Proceedings of the 2011 ORSSA Annual Conference
pp. 19–28
`http://www.orssa.org.za/XXXXX`

ORSSA Proceedings
ISBN
978-0-7972-1351-7
©2012

Constructive heuristics for the Residential Waste Collection Problem

EJ Willemse* JW joubert†

Abstract

The Residential Waste Collection Problem (RWCP) is a realistic extension of the classical Capacitated Arc Routing Problem (CARP), with application in municipal waste collection. Surprisingly, the problem with its extensions have not been solved in literature. This paper presents two heuristics that are capable of solving the RWCP. The heuristics are based on modifications of the classical Path-Scanning and Augment-Merge heuristics for the CARP. The modified heuristics are tested on new benchmark problems for the RWCP, and results show that the algorithms are capable of quickly solving the problem.

Key words: Capacitated Arc Routing Probelm, Intermediate Facilities, Mixed Network, Constructive heuristics

1 Introduction

Solid waste collection and transportation consists of the collection, transportation and disposal of waste at landfill sites, usually through waste collection vehicles. It is well recognised as being the most costly component of the waste management function and can account for between 50-80% of a municipality's solid waste management budget [14, 21, 22]. Of the different types of waste collected by municipalities, residential waste forms the biggest percentage [15]. As such it consumes the largest part of the municipal budget, making it a promising area to target for cost reductions.

The state of practice method of residential waste collection is through curb side collection. Municipalities have to collect the waste of each household at least once a week. Households place their generated waste, which are stored in either bins or bags, on the designated days in front of their properties where waste collection vehicles can then collect the waste. This process is highly repetitive and performed throughout the year, therefore even a small improvement in waste collection and transfer operations can lead to significant savings in costs.

*Corresponding author: CSIR Built Environment, South Africa, email: `ejwillemse@gmail.com`
†University of Pretoria, South Africa, email: `johan.joubert@up.ac.za`

A promising improvement area is to design better waste collection routes. In this paper we show that designing collection routes can be modelled as an extension of the classical Capacitated Arc Routing Problem (CARP). Two constructive heuristics that are capable of solving the problem are developed and tested on benchmark problems, created to mimic waste collection in residential areas.

The remainder of this paper is organised as follows. The next section gives a brief overview of previous work, related to the RWCP. The two heuristics are presented in Section 3 with computational results provided in Section 4. The paper concludes with Section 5 in which we summarize our main findings.

2 Related work

Residential waste collection requires waste to be collected on a street-by-street basis. As such, the problem of designing collection routes can be modelled as an Arc Routing Problem (ARP). Its aim is defined by Eiselt et al. [6] as determining a least-cost traversal of a specified subset of a graph, with or without constraints. Other ARP application areas include, for example, winter gritting [5, 17], postal delivery [13], security guard routing [23], street sweeping [2] and railway maintenance [12]. For a comprehensive review of ARPs the reader is referred to [3, 4, 6, 7].

As most practical routing applications contain capacity restrictions, the Capacitated Arc Routing Problem, first proposed by Golden and Wong [11], is probably the most important problem in the area of arc routing [7]. With the CARP a fleet of homogeneous vehicles are based at a depot and are tasked with serving all the street segments with waste. The problem consists of designing vehicle routes for the fleet of total minimal length so that each route starts and ends at the depot, each road segment with demand is serviced exactly once by a single vehicle, and the sum of demand on any route does not exceed vehicle capacity.

Two CARP extensions inspired by waste collection have been proposed in literature. The Mixed Capacitated Arc Routing Problem, studied by Lacomme et al. [16], Mourão and Amado [18], Mourão et al. [19] and Belenguer et al. [1], allows the modelling of more realistic street networks, containing one and two way streets. The extension further allows for streets that require each side to be serviced separately. The second extension, referred to as the Arc Routing Problem with Intermediate Facilities under Capacity and Length Restrictions (CLARPIF), accounts for intermediate facilities and dumpsites. The extension, first proposed by Ghiani et al. [8], allows for the collection vehicle to unload its cargo at a nearest Intermediate Facility (IF), including dumpsites, and then resume its collection route. A length (or cost) restriction, independent from capacity and typically presenting the number of working hours in a day, is also placed on each vehicle's route.

