A Lower Bound for the Node, Edge, and Arc Routing Problem

Lukas Bach
Centre for OR Applications in Logistics
Dept. of Economics and Business
Aarhus University
Fuglesangs Allé 4
DK-8210 Aarhus V, Denmark
`luba@asb.dk`

Geir Hasle
Dept. of Applied Mathematics
SINTEF ICT
P.O. Box 124 Blindern
NO-0314, Oslo, Norway
Phone: +47-93058703
`Geir.Hasle@sintef.no`

Sanne Wøhlk
Centre for OR Applications in Logistics
Dept. of Economics and Business
Aarhus University
Fuglesangs Allé 4
DK-8210 Aarhus V, Denmark
`sanw@asb.dk`

November 15, 2012

Abstract

The Node, Edge, and Arc Routing Problem (NEARP) was defined by Prins and Bouchenoua in 2004, although similar problems have been studied before. This problem, also called the Mixed Capacitated General Routing Problem (MCGRP), generalizes the classical Capacitated Vehicle Routing Problem (CVRP), the Capacitated Arc Routing Problem (CARP), and the General Routing Problem. It captures important aspects of real-life routing problems that were not adequately modeled in previous Vehicle Routing Problem (VRP) variants. The authors also proposed a memetic algorithm procedure and defined a set of test instances called the CBMix benchmark. The NEARP definition and investigation contribute to the development of rich VRPs. In this paper we present the first lower bound procedure for the NEARP. It is a further development of lower bounds for the CARP. We also define two novel sets of test instances to complement the CBMix benchmark. The first is based on well-known CARP instances; the second consists of real life cases of newspaper delivery routing. We provide numerical results in the form of lower and best known upper bounds for all instances of all three benchmarks. For three of the instances, the gap between the upper and lower bound is closed. The average gap is 25.1%. As the lower bound procedure is based on a high quality lower bound procedure for the CARP, and there has been limited work on approximate solution methods for the NEARP, we suspect that a main reason for the rather large gaps is the quality of the upper bound. This fact, and the high industrial relevance of the NEARP, should motivate more research on approximate and exact methods for this important problem.

Keywords: Vehicle Routing; Node Routing; Arc Routing; General Routing; VRP; CARP; NEARP; MCGRP; Bound; Benchmark; Experiment

1 Introduction

The Vehicle Routing Problem (VRP) captures the essence of allocation and routing of vehicles at minimal cost, given transportation demand. Hence, it is central to effective and efficient transportation management. VRP research is regarded as one of the great successes of Operations Research, partly due to the emergence of a solution tool industry. Results have been disseminated and exploited in industry. The VRP, construed in a wide sense, is a family of problems. Since the first definition of the classical, Capacitated VRP (CVRP) in 1959 [17], many generalizations have been studied in a systematic fashion. Typically, exact and approximate solution methods have been proposed and investigated for each new VRP variant that has been defined. For an introduction and a survey of the VRP literature, we refer to [35, 24].

The VRP is a computationally very hard discrete optimization problem. For industrial cases of reasonable size, one normally has to resort to approximate methods. Efficient procedures for generating proven lower bounds for the optimal value are important both to practice and theory. First, they may speed up exact methods. Second, they provide a benchmark for approximate methods that provide feasible solutions and hence upper bounds on the optimal value. Obviously, a zero gap between an upper and a lower bound for a given instance proves that the value is optimal. A large gap may be due to a poor quality lower bound, a feasible solution of bad quality, or both.

There has been a tremendous increase in the ability to produce exact and approximate solutions to VRP variants over the past 50 years. A few years ago, the best exact methods could consistently solve instances of the CVRP with up to some 70 customers to optimality in reasonable time. Today, the number is above 100, see for instance [7]. Approximate methods such as metaheuristics, matheuristics, and heuristic column generation seem to provide high quality solutions in realistic times even for large-size instances of complex VRP variants. For a categorized bibliography of metaheuristics for the VRP, we refer to [23]. Doerner and Schmid give a survey of matheuristics for VRPs in [18]. In [21], Feillet gives a tutorial on column generation for the VRP.

As problems are regarded as being solved for practical purposes, researchers turn to new extensions and larger-size instances. This trend is enhanced by market pull from the tool industry and their end users. The somewhat imprecise term "rich VRP" has recently been introduced to denote variants that are close to capturing all the essential aspects of some subset of real-life routing problems. Generalizations of models in the literature are defined, exact and approximate methods are proposed and investigated, and lower bounds are developed.

In contrast to the CVRP where demand for service is located in the nodes of the network, arc routing problems have been proposed to model the situation where demand is located on edges or arcs in a transportation network [19]. Of particular industrial relevance is the Capacitated Arc Routing Problem (CARP) defined by Golden and Wong in 1981 [25] and its generalizations, as the CARP model contains multiple vehicles with capacity.

There has been a tendency in the literature to dichotomize routing problems into arc routing problems and node routing problems. Some cases are naturally modeled as arc routing because the demand is fundamentally defined on arcs or edges in a transportation network. Prime examples are street sweeping, gritting, and snow clearing. However, the arc routing model has been advocated in the literature for problems where the demand is located in nodes, for instance distribution of subscription newspapers to households and municipal pickup of waste, particularly in urban areas. In real-life cases, there are often thousands or tens of thousands of points to be serviced along a subset of all road links in the area. Such cases are often formulated as CARPs, typically with a drastic reduction of problem size.

In their 2004 paper [34], Prins and Bouchenoua motivate and define the Node, Edge, and Arc Routing Problem (NEARP)[1]. They state that:

> Despite the success of metaheuristics for the VRP and the CARP, it is clear that these two problems cannot formalize the requirements of many real-world scenarios.

Their example is urban waste collection, where most demand may adequately be modeled on street segments, but there may also be demand located in points, for instance at supermarkets. Hence, they motivate a generalization of both the classical CVRP and the CARP. To this end, they define the NEARP as a combination of the CVRP and the CARP, which can also be viewed as a capacitated extension of the General Routing Problem [32]. They propose a memetic algorithm for the NEARP and investigate it empirically on standard CVRP and CARP instances from the literature. The authors also create a NEARP benchmark consisting of 23 grid-based test cases, the so-called CBMix-instances, and provide experimental results for their proposed algorithm.

We would like to enhance the motivation for the NEARP and further emphasize its high importance to practice. The arc routing model for node-based demand cases such as subscription newspaper delivery is based on an underlying idea of abstraction. Some form of abstraction may be necessary to contain the computational complexity resulting from a large number of demand points in industrial routing. The assumption that all point-based demands can be aggregated into edges or arcs may be crude in practice. It may lead to solutions that are unnecessarily costly, as partial traversal of edges is not possible. In industry, a route planning task may cover areas that have a mixture of urban, suburban, and rural parts where many demand points will be far apart and aggregation would impose unnecessary constraints on visit sequences. A more sophisticated type of abstraction is aggregation of demand based on the underlying transportation network topology. Such aggregation procedures must also take capacity, time, and travel restrictions into consideration to avoid aggregation that would lead to impractical or low quality plans. In general, such procedures will produce a NEARP instance with a combination of demands on arcs, edges, and nodes. It is therefore imperative to eliminate the arc/node routing dichotomy and thus enable the modeling of the continuum of node and arc routing problems needed for representational adequacy in real-life situations. The introduction of the NEARP was a significant step towards the goal of rich VRP.

[1]The NEARP may also be denoted the Mixed Capacitated General Routing Problem.

Despite its importance, studies of the NEARP are scarce in the literature. The first we know of is the paper by Pandit and Muralidharan from 1995 [33]. They address a generalized version of the NEARP, i.e., routing a heterogeneous fixed fleet of vehicles over specified segments and nodes of a street network, and also include a route duration constraint. The problem is denoted the Capacitated General Routing Problem (CGRP). The authors formally define the CGRP and design a heuristic for solving it. They generate random test instances inspired from curb-side waste collection in residential areas on a network with 50 nodes and 100 arcs. They also investigate the proposed method on random instances of the the Capacitated Chinese Postman Problem for which they had two lower bound procedures.

In [26], the homogeneous fleet specialization of the CGRP studied by Pandit and Muralidharan is investigated by Gutierrez, Soler, and Hervaz. They call the problem the Capacitated General Routing Problem on Mixed Graphs (CGRP-m) and propose a heuristic that compares favorably with the heuristic by Pandit and Muralidharan on the homogeneous fleet case.

Kokubugata, Moriyama, and Kawashima [29] study problems from city logistics, including the VRP with Time Windows and the NEARP. They propose a Simulated Annealing metaheuristic for solving these problems. Computational results for the CBMix instances of Prins and Bouchenoua are presented, with several improvements. In [28], Hasle et al. describe results from experiments on NEARP test instances using their industrial VRP solver Spider [27, 3], and report new best-known results.

