New Heuristics and Lower Bounds for the Min-Max k-Chinese Postman Problem

Dino Ahr and Gerhard Reinelt

Institute for Computer Science, University of Heidelberg
Im Neuenheimer Feld 368, 69120 Heidelberg, Germany
{dino.ahr,gerhard.reinelt}@informatik.uni-heidelberg.de

Abstract. Given an undirected edge-weighted graph and a depot node, postman problems are generally concerned with traversing the edges of the graph (starting and ending at the depot node) while minimizing the distance traveled. For the Min-Max k-Chinese Postman Problem (MM k-CPP) we have $k > 1$ postmen and want to minimize the longest of the k tours. We present two new heuristics and improvement procedures for the MM k-CPP. Furthermore, we give three new lower bounds in order to assess the quality of the heuristics. Extensive computational results show that our algorithms outperform the heuristic of Frederickson et al. [12].

Keywords: Arc Routing, Chinese Postman Problem, Min-Max Optimization, Heuristics, Lower Bounds.

1 Introduction

1.1 Problem Definition and Contributions

We consider arc routing problems with multiple postmen. Given an undirected graph $G = (V, E)$, weights $w : E \to \mathbb{R}^+$ for each edge (which we usually interprete as distances), a distinguished depot node $v_1 \in V$ and a fixed number $k > 1$ of postmen, we want to find k tours where each tour starts and ends at the depot node and each edge $e \in E$ is covered by at least one tour. In contrast to the usual objective to minimize the total distance traveled by the k postmen, for the *Min-Max k-Chinese Postman Problem* (MM k-CPP) we want to minimize the length of the longest of the k tours.

This kind of objective function is preferable when the aim is to serve each customer as early as possible. Furthermore, tours will be enforced to be more balanced resulting in a "fair" scheduling of tours.

The MM k-CPP was introduced in [12] and shown to be $\mathcal{NP}$-hard by a reduction from the k-partition problem. Furthermore, in [12] a $(2 - \frac{1}{k})$-approximation algorithm for the MM k-CPP is proposed, which we call *FHK-heuristic* in the following. To the best of our knowledge these are the only results for the MM-k-CPP in the literature.

In this paper we present two new constructive heuristics for the MM k-CPP as well as improvement procedures which are used as postprocessing steps for the constructive heuristics. We implemented our new heuristics and the improvement procedures as well as the FHK-heuristic and compared them on a variety of test instances from the literature as well as on random instances. We were able to improve the results obtained by the FHK-heuristic for most instances, sometimes considerably. In order to assess the solution quality of the heuristics in general, we also derived three new lower bounds for the MM k-CPP.

1.2 Related Work

Arc routing problems of a great variety are encountered in many practical situations such as road or street maintenance, garbage collection, mail delivery, school bus routing, etc. For an excellent survey on arc routing we refer the reader to [8], [2], [10] and [11].

The *k-Chinese Postman Problem* (k-CPP) has the same set of feasible solutions as the MM k-CPP except that the objective is to minimize the total distance traveled by the k postmen. In [22] it is shown that the k-CPP is polynomially solvable by applying a lower bound algorithm for the Capacitated Arc Routing Problem from [3].

Because of its great practical relevance much attention has been paid to the *Capacitated Chinese Postman Problem* (CCPP) introduced in [4] and the *Capacitated Arc Routing Problem* (CARP) introduced in [14]. Both problems have the same input data as the MM k-CPP but in addition there are edge demands ($q : E \to \mathbb{R}^+$), which can be interpreted as the amount of load to collect or deposit along the edge, and a fixed vehicle capacity Q. A feasible set of k tours $C_1, \ldots, C_k$, each tour starting and ending at the depot node, has to cover all edges e with positive demand ($q(e) > 0$) while obeying the restricted capacity of each vehicle ($\sum_{e \in C_i} q(e) \leq Q$ for $i = 1, \ldots, k$). The objective is to find a feasible set of k tours which minimizes the total distance traveled. The CCPP is a special case of the CARP, where all edges have a positive demand. Both problems are hard to solve, in fact in [14] it is shown that even 1.5-approximation of the CCPP is $\mathcal{NP}$-hard.

The CARP is strongly related to the MM k-CPP since again tours are forced to be balanced because of the capacity constraints.

