International Journal of Industrial Engineering Computations 3 (2012) 767–776

Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

A novel heuristic method to solve the capacitated arc routing problem

Alireza Eydi* and Leila Javazi

Faculty of Engineering, University of Kurdistan, Pasdaran Blvd., Post box no.: 416, Sanandaj, Iran

A R T I C L E I N F O

Article history:
Received 18 June 2012
Received in revised format
7 July 2012
Accepted July 31 2012
Available online
7 August 2012

Keywords:
Capacitated arc routing problem
Heuristic

A B S T R A C T

The capacitated arc routing problem is one of the most important routing problems with many applications in real world situations such as snow removing, winter gritting, refuse collection, etc. Since this problem is NP-hard, many of researchers have been developed numerous heuristics and metaheuristics to solve it. In this paper, we propose a new constructive and improvement heuristic in which forming a vehicle's tour is based on choosing an unserved edge randomly as current partial tour and then extending this partial tour from its both of end nodes base on four effective proposed criteria. When the vehicle load is near its capacity, it should come back to the depot immediately. Finally, the constructed tours are merged into more efficient and cheaper tours. The quality of this new approach was tested on three standard benchmark instances and the results were compared with some known existing heuristics and metaheuristics in the literature. The computational results show an excellent performance of our new method.

1. Introduction

The capacitated arc routing problem (CARP) is one of the most important routing problems in literature, and attracted interest of many researchers. It has numerous applications in real world situations such as refuse collection (Dijkgraaf & Gradus, 2007), winter gritting (Eglese & Li, 1992), snow removing (Labelle et al., 2002), inspection of gas pipeline (Han et al., 2004), street sweeping (Tobin & Brinkmann, 2002), and electric meter reading (Stern & Dror, 1979). The CARP was first introduced by Golden and Wong (1981) and deals with connected and undirected graph $G= (V, E)$, where V is the set of vertices (nodes) and E is the set of edges. Each edge of E has a definite travel cost and some edges have positive demand, called required edges, which must be serviced by some vehicles with limited capacity. All vehicles are identical and located at a single depot. The aim of CARP is to design a set of vehicle tours of minimum total routing cost such that each tour starts and ends at the depot, each required edge is serviced by exactly one vehicle, and the total demand serviced by any vehicle most not exceed the vehicle's capacity. Some instances (in small size) of CARP can be solved for optimality by implementing exact methods such as branch and bound (Hirabayashi et al., 1992), branch and cut and price algorithm (Aragão et al., 2006), and cutting plane algorithm (Belenguer & Benavent, 2003). Wøhlk (2006) proposed a new lower bound, the Multiple Cuts Node Duplication

* Corresponding author. Tel:+988716665813
E-mail: Alireza.eydi@uok.ac.ir (A. Eydi)

doi: 10.5267/j.ijiec.2012.08.002

Lower Bound, for the undirected CARP. However, the CARP is a NP-hard problem (Golden & Wong, 1981) and these exact methods are not able to solve the large-scale instances in polynomial time. Therefore, due to the computational complexity of the problem, there have been remarkable attempts by researchers in developing heuristic and metaheuristics algorithms to solve it. Tabu search is the first metaheuristics proposed by Hertz et al., (2000). Here solutions breaking vehicle capacity are accepted but penalized. Three improvement procedures (Shorten, Drop, Add) initially explained by Hertz et al. (1999) and four new ones (Paste, Cut, Switch, and Postopt) are used. Lacomme et al. (2004a) proposed a memetic algorithm to solve an extended version of the CARP; each required edge is represented by two directions. The chromosomes are encoded as large tours. Each chromosome is evaluated optimally using a splitting procedure, which partitions the large tour into feasible trips. Ant colony system (Lacomme et al., 2004b) is one of the other metaheuristics in which two types of ant are used to work through the problem. These are elitist ants that make the solutions converge towards a minimum cost solution and non-elitist ants that guarantee diversification to prevent being trapped in a local minimum. Beside metaheuristics, heuristics are better with required short CPU time. Furthermore, they are implemented easier and provide a good initial solution to start of many metaheuristics. However, metaheuristics give solutions with more quality. Augment-Merge (Golden & Wong, 1981), Path-scanning (Golden et al., 1983), Double Outer Scan heuristic (Wøhlk, 2005), Ulusoy's heuristic (Ulusoy, 1985), Ellipse Rule based Path-scanning heuristic (Santos et al., 2009), and Construct-Strike (Pearn, 1989) are some of the known heuristics to solve the CARP. For a detailed overview of the main characteristics of the heuristics in the literature, the readers may refer to Wøhlk (2008). Yet, the development of enhanced heuristics is an important research area for the CARP.