The first integer programming formulation for the NEARP was developed in a forth by Bosco et al. [11]. They extended valid inequalities for the CARP to the NEARP, and embedded them into a branch-and-cut algorithm that was tested on 12 sets of instances constructed from CARP benchmarks. The proposed method could solve only small-size instances, involving at most seven vehicles. Optimal solutions were also provided for two of the smallest CBMix instances.

Lower bounds have been developed for many VRP variants. Many of these are based on cutting planes. See [22] and [30] for state-of-the-art lower bounds for the CVRP. Also for the General Routing Problem, there is a tradition of obtaining lower bounds through algorithms involving cutting planes. See [14], [15], and [16] for some of the best lower bound algorithms for this problem.

For the CARP, the academic tradition has been to develop combinatorial lower bounds. Such lower bounds are based on the theory from combinatorial optimization rather than on linear programming. The majority of these bounds are based on the construction of one or several matchings. The best such lower bound is the Multiple Cuts Node Duplication Lower Bound (MCNDLB), [36], with the extensions added in [4]. Good lower bounds based on other strategies are the Hierarchical Relaxations Lower Bound, [5], and LP-based bounds, [9, 31]. Recent exact algorithms using strong lower bounding procedures are found in [8, 13]. See [4] for an overview of CARP lower bounds and [37] for a recent survey on CARP in general.

The main contribution of this paper is to provide the first (to the best of our knowledge) lower bound procedure for the NEARP. This bound is inspired by the MCNDLB for CARP and its extensions. We also define two new sets of test instances that complement the grid-based CBMix instances of Prins and Bouchenoua. The first set is called the BHW benchmark. It is based on 20 well-known CARP instances from the literature. The second is called the DI-NEARP benchmark, and consists of 24 instances defined from real cases of newspaper delivery routing. For all test instances, we provide numerical results in the form of lower and best known upper bound.

The remainder of this paper is organized as follows. In Section 2, we formally state the Node, Edge, and Arc Routing Problem and in Section 3, we describe our lower bound algorithm for the problem and argue its correctness. In Section 4, we present two new benchmarks for the NEARP, and in Section 5 we give computational results. Finally, in Section 6, we offer a summary, our concluding remarks, and future lines of work.

2 The Node, Edge, and Arc Routing Problem

The Node, Edge, and Arc Routing Problem (NEARP) is defined on a connected multi-graph $G = (N, E, A)$, where N is the set of nodes, E is the set of undirected edges, and A is the set of directed arcs. Let c_e denote the non-negative traversal cost for $e \in E \cup A$, also known as deadheading cost, i.e., the cost for traversing the edge/arc without servicing it. The traversal cost is zero for nodes. Let $N_R \subseteq N$ be the set of required nodes, and let q_i denote the demand and p_i the processing cost of node $i \in N_R$. Similarly, let E_R and A_R be the set of required edges and arcs, respectively, and let q_e and p_e denote the demand and processing cost of $e \in E_R \cup A_R$, respectively. The processing cost is the total cost that accrues when the required edge or arc is serviced. It is the sum of the traversal and servicing costs. To follow the convention of [34], we only report the total traversal cost of a solution.

A fleet of identical vehicles each with capacity Q is initially located in a special depot node, here denoted node 1. It is assumed that the size of the fleet is unbounded.

The goal is to identify a number of tours for the vehicles such that 1) every node $i \in N_R$, every edge $e \in E_R$, and every arc $e \in A_R$ is serviced by exactly one vehicle, 2) the sum of demands serviced by each vehicle does not exceed Q, and 3) the total cost of the tours is minimized.

The total servicing cost for any feasible solution to a given NEARP is constant. Hence, we do not need to consider servicing costs in our lower bound procedure. Also, the convention for reporting results on the CBMix benchmark is such that the constant sum of servicing costs has been subtracted. We introduce the concepts of servicing and processing cost here to be compatible with Prins and Bouchenoua, and to prepare for extensions to the NEARP with temporal constraints. Moreover, some heuristics, such as greedy construction heuristics, may use servicing costs.

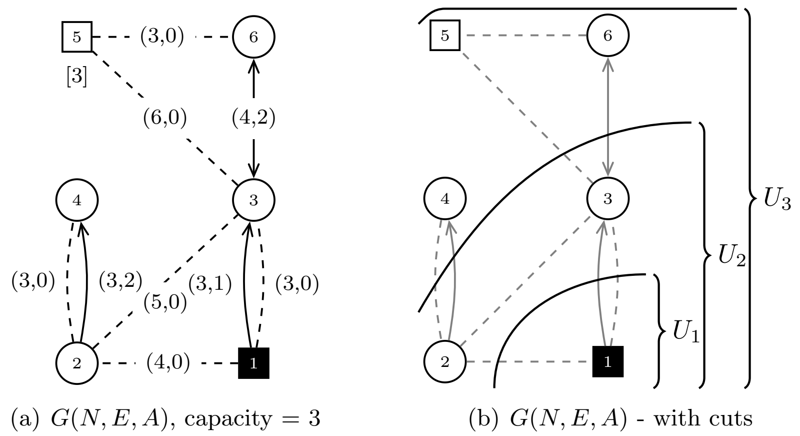
3 Lower Bound for NEARP

The algorithm is a further development of the Multiple Cuts Node Duplication Lower Bound (MCNDLB) for the CARP, [36]. We start by giving an intuitive description of the structure of the algorithm followed by a small example, and then provide a formal description.

For notational reasons, in the description of the algorithm we will assume that the graph is simple, i.e., that there is at most one required link between any pair of nodes. For $e = (i, j) \in E \cup A$, we use the notation c_{ij}, q_{ij}, and p_{ij} to denote traversal cost, demand and processing cost, respectively. The algorithm can, however, easily be extended to the non-simple case.

In the following we use $SPL(i, j)$ to denote the cost of a shortest path in G from i to j. Let $N' \subset N$ be a subset of the nodes. We define $\delta^-(N') = \{e = (i, j) \in E \cup A | i \in N \setminus N' \text{ and } j \in N'\}$ to be the set of links entering N' and $\delta^+(N') = \{e = (i, j) \in E \cup A | i \in N' \text{ and } j \in N \setminus N'\}$ to be the set of links leaving N'. Note that due to the existence of undirected edges, $\delta^-(N')$ and $\delta^+(N')$ are not necessarily disjoint. Finally, we define $\delta(N') = \delta^-(N') \cup \delta^+(N')$ to be the set of links connecting N' to the remaining graph. Finally, for any set of nodes, U, we use $G(U)$ to denote the graph induced by U.

Figure 1: NEARP example



(a) $G(N, E, A)$, capacity = 3 (b) $G(N, E, A)$ - with cuts

Node 1 is the depot node, a circle represents a node without demand and a square is a node with demand, the quantity of the demand is given in brackets. The lines are dashed for non-demand edges/arcs and solid for those with demand. The direction of the demand arcs are given by the arrows. The deadheading cost and demand quantity is given in parenthesis for all edges/arcs

Starting with $U_1 = \{1\}$, we consider mutually disjoint cuts $(U_k\,,\, N \setminus U_k)$ such that $U_1 \subset U_2 \subset \ldots \subset U_k \subset U_{k+1}$. For each such cut, U_k, the graph induced by $N \setminus U_k$ will consist of one or more connected components, $G'_s = (N'_s, E'_s, A'_s), s = 1, \ldots, t$, as illustrated in Figure 1(b). The number of vehicles needed to service the demand in G'_s, and the demand of links connecting G'_s to U_k can be estimated by $p_s = \lceil (\sum_{i \in N'_s} q_i + \sum_{(i,j) \in E'_s \cup A'_s \cup \delta(N'_s)} q_{ij})/Q \rceil$, which is a simple lower bound for the bin-packing problem.

Ideally, each vehicle would service the demand of an edge or arc when entering G'_s and when leaving G'_s. When this is not the case, we say that the vehicle is using a deadheading link. Such links can be either links without demand or links with demand not currently being serviced. We estimate the number of deadheading links (entering arcs, leaving arcs, and undirected edges) needed for all vehicles to both enter and leave G'_s. With this, we can estimate the cost of servicing demand in G'_s and demand of links connecting G'_s to U_k by constructing a node duplicated network and letting m_s be the value of a minimum cost perfect matching in this network. We do this for all the connected components and hence, $L = \sum_{s=1}^{t} m_s$ estimates the cost of servicing everything outside $G(U_k)$.

To estimate the cost of servicing demand in $G(U_k)$, we use the minimum cost $\overline{c}\,_s^u$ of a link between U and each component, G'_s and multiply this by the number of deadheading links needed to connect the two: r_s^u. Iterating over all the mutually disjoint cuts and all the connected components of these, we can estimate the cost of servicing the demand in $G(U_k)$ as $L1 = \sum_{j=1}^{k-1} \sum_{s=1}^{t} \overline{c}\,_s^u r_s^u$.

For each of these cuts, $L + L1$ is a lower bound on the cost, and the algorithm selects the cut with the highest value.

Note that in the algorithm, the calculations become more complex than outlined above due to the existence of both directed and undirected links.