Great effort has been spent to devise heuristics for the CARP. We will mention some of them, inspiring our heuristics, in the subsequent sections. For an extensive survey of CARP heuristics we refer the reader to [19].

Recently the postoptimization tools developed in [18] for the Undirected Rural Postman Problem have been embedded into a tabu search algorithm for the CARP [17] outperforming all other heuristics. Motivated by these results, we developed improvement procedures suitable for the MM k-CPP with comparable success.

In [1] a special Min-Max Vehicle Routing Problem is considered, where the objective is also to minimize the longest tour. The difference to the MM k-CPP is that nodes have to be traversed instead of edges. In particular, the paper describes the solution process of a special instance with 4 vehicles and 120 customer nodes, issued in the scope of a mathematical contest in 1996. The fact that this instance could only be solved with a sophisticated distributed branch-and-cut based implementation, taking 10 days on a distributed network of 188 processors, impressively demonstrates the inherent difficulty of problems subjected to a min-max objective.

The paper is organized as follows. In Sect. 2 we describe the heuristics and improvement procedures we have implemented for the MM k-CPP. Sect. 3 deals with the lower bounds developed for the MM k-CPP. In the subsequent section we report computational results which show that our new algorithms perform well. Finally, we give conclusions and directions for future work.

2 Algorithms

2.1 Terminology

For the MM k-CPP we have the following input data: a connected undirected graph $G = (V, E)$, weights for each edge $w : E \to \mathbb{R}^+$, a distinguished depot node $v_1 \in V$ and a fixed number $k > 1$ of postmen. A feasible solution, called *k-postman tour*, is a set $\mathcal{C}$ of k closed walks, $\mathcal{C} = \{C_1, \ldots, C_k\}$, such that each tour C_i contains the depot node v_1, all edges $e \in E$ are covered by at least one tour C_i and each postman is involved.

We extend the weight function w to edge sets $F \subseteq E$ by defining $w(F) = \sum_{e \in F} w(e)$. Now, for a k-postman tour $\mathcal{C}$, let us denote the maximum weight attained by a single tour C_i as $w_{\max}(\mathcal{C})$, i.e.

$$w_{\max}(\mathcal{C}) = \max_{i=1,\ldots,k} w(C_i).$$

The objective of the MM k-CPP is to find a k-postman tour $\mathcal{C}^*$ which minimizes $w_{\max}$ among all feasible k-postman tours, i.e.

$$w_{\max}(\mathcal{C}^*) = \min\{w_{\max}(\mathcal{C}) \mid \mathcal{C} \text{ is a } k\text{-postman tour}\}.$$

We denote by $SP(v_i, v_j)$ the set of edges on the shortest path between nodes $v_i, v_j \in V$. The distance of the shortest path between v_i and v_j is given by $w(SP(v_i, v_j))$.

For each node set W let $E(W) = \{e \in E \mid e = \{v_i, v_j\} \text{ and } v_i, v_j \in W\}$ be the set of edges having both endnodes in W.

2.2 General Remarks

Heuristics for multiple vehicle arc routing problems can be broadly classified into three categories: *simple constructive methods*, *two-phase constructive methods* and adaptions of *meta-heuristics* resp. *improvement procedures*. The class of two-phase constructive heuristics can be further subdivided into *route first - cluster second* and *cluster first - route second* approaches.

In the next section we describe the FHK-heuristic which follows the route first - cluster second paradigm. In Sect. 2.4 and 2.5 we present our new algorithms *Augment-Merge* and *Cluster* which are a simple constructive heuristic and a cluster first - route second heuristic, respectively. In the last subsection we explain our improvement procedures used as postprocessing steps for the augment-merge and cluster heuristics.

2.3 The Frederickson-Hecht-Kim Algorithm

The strategy of the FHK-heuristic [12] is to compute a single tour (a 1-postman tour) covering all edges $e \in E$ in a first step and then to partition the tour into k parts.

Computing a 1-postman tour is the well-known *Chinese Postman Problem* (CPP), first stated by Guan [16], which is polynomially solvable by the algorithm of Edmonds and Johnson [9].

The 1-postman tour R computed in the first step will be subdivided in the following way: First, $k-1$ so called *splitting nodes* on R are determined in such a way that they mark tour segments of R approximately having the same length. Then k tours are constructed by connecting these tour segments with shortest paths to the depot node. For details see [12].