Both extensions contain elements fundamental to residential waste collection. In response we combine the CARP and CLARPIF into a new problem termed the Residential Waste Collection Problem (RWCP). The RWCP entails designing vehicle routes of total minimum length, so that each route starts and ends at the depot, each road segment with a demand (waste) is serviced exactly once by a vehicle, and the total cost of a vehicle route does not exceed a maximum allowed trip length or cost. Each route contains visits to IFs, which may or may not include the depot, and the sum of demand on the sub-route between dumpsite visits does not exceed vehicle capacity. Lastly, at the end of each route each vehicle must visit a dumpsite before returning to the depot.

To our knowledge the RWCP is new and has not been formally studied or solved in literaure. Various solution approaches have, however, been developed for the MCARP and CLARPIF. Since both problems, and by extension the RWCP, are $\mathscr{N}\mathscr{P}$-hard, the most effective methods for the solving the problems are based on heuristic techniques. The main studies on the problems are by Belenguer et al. [1], Lacomme et al. [16], Mourão and Amado [18] and Mourão et al. [19], all whom develop heuristics for the MCARP; and the works of Ghiani et al. [8], Ghiani et al. [9], and Polacek et al. [20], whom develop heuristics for the CARP with IFs. In all the studies on the MCARP, except the work of Mourão and Amado [18], one of two classical CARP heuristics are modified, namely Path-Scanning and Augment-Merge. The focus of this paper is to further modify the two heuristics to deal with IFs, thus making them capable of solving the RWCP.

3 Constructive heuristics for the RWCP

Formally defined, the RWCP consists of a mixed graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E} \cup \boldsymbol{A})$ where $\boldsymbol{V}$ represents the set of vertices, $\boldsymbol{E}$ represents the set of edges and $\boldsymbol{A}$ represents the set of arcs. A subset of required edges and arcs, $\boldsymbol{E}_r \subseteq \boldsymbol{E}$ and $\boldsymbol{A}_r \subseteq \boldsymbol{A}$, must be serviced by a fleet of K homogenous vehicles with limited capacity Q that are based at the depot vertex v_1. The fleet size, K, may be fixed or left as a decision variable. The vehicles are allowed to unload their waste at any IF at a cost of λ. The set of IFs are modelled by $\boldsymbol{\Gamma}$ where $\boldsymbol{\Gamma} \subset \boldsymbol{V}$. Unless $v_1 \in \boldsymbol{\Gamma}$, a vehicle has to visit an IF before returning to the depot.

Before presenting the modified heuristics for the RWCP, we first show how the mixed graph $\boldsymbol{G}$ can be transformed into a directed graph and we introduce the algorithm encoding scheme used in the remainder of this section.

3.1 Graph transformation and encoding scheme

Consistent with the work of Belenguer et al. [1] and Lacomme et al. [16], the mixed graph $\boldsymbol{G}$ is transformed into a fully directed graph $\boldsymbol{G}^* = (\boldsymbol{V}, \boldsymbol{A}^*)$ by replacing each edge $(v_i, v_j) \in \boldsymbol{E}$ by two opposite arcs $\{(v_i, v_j), (v_j, v_i)\} \in \boldsymbol{A}^*$. Arcs in $\boldsymbol{A}^*$ are indentified by indices from 1 to m where $m = |\boldsymbol{A}^*|$. Each arc $u \in \boldsymbol{A}^*$ has a beginning vertex $b(u)$ and an end vertex $e(u)$. If arc u represents (v_i, v_j) then $b(u) = v_i$ and $e(u) = v_j$. Lastly, each arc u has a deadheading cost $c(u)$. Lastly, the total cost of any route must not exceed the maximum allowed trip cost, L.

The required arcs, $\boldsymbol{A}_r$, and edges, $\boldsymbol{E}_r$, of $\boldsymbol{G}$ correspond in $\boldsymbol{G}^*$ to a subset $\boldsymbol{R} \subseteq \boldsymbol{A}^*$ of required arcs, such that $|\boldsymbol{R}| = 2|\boldsymbol{E}_r| + |\boldsymbol{A}_r|$. Each arc $u \in \boldsymbol{R}$ has a demand $q(u)$, a collection cost $w(u)$ and a pointer $inv(u)$. Each required arc in the original graph $\boldsymbol{G}$ is coded in $\boldsymbol{R}$ by one arc u with $inv(u) = 0$, while each required edge is encoded as two opposite arcs u and v, such that $inv(u) = v$, $inv(v) = u$, $q(u) = q(v)$, $c(u) = c(v)$ and $w(u) = w(v)$. An arc task u represents an edge if $inv(u) \neq 0$. The depot is modelled by including in $\boldsymbol{A}^*$ a fictitious loop $\sigma = (v_1, v_1)$, with $b(\sigma) = e(\sigma) = v_1$, $inv(\sigma) = 0$ and $q(\sigma) = w(\sigma) = c(\sigma) = 0$. Similarly, the set of IFs are modelled in $\boldsymbol{A}^*$ as a set of dummy arcs $\boldsymbol{I}$, such that each IF in $\boldsymbol{\Gamma}$ is modelled as a fictitious loop $\Phi_i \in \boldsymbol{I}$, and Φ_i has the same start and end vertex, and zero cost and demand.