3.1 Lower bound - Example

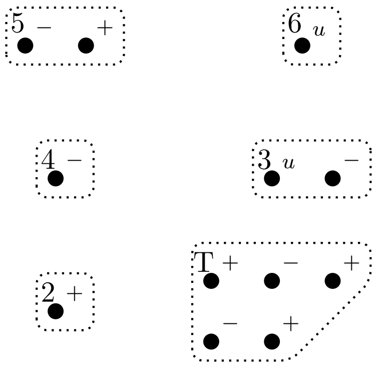
In this section we provide an example of the algorithm applied to a simple NEARP instance, presented in Figure 1. The algorithm performs a main iteration for each cut, see Figure 1(b). Thus we start with U_1 in the first iteration, then U_2. We do not consider U_3 because it is the full graph, which is also the stopping criterion.

With $U_1 = \{1\}$ there is only one connected component. The node duplicated network is shown in Figure 2. It is constructed by making a copy of a node every time it is incident to a demand edge/arc. If the demand is outgoing, it gets a positive polarity; if it is incoming it gets a negative polarity; and if it is an edge it gets a neutral charge. The demand $2 \rightarrow 4$ results in a copy of node 2 with a positive charge and a copy of node 4 with a negative charge. If a node has a demand but is not incident to any nodes, e.g., node 5, two copies of that node are made, i.e., one positive and one negative.

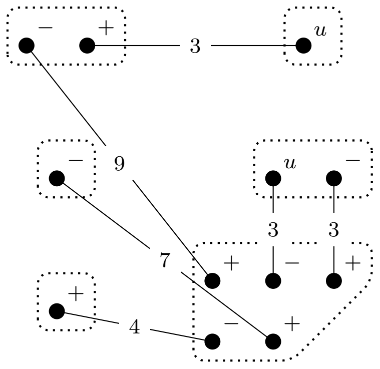
The set T in the node duplicated network represents copies of the set U, the number of nodes herein is determined by p_s. For each vehicle we must enter and leave the area T once, thus we add a negative and a positive charged node. For each demand entering T from the outside we remove a positive node and vice versa if we leave T. If this is an edge we turn a positive node into a neutral node and delete a negative node.

Based on the node duplicated network illustrated in Figure 2(a), we make a complete graph and add cost to each edge. In short, the cost is the shortest path given by Figure 1, with some exceptions, among others that a negative

Figure 2: NEARP - Node duplicated network (U_1)



(a) Node duplication at U_1



(b) Perfect matching, $m_1 = 29$

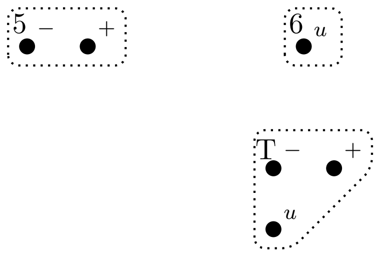
node cannot be matched to another negative node, the same applies to the positive nodes, where neutral nodes can be matched to other neutrals, positives, or negatives. A node cannot be matched to the node opposite the demand that created the node. Nodes in T cannot be matched to each other.

We now perform a minimum cost perfect matching. The result is illustrated in Figure 2(b). The cost of the matching is $L = m_1 = 29$. From Figure 1 we know that servicing the required arcs/edges will cost us 10. We can then calculate the first iteration in the lower bound; $L2 = 29 + 10 = 39$. Finally, we update $L1$ for the use in the next iteration as $L1 = \overline{c}_1^- \cdot r_1^- + \overline{c}_1^+ \cdot r_1^+ + \overline{c}_1^u \cdot r_1^u = 3 \cdot 2 + 3 \cdot 3 + 3 \cdot 0 = 15$.

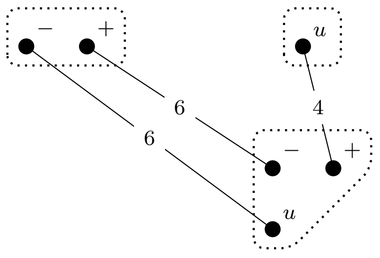
In the next iteration, we apply the second cut of Figure 1 (b) and we now use U_2 represented by the set T. $G(N \backslash U_2)$ consists of two components with $N_1' = \{5, 6\}$ and $N_2' = \{4\}$. The node duplicated network for $G(N_1)$ and for $G(N_2)$ is shown in Figures 3(a) and 3(c), respectively. The corresponding minimum cost perfect matchings are shown in Figures 3(b) and 3(d), respectively. The cost of the matchings is $L = m_1 + m_2 = 16 + 3 = 19$. Thus the lower bound of the second iteration is $L2 = \max\{39, 19 + 15 + 10\} = 44$.

As $U_3 = N$ we terminate the algorithm and return the result from the second iteration (44) as the best lower bound. The optimal solution to the NEARP instance in Figure 1 is 46 which gives a gap of $\frac{46-44}{0.5\cdot(44+46)} = 4.44\%$.

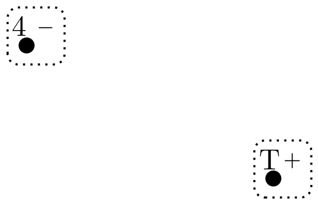
Figure 3: NEARP - Node duplicated network from U_2



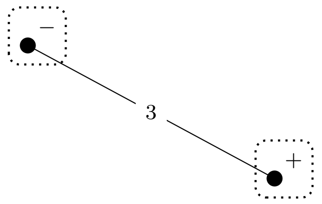
(a) Node duplication for $G(N'_1)$



(b) Perfect matching, $m_1 = 16$



(c) Node duplication for $G(N'_2)$



(d) Perfect matching, $m_2 = 3$

3.2 Lower bound - Algorithm

A detailed pseudo-code for the Multiple Cuts Node Duplicated Lower Bound for NEARP is given in Algorithm 1. The main part of the algorithm lies in the construction of the matching network $G_s^M(N_s^M, E_s^M)$ in line 22. Algorithm 2 performs this task.

We let the node set N_s^M consist of three disjoint sets, S, T, and X, where S consists of copies of nodes from N'_s, T consists of copies of nodes in U, and X can be considered to be extra copies of nodes in U and will be added later.

In line 2 of the algorithm, we first consider the degree information of nodes $i \in N$: Let $D^-(R,i)$ be the number of required arcs entering node i, $D^+(R,i)$ the number of required arcs leaving node i, let $D^u(R,i)$ be the number of required edges incident to node i, and let $D(R,i)$ be the total number of required edges and arcs incident to node i.

In line 8, for each node i in N'_s, we add $D(R,i)$ nodes to S and call these nodes the family of i, denoted by $\chi(i)$. We say that the nodes in $\chi(i)$ are copies of i. Given a node j in $\chi(i)$, we refer to i as the origin of j, denoted by $\omega(j)$. In line 15, we call Algorithm 3, which partitions S into three disjoint

Algorithm 1 MCNDLB algorithm

```
1: **initialize** $U = \{1\},\ L = 0,\ L1 = 0,\ L2 = 0$
2: **while** $U \neq N$ **do**
3:    $N' = N \setminus U$
4:    $G'(N')$ //$G'$ *is the graph induced by* $N'$
5:    $t =$ number of connected components of $G'$
6:    $G'_s = (N'_s, E'_s, A'_s)$, $1 \leq s \leq t$ //*Each component is denoted by* $G'_s$
7:    **for** $s = 1$ to $t$ **do**
8:       //*Number of vehicles needed to service the demand of nodes, edges,*
         *and arcs in* $G'_s$ *and* $\delta(N'_s)$
9:       $p_s = \lceil (\sum_{i \in N'_s} q_i + \sum_{(i,j) \in E'_s \cup A'_s \cup \delta(N'_s)} q_{ij})/Q \rceil$
10:      //*Number of required edges and arcs in cutset*
11:      $\psi_s^u = |\{(i,j) \in \delta(N'_s) \cap E_R\}|$
12:      $\psi_s^- = |\{(i,j) \in \delta^-(N'_s) \cap A_R\}|$
13:      $\psi_s^+ = |\{(i,j) \in \delta^+(N'_s) \cap A_R\}|$
14:      //*Number of deadheading edges and arcs needed in cutset*
15:      $r_s^- = \max\{0, p_s - (\psi_s^- + \psi_s^u)\}$
16:      $r_s^+ = \max\{0, p_s - (\psi_s^+ + \psi_s^u)\}$
17:      $r_s^u = \max\{0, 2p_s - (\psi_s^u + \psi_s^- + \psi_s^+ + r_s^- + r_s^+)\}$
18:      //*Minimum cost of edges and arcs in cutset*
19:      $\bar{c}_s^{\,-} = \min_{(i,j) \in \delta^-(N'_s)} c_{ij}$
20:      $\bar{c}_s^{\,+} = \min_{(i,j) \in \delta^+(N'_s)} c_{ij}$
21:      $\bar{c}_s^{\,u} = \min_{(i,j) \in \delta(N'_s)} c_{ij}$
22:      $G_s^M =$ constructNodeDuplicatedNetwork$(G, G'_s)$ //*See Algorithm 2*
23:      $m_s =$ value of minimum cost perfect matching in $G_s^M$
24:   **end for**
25:   $L = \sum_{s=1}^{t} m_s$
26:   $L2 = \max\{L2\,,\ L + L1 + \sum_{(i,j) \in E_R \cup A_R} c_{ij}\}$
27:   $L1 = L1 + \sum_{s=1}^{t} (r_s^u \cdot \bar{c}_s^{\,u} + r_s^+ \cdot \bar{c}_s^{\,+} + r_s^- \cdot \bar{c}_s^{\,-})$
28:   $U' = \{i \in N : i$ is adjacent to a vertex in $U\}$
29:   $U = U \cup U'$
30: **end while**
31: **return** $L2$
```