It is note that researchers try to develop various models of classical CARP and consequently develop efficient heuristic and metaheuristic methods to solve these models. For example, recently Kirlik et al. (2012) introduced a new model of CARP with deadheading demands and modified the Ulusoy's heuristic (Ulusoy, 1985) to solve it. Grandinetti et al. (2012) by giving an optimization-based heuristic solved CARP with three objectives: the total transportation cost, the longest route cost, and the number of vehicles. In addition, Salazar-Aguilar et al. (2012) proposed an adaptive large neighborhood search heuristic for synchronized arc routing problem.

The main objective of the current research is to propose a new heuristic for classical CARP that employs some ideas of "Double Outer Scan heuristic" and "Path-scanning whit Ellipse Rule heuristic" to solve the problem. This proposed heuristic is based on selecting an unserved edge randomly, then extending it by both of its end points, into a vehicle's tour based on four effective criteria. When the vehicle load is near its capacity or there is no qualified edge to add the current tour, vehicle should return to the depot by using shortest path. Finally, in order to reduce the total cost and efficient usage of vehicle capacity, the constructed tours are merged into shorter tours. The remainder of the paper is structured as follows: A brief review about Double Outer Scan heuristic (DOS) and Ellipse Rule heuristic based on Path-scanning (RSE-ER), is presented in section 2. In section 3, we describe our heuristic method. Section 4 is devoted to computational results and experimental analysis. Finally, some concluding remarks are stated in section 5.

2. Brief review on DOS and RSE-ER

In this section, we give a brief review of DOS and RSE-ER. Details of these methods can be found in Wøhlk, (2005) and Santos et al. (2009), respectively. Double Outer Scan heuristic was introduced by Wøhlk (2005) and combines the Augment-Merge algorithm and the Path-scanning method. Unlike the Augment-Merge which always selects the edge that has the shortest path from the end points of the current tour, here the neighbor edges is considered. In each iteration the unserved edge that is farthest away from the depot is selected, and from this edge, vehicle scan in the Path-scanning heuristic way to service the other edges, but unlike the Path-scanning heuristic, this done from both ends of the current partial tour. Finally, the obtained tours are merged into shorter tours. Ellipse Rule based on path-scanning heuristic is a modification of the path scanning algorithm in which, when the vehicle is near the end of a route, in other words, when the vehicle load is around its capacity, the ellipse rule impels the vehicle to service only arcs near the shortest path between the last serviced arc and the depot.

Furthermore, Path-scanning is based on construction of each tour by adding one edge to the partial tour at a time. In order to choose the next edge, if the tie occurs, the five criteria (Golden et al., 1983) including: 1) Minimize the distance to the depot; 2) Maximize the distance to the depot; 3) Minimize the distance per unit demand ; 4) Maximize the distance per unit demand; 5) Use criterion 1, if the vehicle is more than half-full, otherwise use criterion 2; are used and among the obtained five solutions, the best one is selected as the final solution. Pearn (1989) used a modified path-scanning heuristic based on random selection of the five criteria. Belenguer et al. (2006) suggested another path-scanning based upon random selection of the tied arcs. Recently, Santos et al. (2009) indicate that the solutions achieved by the random selection of tied arcs are similar quality to those identified by the five criteria of Golden et al. (1983) and Pearn (1989). So Santos et al. (2009) used random-add approach in the Path-scanning with Ellipse Rule heuristic.