For the RWCP a solution $\boldsymbol{T}$ is a list $(\boldsymbol{T}_1, \ldots, \boldsymbol{T}_K)$ of K vehicle trips. Each trip, $\boldsymbol{T}_i$, is a list of subtrips $(\boldsymbol{T}_{i1}, \boldsymbol{T}_{i2}, \ldots, \boldsymbol{T}_{i,|\boldsymbol{T}_i|})$ which then consists of a list of tasks $(\boldsymbol{T}_{ij1}, \boldsymbol{T}_{ij2}, \ldots, \boldsymbol{T}_{ij,|\boldsymbol{T}_{ij}|})$. The first subtrip, $\boldsymbol{T}_{i1}$, starts at the depot, and the last subtrip, $\boldsymbol{T}_{i,|\boldsymbol{T}_i|}$, ends with an intermediate facility and depot visit. All other subtrips start and end with IF visits whilst taking care that the starting IF of a subtrip coincides with the end IF of the previous subtrip. It is assumed that the shortest path, which can be efficiently calculated using a modified version of as Dijkstras' algorithm [16], is always followed between consecutive tasks. Lastly, denote by $\boldsymbol{D}(u, v)$ the cost of the shortest path from arc u to arc v, excluding the costs of deadheading u and v.

The best IF to visit after servicing arc u and before servicing arc v can be easily pre-calculated, as shown by Ghiani et al. [8]. The best IF to visit is given by $dump(u, v)$ and the cost of the visit, including unload and deadheading costs, is given by $term(u, v)$. Lastly we define $load(\boldsymbol{T}_i)$ as the total demand of trip $\boldsymbol{T}_i$, and $cost(\boldsymbol{T}_i)$ as the total cost of the trip. The same terms are also used to define the demand and cost of subtrips.

3.2 Path-Scanning

The first heuristic that we modify for the RWCP is the Path-Scanning heuristic developed by Golden et al. [10]. Our modification is based on the algorithm of Belenguer et al. [1] for the MCARP. Path-Scanning systematically builds a vehicle trip by adding the closest unserviced arc to the end of the trip. If there are multiple closest arcs one of seven rules (Table 1) is used to break the tie. The algorithm adds the closest unserviced arc to a vehicle

Table 1: *Tie-break rules used with the* Path-Scanning *heuristic to choose arc $u \in \boldsymbol{A}_c$, where $\boldsymbol{A}_c$ is the set of closest arcs.*

Rule	Description
1	Maximise the distance to the depot; $\max\{\boldsymbol{D}(u, \sigma) : u \in \boldsymbol{A}_c\}$
2	Minimise the distance to the depot; $\min\{\boldsymbol{D}(u, \sigma) : u \in \boldsymbol{A}_c\}$
3	Maximise the arc yield; $\max\{q(u)/w(u) : u \in \boldsymbol{A}_c\}$
4	Minimise the arc yield; $\min\{q(u)/w(u) : u \in \boldsymbol{A}_c\}$
5	Use Rule 1 if the vehicle is less than half-full, else use Rule 2
6	Randomly use any of the five rules
7	Do not use any rule and randomly choose u from $\boldsymbol{A}_c$

trip until the vehicle is full, at which point the trip is closed by adding a depot arc visit. A new empty trip is created and the process repeats until all required arcs are serviced.