Algorithm 2 constructNodeDuplicatedNetwork(G, G'_s)

```
1: initialize $S = \emptyset$, $T = \emptyset$, and $X = \emptyset$
2: for all $i \in N$ do
3:     $D^-(R,i) = |\{(j,i) \in A_R\}|$ //Required arcs entering node i
4:     $D^+(R,i) = |\{(i,j) \in A_R\}|$ //Required arcs leaving node i
5:     $D^u(R,i) = |\{(i,j) \in E_R\}|$ //Required edges incident to node i
6:     $D(R,i) = D^-(R,i) + D^+(R,i) + D^u(R,i)$
7: end for
8: for all $i \in N'_s$ do //Populate S
9:     for $n = 1$ to $D(R,i)$ do
10:        Add new node $j$ to $G^M_s$
11:        $S = S \cup j$
12:        $\chi(i) = \chi(i) \cup j$ //These nodes are the family of i
13:        $\omega(j) = i$ //i is the origin node of j, denoted by $\omega(j)$
14:    end for
15:    assignDirectionS($\gamma^-, \gamma^+, \gamma^u, D^-(R,i), D^+(R,i), D^u(R,i)$)
16: end for
17: for all $i \in N'_s \mid D(R,i) = 0$ do //Required nodes not incident to required
    arcs or edges
18:    Add new nodes $j$ and $k$ to $G^M_s$
19:    $S = S \cup j \cup k$
20:    $\chi(i) = \chi(i) \cup j \cup k$ //These nodes are the family of i
21:    $\omega(j) = i$, $\omega(k) = i$
22:    $\gamma^- = \gamma^- \cup j$, $\gamma^+ = \gamma^+ \cup k$
23: end for
24: for $n = 1$ to $2p_s - |\delta(N'_s) \cap \{E_R \cup A_R\}|$ do //Populate T
25:    Add new node $j$ to $G^M_s$
26:    $T = T \cup j$
27: end for
28: assignDirectionT($\tau^-, \tau^+, \tau^u, r^-_s, r^+_s, r^u_s$)
29: for $n = 1$ to cardinalityX($S, T$) do //Populate X
30:    Add new node $j$ to $G^M_s$
31:    $X = X \cup j$
32: end for
33: Make $G^M_s$ a complete undirected graph
34: assignDemands($G^M_s, G'_s, G, \chi(), q, \gamma^-, \gamma^+, \gamma^u$)
35: for all $(i,j) \in E^M_s$ do //Set cost of edges
36:    Set $c_{ij}$ //According to equation 1
37: end for
38: return $G^M_s$
```

Algorithm 3 assignDirectionS($\gamma^-, \gamma^+, \gamma^u, D^-(R,i), D^+(R,i), D^u(R,i)$)

```
1: for $n = 1$ to $D^-(R,i)$ do
2:     $\gamma^- = \gamma^- \cup j \mid j \in \chi(i) \setminus (\gamma^-)$
3: end for
4: for $n = 1$ to $D^+(R,i)$ do
5:     $\gamma^+ = \gamma^+ \cup j \mid j \in \chi(i) \setminus (\gamma^- \cup \gamma^+)$
6: end for
7: for $n = 1$ to $D^u(R,i)$ do
8:     $\gamma^u = \gamma^u \cup j \mid j \in \chi(i) \setminus (\gamma^- \cup \gamma^+ \cup \gamma^u)$
9: end for
```

Algorithm 4 assignDirectionT($\tau^-, \tau^+, \tau^u, r_s^-, r_s^+, r_s^u$)

```
1: for $n = 1$ to $r_s^-$ do
2:     $\tau^- = \tau^- \cup j \mid j \in T \setminus (\tau^-)$
3: end for
4: for $n = 1$ to $r_s^+$ do
5:     $\tau^+ = \tau^+ \cup j \mid j \in T \setminus (\tau^- \cup \tau^+)$
6: end for
7: for $n = 1$ to $r_s^u$ do
8:     $\tau^u = \tau^u \cup j \mid j \in T \setminus (\tau^- \cup \tau^+ \cup \tau^u)$
9: end for
```

subsets γ^-, γ^+ and γ^u. For each node i in N'_s, we consider the family $\chi(i)$ consisting of $D(R,i) = D^-(R,i) + D^+(R,i) + D^u(R,i)$ nodes. Of these, we associate $D^-(R,i)$ nodes with γ^-, $D^+(R,i)$ with γ^+, and $D^u(R,i)$ with γ^u.

In line 17, we consider the nodes in $N'_s \cap N_R$ for which $D(R,i) = 0$, i.e., required nodes without incident required arcs or edges. Note that these nodes were not considered above. For each such node, i, we add two nodes to S and call these the family of i, denoted by $\chi(i)$. We add one of the nodes to γ^- and the other to γ^+. As above, we call these nodes copies of i and for a node j in $\chi(i)$, we say that i is the origin of j, denoted by $\omega(j)$.

The set T consists of $2p_s - |\delta(N'_s) \cap \{E_R \cup A_R\}|$ nodes, which can be considered copies of nodes in U. Because we know the minimum number of deadheading edges needed in U, we can partition T into three disjoint subsets τ^-, τ^+ and τ^u, where the values of r_s^-, r_s^+, and r_s^u determine the number of nodes in each of the three subsets, respectively. This is handled in lines 24 through 27.

Algorithm 5 assignDemands($G_s^M, G'_s, G, \chi(), q, \gamma^-, \gamma^+, \gamma^u$)

```
1: for all $(i,j) \in E'_s \cap E_R \mid i,j \in S$ do //Assign demand of required edges
2:     $q_{kl} = q_{ij} \mid k \in \chi(i) \cap \gamma^u$ and $l \in \chi(j) \cap \gamma^u$ //Each k or l can only be
       assigned 1 demand
3: end for
4: for all $(i \to j) \in A'_s \cap A_R$ do //Assign demand of required arcs
5:     $q_{kl} = q_{ij} \mid k \in \chi(i) \cap \gamma^+$ and $l \in \chi(j) \cap \gamma^-$ //Each k or l can only be
       assigned 1 demand
6: end for
7: for all $i \in N'_s \cap N_R \mid D(R,i) = 0$ do //Assign demand of required nodes
8:     $q_{kl} = q_i \mid k \in \chi(i) \cap \gamma^+$ and $l \in \chi(i) \cap \gamma^-$ //Each k or l can only be
       assigned 1 demand
9: end for
```

Algorithm 6 cardinalityX(S, T)

```
1: if $|S| - |T| > 0$ then
2:     $x = |S| - |T|$
3: else if $|S| - |T| \bmod 2 <> 0$ then
4:     $x = 1$
5: else
6:     $x = 0$
7: end if
8: return $x$
```

To finalize the construction of G_s^M, let the number of nodes in X be determined by Algorithm 6, which is called from Algorithm 2 in line 29. X can be considered to be extra copies of nodes in U. Nodes in X are not connected to nodes in T. There are now enough nodes in $T \cup X$ for every node in S to be matched to one of these at cost $m^-(i)$ or $m^+(i)$. X is necessary because for any two nodes, i and j in S, it might be cheaper to match both i and j to something in U instead of matching them to each other, illustrating the vehicle driving back to subgraph U and then returning to S. Note that although the triangle inequality may not apply in the original graph, it does apply in this matching network as long as no edges with cost infinity are involved.

The demand of required arcs and edges in G'_s is assigned to edges in G_s^M as explained in Algorithm 5 which is called from Algorithm 2 in line 34. These assignments are done in such a way that no node in N_s^M is chosen more than once and no demand in G'_s is assigned more than once. All other edges in E^M_s have zero demand.

Consider the s'th component, represented by the graph $G'_s = (N'_s, E'_s, A'_s)$. For every node i in N'_s set $m^-(i) = \min_{u \in U} SPL(u,i)$ and similarly $m^+(i) = \min_{u \in U} SPL(i,u)$, i.e., $m^-(i)$ and $m^+(i)$ are the lengths of a shortest path from any node in U to i and from i to any node in U, respectively.