The time complexity of the algorithm is dominated by the computation of a 1-postman tour which can be accomplished in $\mathcal{O}(|V|^3)$ [9]. In [12] it is shown that this algorithm yields a $(2-\frac{1}{k})$-approximation for the MM k-CPP.

2.4 The Augment-Merge Algorithm

This new algorithm is based on the augment-merge algorithm for the CARP given in [14]. The idea of the algorithm is roughly as follows. We start with a closed walk C_e for each edge $e = \{v_i, v_j\} \in E$, which consists of the edges on the shortest path between the depot node v_1 and v_i, the edge e itself, and the edges on the shortest path between v_j and v_1, i.e. $C_e = (SP(v_1, v_i), e, SP(v_j, v_1))$. Then we successively merge two closed walks - trying to keep the tour weights low and balanced - until we arrive at k tours. In detail:

Algorithm: AUGMENT-MERGE

(1) Sort the edges e in decreasing order according to their weight $w(C_e)$.

(2) In decreasing order according to $w(C_e)$, for each $e = \{v_i, v_j\} \in E$, create the closed walk $C_e = (SP(v_1, v_i), e, SP(v_j, v_1))$, if e is not already covered by an existing tour. Let $\mathcal{C} = (C_1, \ldots, C_m)$ be the resulting set of tours. Note that the tours are sorted according to their length, i.e. $w(C_1) \geq w(C_2) \geq \ldots \geq w(C_m)$.
If $m \leq k$ we are done and have computed an optimal k-postman tour, since no tour is longer than the shortest path tour lower bound (see Sect. 3.1).
If $m < k$ we add $k - m$ "dummy" tours to $\mathcal{C}$, each consisting of twice the cheapest edge incident to the depot node.

(3) While $|\mathcal{C}| > k$ we merge tour C_{k+1} with a tour from $C_1, \ldots, C_k$ such that the weight of the merged tour is minimized.

The time complexity of the algorithm is $\mathcal{O}(|E|^2)$ since in the third step we have to merge $\mathcal{O}(|E|)$ times two tours. Merging of two tours can be accomplished in linear time in the size of the input tours which is $\mathcal{O}(|E|)$.

2.5 The Cluster Algorithm

Our second new algorithm follows the cluster first - route second approach. In the first step we divide the edge set E into k clusters and in the second step we compute a tour for each cluster.

Going back to the second step of the augment-merge algorithm, we observe that for some edges e we explicitly have to create a tour C_e and for other edges we do not, since they are contained in an already existing tour. Let us denote the first kind of edges as *critical edges* since we have to take care that they are served, whereas serving edges classified as non-critical is “for free”, since they are on some shortest path connecting a critical edge and the depot node.

Motivated by this observation the cluster step of our algorithm first performs a k-clustering of the critical edges into edge sets $F_1, \ldots, F_k$. After that, each cluster F_i is supplemented by shortest path edges connecting the contained critical edges to the depot. The routing step consists of computing the optimum 1-postman tour for each subgraph induced by F_i.

The *k-clustering step* is based on the *farthest-point clustering* algorithm of [15] and works as follows. Let F be the set of critical edges. First, we determine k *representative* edges $f_1, \ldots, f_k \in F$. Using a distance function $d : E \times E \to \mathbb{R}^+$, let $f_1 \in F$ be the edge having the maximum distance from the depot and $f_2 \in F$ the edge having maximum distance from f_1. Then the representatives $f_i \in F, i = 3, \ldots, k$ are successively determined by maximizing the minimum distance to the already existing representatives $f_1, \ldots, f_{i-1}$. The remaining critical edges g of F will be assigned to the edge set F_i which minimizes the distance between its representative f_i and g.

Algorithm: Cluster

(1) Let $\phi_{SP}(e) = 0$ for all $e \in E$. For each edge $e = \{v_i, v_j\} \in E$ increment the frequency $\phi_{SP}(g)$ by one for edges g on the shortest paths $SP(v_1, v_i)$ and $SP(v_j, v_1)$. Now the set of critical edges is $F = \{e \in E \mid \phi_{SP}(e) \leq 1\}$.

(2) Define the distance d between two edges $e = \{u, v\}, f = \{w, x\} \in F$ as

$$d(e, f) = \max\{w(SP(u, w)), w(SP(u, x)), w(SP(v, w)), w(SP(v, x))\}.$$

(3) Compute the k-clustering $F_1, \ldots, F_k$ according to the distance function d as described above.