Our RWCP version of Path-Scanning sees the introduction of subtrips resulting from IF visits. Instead of starting with a trip, the algorithm starts with a subtrip. The first subtrip starts at the depot and the algorithm adds the closest unserviced arc to the end of the subtrip. When the subtrip is full the algorithm closes it by adding an IF visit, and a new subtrip is then created starting at this IF. The process repeats until no unserviced arcs can be added without exceeding the maximum allowed trip length. At this point the subtrip is closed by adding a visit to an IF and a depot. The combined subtrips now form a single vehicle trip whose cost does not exceed the maximum allowed trip length. The process is repeated until all arcs are serviced. Algorithm 1 summarises our RWCP version of the Path-Scanning algorithm.

Algorithm 1: Path-Scanning for the MCARP

```
Input  : The directed transformed graph $\boldsymbol{G}^*$.
Output: A feasible solution $\boldsymbol{T}$ for the RWCP.
Step 0: Set $\boldsymbol{G}' \leftarrow \boldsymbol{G}^*$, $i \leftarrow 1$ and $j \leftarrow 1$.
Step 1: Let the first task in $\boldsymbol{T}_{i1}$ be the depot arc $\sigma$ and set $\boldsymbol{G}'' \leftarrow \boldsymbol{G}'$.
Step 2: Let $u$ be the last task in subtrip $\boldsymbol{T}_{ij}$. Remove from $\boldsymbol{G}''$ all arcs $v$ with
        $costIF(\boldsymbol{T}_i) + \boldsymbol{D}(u,v) + term(v,\sigma) - \lambda > L$. If $\boldsymbol{G}''$ is empty, close $\boldsymbol{T}_{ij}$ by adding
        $dump(u,\sigma)$ and $\sigma$ to the end of the trip; let $i \leftarrow i+1$ and return to Step 1. If $\boldsymbol{G}''$
        is not empty go to Step 3.
Step 3: Set $\boldsymbol{G}''' \leftarrow \boldsymbol{G}''$ and remove from $\boldsymbol{G}'''$ all arcs $v$ with $load(\boldsymbol{T}_{ij}) + q(v) > Q$. If $\boldsymbol{G}'''$
        is empty find the arc $v$ in $\boldsymbol{G}'$ that minimises $term(u,v)$; close the subtrip by adding
        a visit to $dump(u,v)$ to the end of the subtrip; set $j \leftarrow j+1$; create a new subtrip
        $\boldsymbol{T}_{ij}$ that starts at $dump(u,v)$ and return to Step 2. If $\boldsymbol{G}'''$ is not empty go to Step
        4.
Step 4: From the last subtrip task $u$, find the closest arc $v$ in $\boldsymbol{G}'''$. If there are more
        than one closest arc, choose an arc according to a tie-break rule. Add the closest
        chosen arc $v$ to the end of $\boldsymbol{T}_{ij}$ and remove $v$ and $inv(v)$, if it exists, from $\boldsymbol{G}'$ and
        $\boldsymbol{G}''$. If all edges in $\boldsymbol{G}$ are covered, thus $\boldsymbol{G}'$ is empty, go to Step 5, else return to
        Step 2.
Step 5: Close $\boldsymbol{T}_{ij}$ with a trip to $dump(v,\sigma)$ and the $\sigma$ and stop the algorithm. The
        solution $\boldsymbol{T}$ is a feasible solution for the RWCP.
```

Traditionally the Path-Scanning algorithm is executed five times using one of Rules 1-5 (Table 1) in each execution to break closest arc ties. The best of the five solutions is then chosen for implementation. A similar approach can be followed using Rule 6 or 7 of the table with the advantage that an arbitrary number of solutions can be generated, owing to randomness of the rules. We test both the deterministic scheme (Rules 1 to 5) and random schemes (Rules 6 and 7) on the RWCP. The first scheme is the default scheme and is just referred to as Path-Scanning, or PS for short. The second and third schemes, using Rule 6 and 7, are referred to as PS Random Rule 200 and PS Random Arc 200, respectively, where 200 refers to the number of solutions generated.

3.3 Augment-Merge

The second algorithm that we modify for the RWCP is the MERGE algorithm of Belenguer et al. [1] for the MCARP. The algorithm starts by creating a solution that contains a vehicle trip for each required arc. The trips are then systematically reduced through mergers. Initially trips containing only one arc task are merged, but as the process continues, trips with multiple tasks start to develop and these are also merged. As the algorithm progresses the trips become fuller. The algorithm stops when all the trips are full, or near full, and no more mergers are possible.