In lines 35 through 37, the costs of edges (i,j) in E^M_s are set by c_{ij} in equation 1.

$$c_{ij} = \begin{cases} \infty & \text{if } q_{ij} > 0 \\ \\ 0 & \text{if } i,j \in S \text{ and } \omega(i) = \omega(j) \text{ and } i \in \gamma^- \text{ and } j \notin \gamma^- \\ 0 & \text{if } i,j \in S \text{ and } \omega(i) = \omega(j) \text{ and } i \in \gamma^+ \text{ and } j \notin \gamma^+ \\ 0 & \text{if } i,j \in S \text{ and } \omega(i) = \omega(j) \text{ and } i \in \gamma^u \\ \\ \infty & \text{if } i,j \in S \text{ and } \omega(i) \neq \omega(j) \text{ and } i \in \gamma^- \text{ and } j \in \gamma^- \\ SPL(i,j) & \text{if } i,j \in S \text{ and } \omega(i) \neq \omega(j) \text{ and } i \in \gamma^- \text{ and } j \notin \gamma^- \\ \infty & \text{if } i,j \in S \text{ and } \omega(i) \neq \omega(j) \text{ and } i \in \gamma^+ \text{ and } j \in \gamma^+ \\ SPL(j,i) & \text{if } i,j \in S \text{ and } \omega(i) \neq \omega(j) \text{ and } i \in \gamma^+ \text{ and } j \notin \gamma^+ \\ SPL(j,i) & \text{if } i,j \in S \text{ and } \omega(i) \neq \omega(j) \text{ and } i \in \gamma^u \text{ and } j \in \gamma^- \\ SPL(i,j) & \text{if } i,j \in S \text{ and } \omega(i) \neq \omega(j) \text{ and } i \in \gamma^u \text{ and } j \in \gamma^+ \\ \min\{SPL(i,j), SPL(j,i)\} & \text{if } i,j \in S \text{ and } \omega(i) \neq \omega(j) \text{ and } i \in \gamma^u \text{ and } j \in \gamma^u \\ \\ \infty & \text{if } i,j \in T \\ \infty & \text{if } i \in S \cap \gamma^+ \text{ and } j \in T \cap \tau^+ \\ m^-(i) & \text{if } i \in S \cap \gamma^+ \text{ and } j \in T \setminus \tau^+ \\ \infty & \text{if } i \in S \cap \gamma^- \text{ and } j \in T \cap \tau^- \\ m^+(i) & \text{if } i \in S \cap \gamma^- \text{ and } j \in T \setminus \tau^- \\ \min\{m^-(i), m^+(i)\} & \text{if } i \in S \cap \gamma^u \text{ and } j \in T \\ \infty & \text{if } i \in T \cap \tau^+ \text{ and } j \in S \cap \gamma^+ \\ m^-(j) & \text{if } i \in T \setminus \tau^+ \text{ and } j \in S \cap \gamma^+ \\ \infty & \text{if } i \in T \cap \tau^- \text{ and } j \in S \cap \gamma^- \\ m^+(j) & \text{if } i \in T \setminus \tau^- \text{ and } j \in S \cap \gamma^- \\ \min\{m^-(j), m^+(j)\} & \text{if } i \in T \text{ and } j \in S \cap \gamma^u \\ \\ 0 & \text{if } i,j \in X \\ m^-(i) & \text{if } i \in S \cap \gamma^+ \text{ and } j \in X \\ m^+(i) & \text{if } i \in S \cap \gamma^- \text{ and } j \in X \\ \min\{m^-(i), m^+(i)\} & \text{if } i \in S \cap \gamma^u \text{ and } j \in X \\ \infty & \text{if } i \in T \text{ and } j \in X \\ \infty & \text{if } i \in X \text{ and } j \in T \end{cases} \quad (1)$$

In order to tighten the bound, consider every pair of demand edges, (i,j) and (k,l) in E^M_s. If $q_{ij} + q_{kl} > Q$, we set the cost of the edges $(i,k), (i,l), (j,k)$, and (j,l) to ∞, since (i,j) and (k,l) cannot be serviced on the same tour. For the sake of simplicity, the example given in Section 3.1 does not include this part.

3.3 Correctness of the Lower Bound

Since the bound is an extension of the MCNDLB for the CARP, which was proven to be valid in [36], we focus on the changes that are made to the original algorithm.

The first change occurs in the calculation of p_s, i.e., the number of vehicles needed to service component s and the links connecting it to U in Algorithm 1 line 9. In the original algorithm the demand was summed over all demand edges. Because, in the NEARP, we have both required edges, arcs, and nodes, clearly the summation should be over all of these.

In Algorithm 1 lines 11 through 13, we calculate the required number of deadheading links. In the original algorithm, this was calculated as $r_s = \max\{0\,,\, 2p_s - q_s\}$. Needing at least p_s vehicles, each of which must both enter and leave the component, and having r_s required edges in the cut, this is clearly correct. For the NEARP, we first consider entering vehicles. Note that we must have at least p_s of these. We have ψ_s^- entering arcs and ψ_s^u edges in the cut. Hence, up to $\psi_s^- + \psi_s^u$ vehicles can use these existing links and we need to construct $\max\{0\,,\, p_s - (\psi_s^- + \psi_s^u)\}$ deadheading entering arcs. The argumentation for arcs leaving the component is symmetrical.

Needing at least $2p_s$ links in total, we now add the number of undirected edges corresponding to the difference between $2p_s$ and the sum of all required links (arcs and edges) and the number of deadheading arcs added to the network. Thereby the correctness of lines 11 through 13 has been shown.

With these definitions in place, it follows directly that the estimate for servicing everything inside U, $L1$, in Algorithm 1 lines 15 through 17 is correctly generalized to the NEARP.

It only remains to show that the construction of the matching network in Algorithm 2 leads to a valid estimate for servicing G'_s and the cutset. The structure of the matching network is similar to the one in the original bound. For each original node, we add $D(R, i)$ nodes to S in the matching network. This is exactly the number of times we must either enter or leave the node due to arc and edge requirements. Clearly, we may partition these into nodes that represent entering, leaving, and undirected demand. We use the same number of nodes in the sets T and X as in the original algorithm, but again, for the set T, we can partition the nodes into sets based on the knowledge described above. For required nodes with no adjacent required links, it is clearly legal to add two nodes to S - one for entering and one for leaving.

The assignment of required edges is done precisely as in the original algorithm, except that now we take the direction of arcs into account when selecting the nodes in each family to be matched. Furthermore, for required nodes we legally select the edge between the two copies of the original node to absorb the demand. As in the original algorithm, the cost of all these demand-assigned edges is set to infinity to prevent them from being used in the matching.

The remaining cost structure is far more complex in this algorithm than in the original one. This is due to the partitioning of families into entering, leaving and undirected sets. When two nodes are in the same family, the cost of the edge connecting them is zero if it is possible to enter through one of the copies and leave through the other. Otherwise, the cost is set to infinity, to prevent this connection from being used in the matching.

For two nodes in different families, the cost is also infinite if either both nodes are entering nodes or both are leaving nodes, as this combination is illegal. If the combination is legal, the cost between such nodes corresponds to the cost of a shortest path between the origins of the nodes, while taking possibility of directions into account.

When considering a node i in S and a node j in T or in X, we use the different estimates of $m(i)$ as in the original algorithm, except that again, we need to take into account the different combinations of entering and leaving, making the expression less pretty. Connections internal to T and X are handled as in the original algorithm.

As can be concluded from the above argumentation, the algorithm presented in this paper indeed yields a feasible lower bound for the NEARP.

3.4 Lower bound - Extensions

To strengthen the quality of the bound, for each of the nodes of U' on line 28, the node is added to U and we skip back to line 2. When we reach line 28 again the node is once more removed from U before the next node is added and we redo the procedure. We only move to line 29 after all nodes of U' have been examined. This can strengthen the quality of the bound in that part of the matching. This procedure has been proven valid for similar lower bound procedures in [4, 12]. When testing the addition of nodes from U' to U, the number of nodes added jointly as well as their combination influences the quality of the bound. Unfortunately, the best number and best combination cannot easily be predicted beforehand. We have chosen to add the nodes individually.

4 New NEARP benchmarks

Only one set of test instances exists for the NEARP: the CBMix instances [34]. These instances are all based on graphs with a grid structure. To ensure more diversity in the test platform for future algorithm developments and for testing the lower bound algorithm described above, we present two new benchmarks. The first, called BHW, is based on classical CARP instances from the literature. The second, called DI-NEARP, is based on real-life instances of an industrial application. We adopt the convention for the CBMix benchmark, i.e., only the total traversal cost of a solution is reported. The instance definition files and numerical results for all three benchmarks are found at SINTEF's NEARP website [2].

4.1 The BHW instances

This benchmark is generated from benchmark instances for the CARP. More specifically, we have used a selection of instances from the Gdb [6], Val [10], and Eglese [20] benchmarks.