(4) Extend edge sets F_i by adding all edges on shortest paths between the endnodes of edges contained in F_i and the depot node.

(5) Compute an optimum 1-postman tour on $G[F_i], i = 1, \ldots, k$, with the algorithm of Edmonds and Johnson [9].

The expensive parts of the cluster algorithm are the computation of an all pairs shortest path ($\mathcal{O}(|V|^3)$) needed for the distance function d, determining the frequency $\phi_{SP}(e)$ of each edge e with time complexity $\mathcal{O}(|E|^2)$ and the computation of the k optimum 1-postman tours which costs $\mathcal{O}(|V|^3)$.

2.6 Improvement Procedures

Having implemented the augment-merge and cluster heuristics and having tested them on a variety of instances, we observed that the solutions produced still leave potential for further improvements. We devised two general deterministic improvement strategies which try to reduce the weight of the longest tour in a greedy manner. These can be applied to an arbitrary k-postman tour $\mathcal{C}$ resulting in a k-postman tour $\tilde{\mathcal{C}}$ with $w_{\max}(\tilde{\mathcal{C}}) \leq w_{\max}(\mathcal{C})$.

Thus, we can use the algorithms augment-merge and cluster to produce good initial solutions of different structure and then apply the improvement procedures as a postoptimization step.

The basic idea of our first improvement procedure is to eliminate edges from the longest tour C_i if these edges are already covered by other tours. In order to easily maintain the closed walk property of C_i, we focus on those edges occurring at least twice in C_i.

Algorithm: ELIMINATE REDUNDANT EDGES

(1) Let $\phi_i(e)$ be the frequency of e occurring in C_i and $\phi(e) = \sum_{i=1}^{k} \phi_i(e)$ the frequency of e occurring in all tours. An edge e is called *multi-redundant* for C_i if $\phi_i(e) \geq 2$ and $\phi(e) - \phi_i(e) > 0$.

(2) As long as there exists a tour which contains multi-redundant edges, choose the longest tour C_i among them, and find the multi-redundant edge e of C_i, which leads to the maximum reduction of the tour length when removed. In detail:
- If $\phi_i(e)$ is even and removal of $\phi_i(e)$ times edge e *keeps* connectivity of C_i, set $n = \phi_i(e)$.
- If $\phi_i(e)$ is even and removal of $\phi_i(e)$ times edge e *destroys* connectivity of C_i, set $n = \phi_i(e) - 2$, i.e. we leave two edges for keeping connectivity.
- If $\phi_i(e)$ is odd, set $n = \phi_i(e) - 1$, i.e. we leave one edge for connectivity.

Now we choose the edge e which maximizes $n \cdot w(e)$. Remove n times edge e from C_i and update the frequencies $\phi_i(e)$ and $\phi(e)$.

In contrast to the first improvement procedure which tries to reduce the tour lengths by eliminating redundant edges totally, we now want to improve the longest tour by removing edges from it and integrate them into a shorter tour.

Algorithm: TWO EDGE EXCHANGE

(1) As long as the longest tour has been improved do the following:
(1.1) Let C_i be the tour with maximum length.

(1.2) Traverse C_i and find consecutive edges $\tilde{e}$ and $\tilde{f}$ which will achieve the maximum reduction of the length of C_i when deleted and replaced by the shortest path between appropriate endnodes of $\tilde{e}$ and $\tilde{f}$.

(1.3) Consider tours C_j with $j \neq i$ and find the tour C_j^* for which the resulting tour after adding $\tilde{e}$ and $\tilde{f}$ (and other edges required to keep the closed walk property) has minimum length.

(1.4) If the tour length of C_j^* is shorter than the original tour length of C_i then replace $\tilde{e}$ and $\tilde{f}$ by the shortest path in C_i and integrate them into C_j^*. If the tour length of C_j^* is not shorter than the original tour length of C_i then goto Step 1.2, find the next better edges $\hat{e}$ and $\hat{f}$ and proceed.

3 Lower Bounds

3.1 Shortest Path Tour Lower Bound (SPT-LB)

This lower bound is based on the observation that in an optimal solution $\mathcal{C}^*$ the length of the longest tour must have at least the length of the shortest path tour $C_{\tilde{e}} = (SP(v_1, v_i), \tilde{e}, SP(v_j, v_1))$ traversing the edge $\tilde{e} = \{v_i, v_j\} \in E$ farthest away from the depot. Since the number of postmen is not taken into account this bound will only produce good results for instances where the number of postmen is suitable for the size of the graph.