The merger of two trips, $\boldsymbol{T}_i$ and $\boldsymbol{T}_j$, is performed through the MERGE-TRIPS-PROCEDURE. Before merging the trips the procedure first checks that the combined load of the two trips does not exceed vehicle capacity. The procedure then merges the trips by splicing them together. The merge procedure also considers adding $\boldsymbol{T}_i$ to the end of $\boldsymbol{T}_j$. The procedure also considers the reversal and then merger of trips, resulting in an additional six possible mergers. MERGE-TRIPS-PROCEDURE evaluates all eight possible mergers between $\boldsymbol{T}_i$ and $\boldsymbol{T}_j$ and returns the best merge as $\tilde{\boldsymbol{S}}_{ij}$, and the best saving as $\Delta\tilde{S}_{ij}$.

The MERGE algorithm performs one merger per iteration. During each iteration the algorithm evaluates all feasible trip mergers and the one with the best saving is implemented. The two original trips are replaced with the merged trip and the algorithm proceeds to the next iteration. This process is repeated until no more mergers are possible. Our implementation of Merge for the MCARP is described in Algorithm 2.

Algorithm 2: MERGE

```
Input  : The directed transformed graph $\boldsymbol{G}^*$
Output: A feasible solution $\boldsymbol{T}$ for the MCARP.

Step 0: Set $\boldsymbol{R}' \leftarrow \boldsymbol{R}$. For each arc v $\in \boldsymbol{R}'$ create a vehicle trip servicing only that arc. If
        $inv(v) \neq 0$ then remove $inv(v)$ from $\boldsymbol{R}'$, thus $inv(v)$ will not be assigned to a trip
        in a latter iteration.

Step 1: For the unique trips $\boldsymbol{T}_i$ and $\boldsymbol{T}_j$ in $\boldsymbol{T}$ check that their merger does not violate
        the capacity constraint, i.e., $load(\boldsymbol{T}_i) + load(\boldsymbol{T}_j) \leq Q$. If the merger is possible
        then determine the best merger $\tilde{\boldsymbol{S}}_{ij}$ and cost saving $\Delta\tilde{S}_{ij}$ through MERGE-TRIP-
        PROCEDURE. If no mergers are possible then go to Step 3, else go to Step 2.

Step 2: Using $\min \Delta\boldsymbol{S}_{ij}$, find the best merge $\tilde{\boldsymbol{S}}_{kl}$ and replace trips $\boldsymbol{T}_i$ and $\boldsymbol{T}_j$ in $\boldsymbol{T}$ with
        $\tilde{\boldsymbol{S}}_{kl}$. If $\boldsymbol{T}$ now consists of only one trip then stop, else return to Step 1.

Step 3: Since no more mergers are possible stop the algorithm and return $\boldsymbol{T}$ as a solution
        for the MCARP.
```

To apply Merge to the RWCP we execute MERGE twice, with minor modifications to the algorithm in each execution. During the first execution each vehicle trip consists of only one vehicle subtrip, with an IF visit before returning to the depot. A merger between two trips is then performed as with the MCARP. Let $\boldsymbol{T}_i = (\sigma, a_1, a_2, \ldots, a_m, \Phi_i, \sigma)$ and let $\boldsymbol{T}_j = (\sigma, b_1, b_2, \ldots, b_n, \Phi_j, \sigma)$, where Φ_i and Φ_j are calculated as $\Phi_i = dump(a_m, \sigma)$ and $\Phi_j = dump(b_n, \sigma)$. A new merged trip $\boldsymbol{S}_{ij}$ is created such that

$$\boldsymbol{S}_{ij} = (\sigma, a_1, a_2, \ldots, a_m, b_1, b_2, \ldots, b_n, \Phi_j, \sigma).$$

Before merging the trips the procedure first checks that the combined load of the two trips does not exceed vehicle capacity. The algorithm evaluates all feasible trip mergers and the one with the best saving is implemented. As with the MCARP the algorithm checks all eight possible merge orientations. The first MERGE execution terminates when no more mergers are possible without violating the vehicle capacity restrictions.