For each instance, we have kept the underlying graph structure, the existing demand, and the vehicle capacity. We have made the following modifications to the instances: Some undirected edges have been replaced by directed arcs. If the edge was required, the demand is transmitted to the arc. Other undirected edges have been replaced by two directed arcs, one in each direction. If the edge was required, the demand is either transferred to one of the arcs or both arcs have been made required, each with a demand equal to the demand of the edge. Finally, some edges have been left unchanged. Furthermore, some of the nodes are made required.

Table 1 gives the most important properties for each instance. The first column states the name of the instance and the second provides a reference to the underlying CARP instance. The next three columns give the total number of nodes, undirected edges, and directed arcs in the graph. The following three columns give the same information for required entities. The next column states the vehicle capacity. Note that the vehicles are assumed to be identical. The remaining six columns provide statistical information regarding required entities. Pairwise, these columns provide the mean and standard deviation of the demand of the required nodes, edges, and arcs in the graph. Note that only the required entities are included in these calculations. All instances have relatively sparse networks, as they simulate real-life situations. The depot is located in node 1 in all BHW instances.

4.2 The DI-NEARP instances

This benchmark is defined from six real-life cases from the design of carrier routes for home delivery of subscription newspapers and other media products in the Nordic countries. The company Distribution Innovation AS (DI) [1] operates a web-based solution for design, revision, management, and control of carrier routes. Route design and revision are based on electronic road and address data provided by commercial GIS vendors. Sophisticated models for travel and service time are utilized. The Spider VRP solver provided by SINTEF [3] is integrated in the solution.

The GIS road network data may have been improved by the user through manual editing due to errors or lack of detail. All delivery points are geocoded, and the enhanced road network data are transformed into an internal graph representation in Spider. The basic node routing problem cases typically have a large number of points. Through road topology based aggregation heuristics in Spider, the original problem has been transformed to a NEARP with side constraints. The graph topology of the instances is taken directly from the Spider graph.

Data for the six instances was retrieved from the DI web server in 2011. These particular cases only have required edges and nodes, but no required arcs. The edges have symmetrical travel costs. The travel and service costs are set to the travel and service times calculated by the models in the DI system, as there is a close correlation between total route duration and the actual cost of the route plan. The index of the depot node is given explicitly.

In the industrial case, vehicle capacity is not constraining, but there is a constraint on route duration. We have transformed the duration constraint to a capacity constraint and selected four different values for capacity that produce a reasonable range for the number of routes, including the actual number from the real application. Hence, the DI-NEARP benchmark consists of 24 instances. They are named DI-NEARP-nr-Qqk, where r is the total number of required nodes, edges, and arcs and q is capacity in thousands. Table 2 gives the most important properties for each instance. The structure of this table is similar to that of Table 1 except for the second column which is not relevant for the DI-NEARP instances.

5 Computational Results

We have implemented the lower bound algorithm in two versions: one version where all nodes neighboring U are added at once, and one version where the addition of each node is tested separately before all nodes are added, as explained in Section 3.4. In this section, we report our experimental results for both implementations, while referring to the latter as *AD1.* All lower bound calculations are performed on a PC with an Intel Core 2 Duo CPU, running at 2.53 GHz and with 2GB of RAM.

The results obtained for the three benchmark sets are given in Tables 3, 4, and 5. In each table, the second column provides the best known upper bound for the instance, hereafter referred to as B^U. For the CBMix instances these are obtained from [34], [29], and [28]. For the BHW and the DI-NEARP instances, the first upper bounds were obtained with the Spider solver [28]. Remember that the cost reported is the total traversal cost.

For each of the two lower bound versions, we give the obtained value of the lower bound algorithm B^L, and the relative optimality gap G^O, i.e., the percentage gap from the best known upper bound to the lower bound, as calculated by the following formula:

$$G^O = \frac{B^U - B^L}{(B^U + B^L)/2} 100$$

Finally, we provide the running time of the lower bound algorithm in seconds. We imposed a time limit of 10,000 seconds. For the large-size DI-NEARP instances, the calculation of the AD1 lower bound was not completed within this time limit. Hence, the AD1 column has been omitted in Table 5.

For the CBMix benchmark, the gaps vary between 0.0% and 39.7% with an average of 23.1%. The variation is larger for the BHW benchmark, where the minimum, maximum, and average gaps are 0.0%, 54.0%, and 24.2%, respectively. The average gap for the large-size DI-NEARP instances is 27.8%, with a minimum of 7.0% and a maximum of 54.8%.

Optimal solutions for CBMix12 and CBMix23 were reported in [11]. For three BHW instances: BHW2, BHW4, and BHW6, the lower bound proves optimality of the best known solutions. For all other instances, the optimal solution is unknown. It is therefore not possible to determine if the size of the gap is mainly explained by the quality of the lower bound or by the quality of the best known solutions. It is, however, noted that for the CARP, the corresponding lower bound closes the gap to the optimal solution for 1/3 of the benchmark instances, [36]. Along with the fact that only few algorithms for the construction of feasible solutions have been developed for the NEARP, this lead us to believe that a large portion of the gap may be explained by the quality of the upper bound.

The quality of the lower bound can be measured in two ways. For a solution for NEARP to be feasible it must satisfy two general conditions, 1) the flow balance, which enforces that each vehicle traverses the graph such that whenever it arrives at a node it will also leave the node; 2) the packing constraints which enforce that each vehicle does not exceed its capacity Q.

The MCNDLB lower bound provides solutions where the flow balance is satisfied to some extent due to perfect matching in the first iteration which enforces the necessary number of deadheading links. However, the bound only avoids 2-cycles and the path is thus not elementary. The other discrepancy is that the cost of these deadheadings in the succeeding cuts may be underestimated. We sum the costs of the cheapest way to cross each cutset, but not the cheapest way to cross all the cutsets. Hence, the bound does not foresee if a more expensive link should have been used.

As regards the packing constraints, we believe that these contribute the most to the quality of the gap. If we assume that the capacity Q accommodates all demand, such that only one vehicle would be required, the packing constraint would not contribute to the gap. If we inspect the solutions to the DI-NEARP instances it is obvious that the gap decreases when the capacity of the vehicles increases. This is because the packing constraints become less important.

In the first iteration of the MCNDLB algorithm, there is a stronger focus on flow balance than packing. As cuts are added in subsequent iterations, emphasis is progressively shifted from flow balance towards packing.

6 Summary and Conclusion

The VRP literature has often been criticized for being based on idealized assumptions that render the proposed models inadequate for real life applications. With only a few exceptions, there has been a strict separation of node routing and arc routing problems in the literature. In [34] Prins and Bouchenoua argued that there are real-life applications that can neither be adequately modeled as pure arc routing problems, nor as pure node routing problems.

In this paper, we have reinforced the claims of Prins and Bouchenoua and argued that the NEARP represents an important, new dimension of VRP model richness. We have also argued that the tradition of modeling applications such as newspaper delivery, mail delivery, and communal waste collection as arc routing problems is problematic. For real-life, large-size instances of such applications, where demand is basically located in nodes, abstraction techniques such as aggregation of demand may be needed to provide high-quality solutions. Reasonable aggregation heuristics will typically produce instances with demand on nodes, edges, and arcs.

The main contribution of this paper is to provide (to the best of our knowledge) the first lower bound procedure for the NEARP. Also, we provide two new sets of test instances: the BHW benchmark derived from 20 well-known CARP instances, and the DI-NEARP benchmark with 24 instances derived from real-life data from carrier routing of subscription newspapers and other media products. The new benchmarks complement the grid-based CBMix benchmark proposed by Prins and Bouchenoua, for which two other papers also provide upper bounds. For the BHW and DI-NEARP benchmarks, the first upper bounds have been produced by Hasle et al. [28], so we now have lower and upper bounds for all test instances. For three instances, the gaps have been closed. The average gap is rather large: 25.1%. The lower bound procedure is based on a high quality lower bound for the CARP. This fact, and the limited amount of work on approximate methods for the NEARP, give us reason to believe that the main reason for the large gaps is the quality of the upper bound.

The NEARP is a theoretically interesting problem with high industrial relevance. The numerical results presented here should motivate the research community to develop better heuristics and exact algorithms that take advantage of the structure of this important problem. Moreover, NEARP extensions should be proposed on the basis of industrial aspects.

7 Acknowledgments

The authors would like to thank the company Distribution Innovation AS for giving us access to real-life case data from their newspaper delivery routing system. The work presented here has been funded by the Research Council of Norway as a part of the Effekt project (contract number 187293/I40, SMARTRANS), and the DOMinant II project (contract number 205298/V30, eVita).

References

[1] Distribution Innovation home page. `http://www.di.no/?lang=en`. Accessed: 27/07/2012.

[2] NEARP web pages. `http://www.sintef.no/NEARP`. Accessed: 27/07/2012.

[3] Spider web pages. `http://www.sintef.no/Projectweb/Transportation-planning/Software/Spider/`. Accessed: 27/07/2012.

[4] Dino Ahr. *Contributions to multiple postmen problems.* PhD thesis, University of Heidelberg, 2004.