3.2 CPP Tour Lower Bound (CPP/k-LB)

We simply compute the optimum 1-postman tour and divide its weight by k. It is clear that we get a valid lower bound since the 1-postman tour includes a minimum augmentation to make the given graph G Eulerian.

3.3 LP Relaxation Lower Bound (LP-LB)

We consider the following integer programming formulation of the MM k-CPP.

$$\begin{array}{ll}
\multicolumn{2}{l}{\min T} \\
\text{s.t.} & \\
\sum_{e\in E} w(e)x^i(e) + w(e)y^i(e) \le T & i = 1,\dots,k \\
\sum_{i=1}^{k} x^i(e) = 1 & \text{for all } e \in E \\
\sum_{e\in\delta(v)} x^i(e) + y^i(e) \equiv 0 \pmod 2 & \text{for all } v \in V, i = 1,\dots,k \\
x^i(\delta(S)) + y^i(\delta(S)) \ge 2x^i(e) & \text{for all } S \subseteq V\setminus\{v_1\}, e \in E(S), i = 1,\dots,k \\
T \in \mathbb{R}^+ & \\
x^i(e) \in \{0,1\}, y^i(e) \in \mathbb{N}_0 & \text{for all } e \in E, i = 1,\dots,k
\end{array}$$

The binary variables $x^i(e)$ indicate if edge e is serviced by tour C_i or not. Integer variables $y^i(e)$ count how often edge e is traversed by tour C_i without servicing it. The variable T models the length of the longest tour which is expressed with the first constraint. The second set of equations ensures that each edge is serviced exactly once. Each single tour of a valid k-postman tours must be a closed walk containing the depot. These requirements are enforced by the third and fourth set of constraints, respectively. For our lower bound we computed the LP relaxation of the above integer program, omitting the fourth set of constraints.

4 Computational Results

We benchmarked the heuristics and lower bounds for $k = 2, \ldots, 10$ on the following instances.

– Ten random generated CARP instances from [3].
– 23 CARP instances from [13].
– Two CARP instances from [20] and [21].
– 24 random generated *Rural Postman Problem (RPP)* instances from [5, 7, 6].
– Three RPP instances from [7, 6].
– Seven random generated instances.

Due to space restrictions we give explicit results only for seven selected instances and summarize the other results.

Table 1 shows the results for the three RPP instances from [7], Table 2 for the two CARP instances from [21] and Table 3 for four selected random instances.

Table 1. RPP instances from [7]

				$w_{\max}$ for $k =$								
Inst.	$\|V\|$	$\|E\|$	Algorithm	2	3	4	5	6	7	8	9	10
A1	116	174	FHK	**8681**	7003	5825	5265	4929	4529	4278	4085	4218
			AM+	9379	7609	6119	5384	5199	4528	4180	4094	3936
			C+	8736	**6754**	**5540**	**4780**	**4396**	**4084**	**4058**	**3648**	**3620**
			SPT-LB	3476	3476	3476	3476	3476	3476	3476	3476	3476
			CPP/k-LB	7703	5136	3852	3082	2568	2201	1926	1712	1541
			LP-LB	6738	4451	3370	2722	2290	1981	1749	1569	1425
A2	102	160	FHK	8968	6858	5680	5252	4900	4596	4428	4236	4148
			AM+	9238	7160	5952	5427	4963	4458	4257	3989	3798
			C+	**8639**	**6597**	**5412**	**4850**	**4290**	**3812**	**3764**	**3764**	**3532**
			SPT-LB	3436	3436	3436	3436	3436	3436	3436	3436	3436
			CPP/k-LB	7706	5137	3853	3083	2569	2202	1927	1713	1542
			LP-LB	6816	4591	3477	2832	2401	2094	1863	1684	1540
A3	90	144	FHK	8673	6307	5306	4655	4246	3917	4050	3625	3556
			AM+	8073	6411	5395	**4457**	4047	3866	3452	**3422**	3248
			C+	**7892**	**5920**	**5000**	4470	**3748**	**3650**	**3316**	4196	**3124**
			SPT-LB	3124	3124	3124	3124	3124	3124	3124	3124	3124
			CPP/k-LB	7199	4800	3600	2880	2400	2057	1800	1600	1440
			LP-LB	6298	4163	3126	2519	2114	1824	1607	1439	1304