The algorithm then proceeds with the second MERGE execution in which vehicle subtrips are merged into vehicle trips. Again let $\boldsymbol{T}_i = (\sigma, a_1, a_2, \ldots, a_m, \Phi_i, \sigma)$ and let $\boldsymbol{T}_j = (\sigma, b_1, b_2, \ldots, b_n, \Phi_j, \sigma)$. A new merged trip, $\boldsymbol{S}_{ij}$, is now created such that

$$\boldsymbol{S}_{ij} = \big((\sigma, a_1, a_2, \ldots, a_m, \Phi_s), (\Phi_s, b_1, b_2, \ldots, b_n, \Phi_j, \sigma)\big).$$

Before merging the trips the procedure checks that the combined cost of the two trips does not exceed the maximum trip length restriction. The algorithm evaluates all feasible trip mergers and the one with the best saving is implemented. The algorithm again checks all eight possible merge orientations. The second MERGE execution terminates when no more mergers are possible without violating the maximum trip length restriction. Since the general structure of the two executions of MERGE for the RWCP is the same as with the MCARP, we do not give a formal description of our implementation.

4 Computational results

To evaluate and compare the heuristics, both are tested on a new set of benchmark problems. The new problems were created by extending the *lpr* problem set of Belenguer et al. [1] by including IFs at vertices $\lfloor |\boldsymbol{V}|/2 \rfloor$ and $2\lfloor |\boldsymbol{V}|/2 \rfloor$. For each problem we introduce a max trip length of 28 800 seconds, which corresponds to an eight hour working day, and the vehicle capacity remains fixed at 10 000kg. Lastly, the number of vehicles required to service the area is not fixed but left as a decision variable. The new RWCP benchmark problems are referred to as the *lpr-IF* problem set.

The heuristics are evaluated on three criteria: total fleet travel time minimisation, vehicle fleet size minimisation and running time. All the algorithms were coded in Python version 2.6 and run on a 3Ghz Intel(R) Core(TM)2 Duo CPU with 3.25GB of RAM.

The results of the three PATH SCANNING heuristics and the MERGE heuristics on the *lpr-IF* problem set are shown in Table 2. The table shows the total cost of each solution generated, the number of vehicles that the solution requires and the total time required to generate the solution. Results show the random versions of Path Scanning to produce slightly better solutions than the deterministic versions, which is expected since they generate more solutions from which to choose the best. The results further show that the Path-Scanning heuristics outperforms Merge on all but four of the problems. Merge faired particularly poorly in minimising the vehicle fleet size and required an extra route with eight of the fifteen solutions.

In terms of the computational time, Path-Scanning significantly outperformed Merge on all the problem instances. On all the *lpr-IF* problems the Path-Scanning variants are capable of generating up to 200 solutions in less than one minute, making them very efficient, even on large problems. Based on the tests for the *lpr-IF*, we conclude that Path-Scanning is better suited than Merge to solve the RWCP.

Table 2: *Computational results for Path-Scanning and Merge on the lpr-IF problem set, with the best solutions for each problem underlined. All time values are given in seconds.*

	PS			PS RA 200			PS RR 200			Merge		
File	K	Cost	Time	K	Cost	Time	K	Cost	Time	K	Cost	Time
a-01	1	13783	0.1	1	13678	0.3	1	13659	0.4	1	13954	0.9
a-02	2	29687	0.1	1	28647	1.1	1	28605	1.1	1	28791	6.9
a-03	3	80261	0.2	3	79545	9.0	3	79748	9.1	4	78695	173.4
a-04	5	134027	0.6	5	133434	26.2	5	133779	26.4	6	141678	802.8
a-05	8	213707	1.6	8	212208	66.2	8	212489	66.8	9	218128	3339.1
b-01	1	15049	0.1	1	14911	0.3	1	14868	0.3	1	15261	0.8
b-02	2	29955	0.1	2	29684	1.1	2	29715	1.1	2	29119	6.3
b-03	3	81708	0.2	3	80779	8.7	3	80912	8.7	4	80302	174.9
b-04	5	134580	0.6	5	133351	22.8	5	133567	23.0	6	136897	793.0
b-05	8	222874	1.5	8	221952	63.2	8	222502	62.5	9	228057	3227.5
c-01	1	18837	0.1	1	18783	0.5	1	18814	0.5	1	18973	0.9
c-02	2	36903	0.1	2	36796	1.6	2	36808	1.6	2	37345	8.2
c-03	4	114763	0.4	4	114593	16.3	4	114179	16.5	5	116422	219.4
c-04	7	177011	1.0	7	175819	41.2	7	176366	41.1	7	172506	916.4
c-05	10	276976	2.1	10	276876	89.2	10	276239	89.7	11	276447	3370.1

Acronyms: PS - Path Scanning, RR - Random Rule, RA - Random Arc, K - Number of routes

5 Conclusion

The RWCP is a new problem combining key elements of the MCARP and CLARPIF consistent with actual waste collection and transportation activities. Two heuristics were developed and tested to solve the RWCP, of which Path-Scanning performed the best on fifteen newly developed benchmark problems. The heuristic also proved efficient and is capable of generating multiple solutions within minutes on large problems. The developed constructive heuristics forms an important first step in studying the RWCP as its starting solutions can be used with developing and testing improvement heuristics and metaheuristics for RWCP.