[5] Anita Amberg and Stefan Voss. A hierarchical relaxations lower bound for the capacitated arc routing problem. *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, 3, 2002.

[6] Edward K. Baker, James S. DeArmon, and Bruce L. Golden. Computational experiments with algorithms for a class of routing problems. *Computers and Operations Research*, 10(1):47–59, 1983.

[7] Roberto Baldacci, Paolo Toth, and Daniele Vigo. Exact algorithms for routing problems under vehicle capacity constraints. *Annals of Operations Research*, 175:213–245, 2010. 10.1007/s10479-009-0650-0.

[8] E. Bartolini, J.-F. Cordeau, and G. Laporte. Improved lower bounds and exact algorithm for the capacitated arc routing problem. *Mathematical Programming*, pages 1–44, 2011.

[9] José M. Belenguer and Enrique Benavent. A cutting plane algorithm for the capacitated arc routing problem. *Computers and Operations Research*, 30(5):705–728, 2003.

[10] Enrique Benavent, Vicente Campos, Angel Corberán, and Enrique Mota. The capacitated arc routing problem: Lower bounds. *Networks*, 22:669–690, 1992.

[11] A. Bosco, D. Lagana, R. Musmanno, and F. Vocaturo. Modeling and solving the mixed capacitated general routing problem. *Optimization Letters*, 2012, forthcoming.

[12] P. Breslin and A. Keane. The Capacitated Arc Routing Problem: Lower Bounds. Master's Thesis. *University College Dublin, Department of Management Information Systems*, 1997.

[13] Bode Claudia and Stefan Irnich. Cut-first branch-and-price-second for the capacitated arc-routing problem. Technical report, LM-2011-03, Johannes Gutenberg University Mainz, Germany, 2011.

[14] Angel Corberán, Adam N. Letchford, and José M. Sanchis. A cutting plane algorithm for the general routing problem. *Mathematical Programming*, 90:291–316, 2001.

[15] Angel Corberán, Enrique Mota, and José M. Sanchis. A comparison of two different formulations for arc routing problems on mixed graphs. *Computers and Operations Research*, 33:3384–3402, 2006.

[16] Angel Corberán, Isaac Plana, and José M. Sanchis. A branch & cut algorithm for the windy general routing problem and special cases. *Networks*, 49:245–257, 2007.

[17] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 80(6), 1959.

[18] Karl Doerner and Verena Schmid. Survey: Matheuristics for rich vehicle routing problems. In Mara Blesa, Christian Blum, Gnther Raidl, Andrea Roli, and Michael Sampels, editors, *Hybrid Metaheuristics*, volume 6373 of *Lecture Notes in Computer Science*, pages 206–221. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-16054-7_15.

[19] M. Dror. *Arc routing: theory, solutions, and applications*. Kluwer Academic, 2000.

[20] Richard W. Eglese and Leon Y.O. Li. An interactive algorithm for vehicle routeing for winter-gritting. *Journal of the Operational Research Society*, 47:217–228, 1996.

[21] Dominique Feillet. A tutorial on column generation and branch-and-price for vehicle routing problems. *4OR*, 8(4):407–424, 2010.

[22] Ricardo Fukasawa, Humberto Longo, Jens Lysgaard, Marcus Poggi de Aragão, Marcelo L. Reis, and Renato Fonseca F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511, 2006.

[23] M. Gendreau, J. Y. Potvin, O. Bräysy, G. Hasle, and A. Løkketangen. Metaheuristics for the vehicle routing problem and its extensions: a categorized bibliography. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem - Latest Advances and New Challenges*. Springer Verlag, Heidelberg, 2008.

[24] B.L. Golden, S. Raghavan, and E.A. Wasil. *The vehicle routing problem: Latest advances and new challenges.* Operations Research Computer Science Interfaces Series. Springer, 2010.

[25] Bruce L. Golden and Richard T. Wong. Capacitated arc routing problems. *Networks*, 11:305–315, 1981.

[26] J. C. A. Gutiérrez, D. Soler, and Antonio Hervás. The capacitated general routing problem on mixed graphs. *Revista Investigacion Operacional*, 23(1):15 – 26, 2002.

[27] Geir Hasle and Oddvar Kloster. Industrial vehicle routing. In Geir Hasle, Knut-Andreas Lie, and Ewald Quak, editors, *Geometric Modelling, Numerical Simulation, and Optimization - Applied Mathematics at SINTEF*, pages 397–435. Springer, 2007.

[28] Geir Hasle, Oddvar Kloster, Morten Smedsrud, and Kevin Gaze. Experiments on the node, edge, and arc routing problem. Technical Report A23265, SINTEF, 2012.

[29] Hisafumi Kokubugata, Ayako Moriyama, and Hironao Kawashima. A practical solution using simulated annealing for general routing problems with nodes, edges, and arcs. In *Proceedings of the 2007 international conference on Engineering stochastic local search algorithms: designing, implementing and analyzing effective heuristics*, SLS'07, pages 136–149, Berlin, Heidelberg, 2007. Springer-Verlag.

[30] Jens Lysgaard, Adam N. Letchford, and Richard W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445, 2004.

[31] Rafael Martinelli, Marcus Poggi, and Anand Subramanian. Improved bounds for large scale capacitated arc routing problem. Technical report, Monografias em Ciencia da Computacao 14/11, Pontificia Universidade Catolica, Rio de Janeiro, Brazil, 2011.

[32] C.S. Orloff. A fundamental problem in vehicle routing. *Networks*, 4:35–64, 1974.

[33] Ram Pandit and B. Muralidharan. A capacitated general routing problem on mixed networks. *Computers & Operations Research*, 22(5):465 – 478, 1995.

[34] Christian Prins and Samir Bouchenoua. A memetic algorithm solving the VRP, the CARP and general routing problems with nodes, edges and arcs. In W Hart, N Krasnogor, and J Smith, editors, *Recent Advances in Memetic Algorithms, Studies in Fuzziness and Soft Computing*, pages 65–85. Springer, 2004.

[35] Paolo Toth and Daniele Vigo, editors. *The vehicle routing problem.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[36] Sanne Wøhlk. New lower bound for the capacitated arc routing problem. *Computers and Operations Research*, 33(12):3458–3472, 2006.

[37] Sanne Wøhlk. A decade of capacitated arc routing. In Bruce L. Golden, S. Raghavan, and Edward A. Wasil, editors, *The vehicle routing problem: Latest advances and new challenges.* Springer, 2008.

Instance	Basis Instance	Total number			Required			Capacity	Node demand		Edge demand		Arc demand	
		Nodes	Edges	Arcs	Nodes	Edges	Arcs		Mean	Std.	Mean	Std.	Mean	Std.
BHW1	GDB 1	12	11	22	7	11	11	5	1.0	0.0	1.0	0.0	1.0	0.0
BHW2	GDB 6	12	0	25	4	0	25	5	1.0	0.0	-	-	1.0	0.0
BHW3	GDB 12	13	8	30	5	8	7	35	7.2	3.5	8.0	4.1	9.0	2.3
BHW4	GDB 22	11	0	44	6	0	44	27	5.0	2.4	-	-	4.0	2.7
BHW5	Val 7a	40	0	132	30	0	132	200	8.5	4.3	-	-	8.0	3.3
BHW6	Val 7a	40	37	58	15	37	58	200	7.3	4.8	8.0	2.5	7.0	3.0
BHW7	Val 10d	50	0	194	35	0	194	75	9.1	4.6	-	-	7.0	2.3
BHW8	Val 10d	50	0	194	20	0	97	75	7.6	3.3	-	-	7.0	1.7
BHW9	Val 10d	50	26	142	10	26	142	75	8.6	3.1	6.0	1.6	7.0	2.3
BHW10	Egl e1 A	77	0	196	40	0	102	305	30.0	13.3	-	-	28.0	15.3
BHW11	Egl e1 A	77	0	196	20	0	51	305	28.3	11.1	-	-	28.0	10.8
BHW12	Egl s1 B	140	0	380	40	0	75	150	29.7	11.1	-	-	18.0	6.9
BHW13	Egl s1 B	140	0	380	25	0	150	150	29.0	11.2	-	-	18.0	9.7
BHW14	Egl e4 C	77	0	196	25	0	196	130	31.2	13.0	-	-	25.0	20.7
BHW15	Egl e4 C	77	0	196	30	0	98	130	33.0	11.9	-	-	25.0	14.6
BHW16	Egl s4 C	140	0	380	30	0	380	120	24.4	16.9	-	-	22.0	16.9
BHW17	Egl s4 C	140	0	380	50	0	190	120	22.3	7.1	-	-	22.0	11.9
BHW18	Egl e3 B	77	0	196	20	0	174	190	25.0	8.1	-	-	25.0	19.6
BHW19	Egl e3 B	77	0	196	20	0	87	190	24.1	9.0	-	-	25.0	13.8
BHW20	Egl s2 A	140	51	278	50	51	192	235	19.0	9.1	23.0	13.4	20.0	13.3

Table 1: Details on the new BHW instances.