Table 2. CARP instances from [21]

				$w_{\max}$ for $k =$								
Inst.	$\|V\|$	$\|E\|$	Algorithm	2	3	4	5	6	7	8	9	10
e1	77	98	FHK	2093	1559	1270	1180	1180	1126	1126	1014	1014
			AM+	**1915**	**1551**	**1265**	**1133**	1070	1038	1023	983	924
			C+	1930	1593	1291	1156	**1020**	**890**	**874**	**874**	**872**
			SPT-LB	820	820	820	820	820	820	820	820	820
			CPP/k-LB	1685	1124	843	674	562	482	422	375	337
			LP-LB	1471	991	760	621	528	462	412	373	343
e2	140	190	FHK	3010	**2171**	**1813**	**1550**	**1431**	1421	1387	1341	**1209**
			AM+	3040	2353	2111	1920	1780	1667	1623	1545	1550
			C+	**2824**	2240	1817	1620	1477	**1374**	**1338**	**1251**	**1209**
			SPT-LB	1027	1027	1027	1027	1027	1027	1027	1027	1027
			CPP/k-LB	2607	1738	1304	1043	869	745	652	580	522
			LP-LB	2324	1560	1179	951	798	690	608	544	494

The first four columns of the tables contain the instance name, the number of nodes $|V|$, the number of edges $|E|$ and the name of the algorithm or lower bound. The remaining columns contain the value $w_{\max}$ computed for $k = 2, \ldots, 10$. The best result is printed in boldface. We use the following abbreviations: FHK for the FHK-heuristic, AM+ and C+ for the augment-merge heuristic and cluster heuristic followed by both improvement procedures, SPT-LB for the shortest path cycle lower bound, CPP/k-LB for the CPP tour lower bound and LP-LB for the LP relaxation lower bound.

For instances from [3] we were able to improve the FHK-heuristic in 91% of all cases with an average improvement of 10%. For instances from [13] we improved the results in 50% of all cases with an average improvement of 7%. For instances from [5] we improved the results in 66% of all cases with an average improvement of 12%.

5 Conclusions

We have presented new heuristics, improvement procedures and lower bounds for the MM k-CPP. In computational experiments we compared our algorithms for a variety of test instances with the only heuristic existing for the MM k-CPP to date, the FHK-heuristic. The results showed that we were able to improve the results obtained by the FHK-heuristic considerably in nearly all cases.

Our future work will comprise further enhancements of the heuristics and improvement procedures as well as the development of stronger lower bounds. The goal is to embed these results in a branch-and-cut framework in order to achieve optimal solutions for the MM k-CPP.

Table 3. Random instances

				$w_{\max}$ for $k =$								
Inst.	$\lvert V\rvert$	$\lvert E\rvert$	Algorithm	2	3	4	5	6	7	8	9	10
`r1`	20	33	FHK	5394	4381	4057	3690	3191	2928	2888	2869	2637
			AM+	5809	5012	4721	**3173**	**2991**	**2687**	**2687**	2687	**2545**
			C+	**4520**	**4100**	**3655**	3306	3061	2802	2802	**2676**	2615
			SPT-LB	2545	2545	2545	2545	2545	2545	2545	2545	2545
			CPP/k-LB	4375	2917	2188	1750	1459	1250	1094	973	875
			LP-LB	4086	2704	2116	1732	1510	1352	1234	1141	1068
`r2`	40	70	FHK	**7758**	6100	5183	4284	4019	3702	3702	3536	3184
			AM+	8624	6017	5104	4324	4032	3971	3394	3525	3026
			C+	8112	**5676**	**4697**	**3910**	**3813**	**3484**	**3295**	**2939**	**2973**
			SPT-LB	2211	2211	2211	2211	2211	2211	2211	2211	2211
			CPP/k-LB	7311	4874	3656	2925	2437	2089	1828	1625	1463
			LP-LB	6542	4390	3368	2755	2346	2054	1835	1664	1528
`r3`	100	199	FHK	12398	**8826**	7588	6628	5726	5267	4803	4475	4437
			AM+	12266	9500	7868	7038	6350	6420	6106	5082	4788
			C+	**11983**	9207	**7152**	**6001**	**5310**	**4732**	**4490**	**4256**	**4102**
			SPT-LB	2900	2900	2900	2900	2900	2900	2900	2900	2900
			CPP/k-LB	11155	7437	5578	4462	3719	3188	2789	2479	2231
			LP-LB	9826	6464	4859	3897	3255	2797	2453	2186	1972
`r4`	100	200	FHK	**11804**	8426	7134	5990	5681	4887	4505	4059	4034
			AM+	11979	9296	7572	6806	5924	5941	5116	4873	4722
			C+	11834	**8252**	**6755**	**5805**	**5074**	**4526**	**4074**	**3833**	**3647**
			SPT-LB	2637	2637	2637	2637	2637	2637	2637	2637	2637
			CPP/k-LB	10877	7251	5439	4351	3626	3108	2720	2417	2176
			LP-LB	9906	6460	4871	3918	3282	2828	2488	2223	2011