Bibliography

[1] Belenguer, J.-M., Benavent, E., Lacomme, P., and Prins, C. (2006). Lower and upper bounds for the mixed capacitated arc routing problem. *Computers & Operations Research*, 33(12):3363–3383.

[2] Bodin, L. D. and Kursh, S. J. (1979). A detailed description of a computer system for the routing and scheduling of street sweepers. *Computers & Operations Research*, 6(4):181–198.

[3] Corbern, A. and Prins, C. (2010). Recent results on arc routing problems: An annotated bibliography. *Networks*, 56(1):50–69.

[4] Dror, M., editor (2000). *Arc Routing: Theory, Solutions, and Applications.* Boston: Kluwer Academic Publishers.

[5] Eglese, R. and Li, L. Y. O. (1992). Efficient routeing for winter gritting. *The Journal of the Operational Research Society*, 43(11):1031–1034.

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

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

[8] Ghiani, G., Guerriero, F., Laporte, G., and Musmanno, R. (2004). Tabu search heuristics for the arc routing problem with intermediate facilities under capacity and length restrictions. *Journal of Mathematical Modelling and Algorithms*, 3(3):209–223.

[9] Ghiani, G., Improta, G., and Laporte, G. (2001). The capacitated arc routing problem with intermediate facilities. *Networks*, 37(3):134–143.

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

[11] Golden, B. L. and Wong, R. T. (1981). Capacitated arc routing problems. *Networks*, 11(3):305–315.

[12] Groves, G., Le Roux, J., and Van Vuuren, J. (2004). Network service scheduling and routing. *International transaction in operational research*, 11(6):613–643.

[13] Irnich, S. (2008). Solution of real-world postman problems. *European Journal of Operational Research*, 190(1):52–67.

[14] Karadimas, N. V., Papatzelou, K., and Loumos, V. G. (2007). Optimal solid waste collection routes identified by the ant colony system algorithm. *Waste Management & Research*, 25(6):139–147.

[15] Korfmacher, K. S. (1997). Solid waste collection systems in developing urban areas of South Africa: an overview and case study. *Waste Management Research*, 15(5):477–494.

[16] Lacomme, P., Prins, C., and Ramdane-Chérif, W. (2004). Competitive memetic algorithms for arc routing problems. *Annals of Operations Research*, 131(4):159–185.

[17] Li, L. Y. O. and Eglese, R. W. (1996). An interactive algorithm for vehicle routeing for winter - gritting. *The Journal of the Operational Research Society*, 47(2):2.

[18] Mourão, M. C. and Amado, L. (2005). Heuristic method for a mixed capacitated arc routing problem: A refuse collection application. *European Journal of Operational Research*, 160(1):139–153.

[19] Mourão, M. C., Nunes, A. C., and Prins, C. (2009). Heuristic methods for the sectoring arc routing problem. *European Journal of Operational Research*, 196(3):856–868.

[20] Polacek, M., Doerner, Karl F. Hartl, R. F., and Maniezzo, V. (2008). A variable neighborhood search for the capacitated arc routing problem with intermediate facilities. *Journal of Heuristics*, 14(5):405–423.

[21] Tchobanoglous, G., Theisen, H., and Vigil, S. (1993). *Integrated solid waste management: Engineering principles and management issues*. McGraw-Hill New York.

[22] Viotti, P., Pelettini, A., Pomi, R., and Innocetti, C. (2003). Genetic algorithms as a promising tool for optimisation of the MSW collection routes. *Waste Management Research*, 21(4):292–298.

[23] Willemse, E. J. and Joubert, J. W. (In Press). Applying min-max k postmen problems to the routing of security guards. *Journal of the Operational Research Society*. Advance online publication, doi:10.1057/jors.2011.26.