Instance	Number of			Required			Capacity	Node demand		Edge demand		Arc demand	
	Nodes	Edges	Arcs	Nodes	Edges	Arcs		Mean	Std.	Mean	Std.	Mean	Std.
DI-NEARP-n240-Q2k	563	815	0	120	120	0	2000	34.2	45.6	78.2	82.7	-	-
DI-NEARP-n240-Q4k	563	815	0	120	120	0	4000	34.2	45.6	78.2	82.7	-	-
DI-NEARP-n240-Q8k	563	815	0	120	120	0	8000	34.2	45.6	78.2	82.7	-	-
DI-NEARP-n240-Q16k	563	815	0	120	120	0	16000	34.2	45.6	78.2	82.7	-	-
DI-NEARP-n422-Q2k	710	871	0	302	120	0	2000	24.4	39.1	74.7	102.1	-	-
DI-NEARP-n422-Q4k	710	871	0	302	120	0	4000	24.4	39.1	74.7	102.1	-	-
DI-NEARP-n422-Q8k	710	871	0	302	120	0	8000	24.4	39.1	74.7	102.1	-	-
DI-NEARP-n422-Q16k	710	871	0	302	120	0	16000	24.4	39.1	74.7	102.1	-	-
DI-NEARP-n442-Q2k	761	917	0	294	148	0	2000	26.1	31.7	69.1	46.9	-	-
DI-NEARP-n442-Q4k	761	917	0	294	148	0	4000	26.1	31.7	69.1	46.9	-	-
DI-NEARP-n442-Q8k	761	917	0	294	148	0	8000	26.1	31.7	69.1	46.9	-	-
DI-NEARP-n442-Q16k	761	917	0	294	148	0	16000	26.1	31.7	69.1	46.9	-	-
DI-NEARP-n477-Q2k	667	837	0	203	274	0	2000	16.9	35.8	68.4	73.9	-	-
DI-NEARP-n477-Q4k	667	837	0	203	274	0	4000	16.9	35.8	68.4	73.9	-	-
DI-NEARP-n477-Q8k	667	837	0	203	274	0	8000	16.9	35.8	68.4	73.9	-	-
DI-NEARP-n477-Q16k	667	837	0	203	274	0	16000	16.9	35.8	68.4	73.9	-	-
DI-NEARP-n699-Q2k	982	1103	0	335	364	0	2000	57.3	67.2	176.1	175.5	-	-
DI-NEARP-n699-Q4k	982	1103	0	335	364	0	4000	57.3	67.2	176.1	175.5	-	-
DI-NEARP-n699-Q8k	982	1103	0	335	364	0	8000	57.3	67.2	176.1	175.5	-	-
DI-NEARP-n699-Q16k	982	1103	0	335	364	0	16000	57.3	67.2	176.1	175.5	-	-
DI-NEARP-n833-Q2k	1120	1450	0	347	486	0	2000	31.9	72.7	104.2	216.9	-	-
DI-NEARP-n833-Q4k	1120	1450	0	347	486	0	4000	31.9	72.7	104.2	216.9	-	-
DI-NEARP-n833-Q8k	1120	1450	0	347	486	0	8000	31.9	72.7	104.2	216.9	-	-
DI-NEARP-n833-Q16k	1120	1450	0	347	486	0	16000	31.9	72.7	104.2	216.9	-	-

Table 2: Details on the DI-NEARP instances.

Instance	Best Known	Lower Bound			Lower Bound AD1		
	B^U	B^L	G^O (%)	CPU (s)	B^L	G^O (%)	CPU (s)
CBMix1	2589	2409	7.2	1.0	2409	7.2	3.1
CBMix2	12220	9742	22.6	76.7	9742	22.6	353.4
CBMix3	3643	3014	18.9	7.5	3014	18.9	30.6
CBMix4	7583	5302	35.4	20.9	5302	35.4	118.8
CBMix5	4531	3747	18.9	3.8	3789	17.8	13.1
CBMix6	7087	4983	34.9	16.2	5201	30.7	43.1
CBMix7	9607	7296	27.3	58.7	7296	27.3	193.6
CBMix8	10524	7956	27.8	33.4	7956	27.8	196.8
CBMix9	4038	3460	15.4	2.5	3460	15.4	7.8
CBMix10	7582	6409	16.8	37.5	6432	16.4	113.0
CBMix11	4494	2998	39.9	4.6	3031	38.9	43.9
CBMix12*	**3138**	**3138**	0.0	2.1	**3138**	0.0	12.9
CBMix13	9110	6489	33.6	19.4	6524	33.1	238.3
CBMix14	8566	5719	39.9	15.7	5731	39.7	107.5
CBMix15	8280	6270	27.6	10.9	6318	26.9	64.3
CBMix16	8886	7416	18.0	24.5	7416	18.0	172.6
CBMix17	4037	3654	10.0	1.8	3654	10.0	22.0
CBMix18	7098	6089	15.3	25.7	6089	15.3	120.9
CBMix19	16347	11065	38.5	110.5	11143	37.9	549.6
CBMix20	4844	3400	35.0	2.3	3452	33.6	15.7
CBMix21	18069	12474	36.6	61.8	12474	36.6	221.5
CBMix22	1941	1825	6.2	1.8	1825	6.2	4.8
CBMix23*	**780**	667	15.6	0.1	667	15.6	0.8

Table 3: Results obtained for the CBMix instances.

Instance	Best Known	Lower Bound			Lower Bound AD1		
	B^U	B^L	G^O (%)	CPU (s)	B^L	G^O (%)	CPU (s)
BHW1	337	324	3.9	0.3	324	3.9	1.3
BHW2*	**470**	**470**	0.0	0.4	**470**	0.0	0.9
BHW3	415	326	24.0	0.2	326	24.0	0.5
BHW4*	**240**	**240**	0.0	0.5	**240**	0.0	3.8
BHW5	506	498	1.6	5.4	502	0.8	52.1
BHW6*	**388**	**388**	0.0	2.9	**388**	0.0	32.3
BHW7	1094	930	16.2	41.7	930	16.2	347.2
BHW8	672	644	4.3	6.8	644	4.3	118.8
BHW9	913	791	14.3	28.0	791	14.3	346.5
BHW10	8556	6810	22.7	21.6	6810	22.7	123.1
BHW11	5021	3986	23.0	6.9	3986	23.0	40.0
BHW12	11042	6346	54.0	33.4	6346	54.0	207.7
BHW13	14510	8746	49.6	86.3	8746	49.6	576.7
BHW14	25194	17762	34.6	113.0	17762	34.6	737.5
BHW15	15509	12193	23.9	20.7	12193	23.9	214.2
BHW16	44527	26014	52.5	787.2	26014	52.5	4905.3
BHW17	26768	15396	53.9	162.7	15396	53.9	900.6
BHW18	15833	11202	34.3	77.9	11202	34.3	435.3
BHW19	9424	7065	28.6	14.1	7080	28.4	101.6
BHW20	16625	10730	43.1	269.9	10730	43.1	1388.4

Table 4: Results obtained for the BHW instances.

Instance	Best Known	Lower Bound		
	B^U	B^L	G^O (%)	CPU (s)
DI-NEARP-n240-Q2k	24371	16376	39.2	368
DI-NEARP-n240-Q4k	18352	14362	24.4	311
DI-NEARP-n240-Q8k	15937	13442	17.0	324
DI-NEARP-n240-Q16k	14953	13116	13.1	334
DI-NEARP-n422-Q2k	18990	11623	48.1	1571
DI-NEARP-n422-Q4k	15987	11284	34.5	1337
DI-NEARP-n422-Q8k	14627	11220	26.4	1049
DI-NEARP-n422-Q16k	14357	11198	24.7	1702
DI-NEARP-n442-Q2k	51656	35068	38.3	1689
DI-NEARP-n442-Q4k	45605	33585	30.4	1715
DI-NEARP-n442-Q8k	44652	32985	30.1	1736
DI-NEARP-n442-Q16k	42797	32713	26.7	1816
DI-NEARP-n477-Q2k	23124	19722	15.9	1572
DI-NEARP-n477-Q4k	20198	18031	11.3	1574
DI-NEARP-n477-Q8k	18561	17193	7.7	1582
DI-NEARP-n477-Q16k	18105	16873	7.0	1575
DI-NEARP-n699-Q2k	59817	34101	54.8	7249
DI-NEARP-n699-Q4k	40473	26891	40.3	6921
DI-NEARP-n699-Q6k	30992	23302	28.3	7133
DI-NEARP-n699-Q8k	27028	21967	20.7	7400
DI-NEARP-n833-Q2k	56877	32435	54.7	8239
DI-NEARP-n833-Q4k	42407	29381	36.3	8739
DI-NEARP-n833-Q8k	35267	28453	21.4	8675
DI-NEARP-n833-Q16k	33013	28233	15.6	8157

Table 5: Results obtained for the DI-NEARP instances.