References

[1] D. Applegate, W. Cook, S. Dash, and A. Rohe. Solution of a Min-Max Vehicle Routing Problem. *INFORMS Journal on Computing*, 14(2):132–143, 2002. 65

[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. 65

[3] E. Benavent, V. Campos, A. Corberán, and E. Mota. The Capacitated Arc Routing Problem: Lower Bounds. *Networks*, 22:669–690, 1992. 65, 71, 72

[4] N. Christofides. The Optimum Traversal of a Graph. *Omega*, 1:719–732, 1973. 65

[5] N. Christofides, V. Campos, A. Corberán, and E. Mota. An Algorithm for the Rural Postman Problem. Technical Report I. C. O. R. 81.5, Imperial College, 1981. 71, 72

[6] A. Corberán, A. Letchford, and J. M. Sanchis. A Cutting Plane Algorithm for the General Routing Problem. *Mathematical Programming Series A*, 90(2):291–316, 2001. 71

[7] A. Corberán and J. M. Sanchis. A polyhedral approach to the rural postman problem. *European Journal of Operational Research*, 79:95–114, 1994. 71

[8] M. Dror. *Arc Routing: Theory, Solutions and Applications*. Kluwer Academic Publishers, 2000. 65

[9] J. Edmonds and E. L. Johnson. Matching, Euler Tours and the Chinese Postman. *Mathematical Programming*, 5:88–124, 1973. 67, 68

[10] H. A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems I: The Chinese Postman Problem. *Operations Research*, 43(2):231–242, 1995. 65

[11] H. A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems II: The Rural Postman Problem. *Operations Research*, 43(3):399–414, 1995. 65

[12] G. N. Frederickson, M. S. Hecht, and C. E. Kim. Approximation Algorithms for some routing problems. *SIAM Journal on Computing*, 7(2):178–193, May 1978. 64, 67

[13] B. L. Golden, J. S. DeArmon, and E. K. Baker. Computational experiments with algorithms for a class of routing problems. *Computers & Operations Research*, 10(1):47–59, 1983. 71, 72

[14] B. L. Golden and R. T. Wong. Capacitated arc routing problems. *Networks*, 11:305–315, 1981. 65, 67

[15] T. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985. 68

[16] M. Guan. Graphic Programming using odd and even points. *Chinese Mathematics*, 1:273–277, 1962. 67

[17] A. Hertz, G. Laporte, and M. Mittaz. A Tabu Search Heuristic for the Capacitated Arc Routing Problem. *Operations Research*, 48(1):129–135, 2000. 65

[18] A. Hertz, G. Laporte, and P. Nanchen Hugo. Improvement Procedures for the Undirected Rural Postman Problem. *INFORMS Journal on Computing*, 11(1):53–62, 1999. 65

[19] A. Hertz and M. Mittaz. Heuristic Algorithms. In M. Dror, editor, *Arc Routing: Theory, Solutions and Applications*, chapter 9, pages 327–386. Kluwer Academic Publishers, 2000. 65

[20] L. Y. O. Li. *Vehicle Routeing for Winter Gritting*. PhD thesis, Department of Management Science, Lancaster University, 1992. 71

[21] 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. 71, 72

[22] W. L. Pearn. Solvable cases of the k-person Chinese postman problem. *Operations Research Letters*, 16(4):241–244, 1994. 65