
| egl-e1-a | 77 | 51 | 5 | **3548** | 143.844 | 1 | 128.391 | 15.1719 | 143.875 |
| egl-e3-a | 77 | 87 | 8 | **5898** | 923.656 | 1 | 831.188 | 91.7969 | 923.766 |

Table 4: Results of the BCP algorithm for the kshs, gdb, bccm and egl instances.