In this paper, by employing some ideas of DOS and RSE-ER, we propose a new robust heuristic method. At each iteration, this heuristic chooses one arc, randomly. This arc forms the current partial tour. Like DOS, we extend the current partial tour by both of its end points, but here we use four proposed criteria that will be presented in the next section. We choose those required edges that are incident to current partial tour. In RSE-ER, Santos et al. (2009) used ellipse rule, which forces the vehicle to serve only edges near the shortest path between the last serviced edge and the depot, when the vehicle is near the end of a route. However, here if the vehicle load is near its capacity or there is no qualified arc to add the current partial tour, we force the vehicle to return to the depot immediately then the chance of saving cost will be increased in merging phase.

3. Problem solving technique

3.1. Problem definition and notations

In this section, we introduce the problem definition and notations to facilitate the description of the heuristic algorithm. Let *G=(V,E)* be a connected and undirected graph, where $V=\{v_1,v_2,...,v_n\}$ is the set of nodes and $E=\{(v_i,v_j)\,|\,v_i,v_j\in V,\ i<j\}$ is the set of edges. Required edges are those with positive demand and can be shown as $E_R\subseteq E$. Each edge $e\in E$ has a nonnegative travel cost $c(e)\geq 0$, and each edge $e\in E_R$ is associated with a positive demand $q(e)>0$. Node v_1 denotes the depot where a fleet of identical vehicles with limited capacity $Q(Q\geq \max\{q(e),\ e\in E_R\})$, are located at v_1. We represent each edge in two directions; positive (from v_i to v_j) and negative (from v_j to v_i) directions that so called arcs. In order to facilitate, the arcs with positive direction is denoted by *p* and the arcs with negative direction is denoted by *n*. Each arc has a tail node *t* and a head node *h*. Further, the opposite direction of arc is denoted by *inv*. Hence, the following features are notable:

$$h(p)=t(n)=v_i;\ t(p)=h(p)=v_j;\ c(p)=c(n);\ q(p)=q(n);\ inv(p)=n;\ inv(n)=p;$$

Let *td* be the total demand, *nre* be the number of required edges, *rvc* be the remaining vehicle capacity, and α be a real parameter. The goal of the problem is to determine a set of least-cost tours of all edges $e\in E_R$ such that each required edge is served by one vehicle exactly, and the total vehicle load at any time does not exceed the definite capacity *Q*.

3.2. Description of the new heuristic algorithm

In this section, we present our new heuristic algorithm. As mentioned before, by employing some features of DOS, and RSE-ER with some differences, we present a new robust heuristic. Like DOS, we extend the current partial tour by its both of ends, but here, we use four new criteria, and like RSE-ER, we force the vehicle back to the depot when its load is near its capacity, but without serving any edge between the last served edge and the depot. Finally, we merge the obtained tours in order to reduce the total cost. Our algorithm can be described as follows:

Step 1. Select one unserved arc randomly, and remove its inverse from unserved arc set. This arc forms the current partial tour.

Suppose that the selected arc is in positive direction and is shown as P_{sel}, so let hp_{sel} and tp_{sel} be the first and last arcs of the current partial tour, respectively.

Step 2. Set $\{p_i\}$ as those tail-neighbor and head-neighbor required arcs with P_{sel} such that head-neighbor $\{p_i \,|\, t(p_i) = h(hp_{sel}) \neq v_1\}$ and tail-neighbor$\{p_i \,|\, h(p_i) = t(tp_{sel}) \neq v_1\}$. For the sake of convenience, we abbreviate tail-neighbor and head-neighbor by *tngb* and *hngb*, respectively. Fig. 1.a shows this step of the algorithm.

Step 3. Randomly choose one of following four criteria with equal probability. Then based on the result, sort the arcs in *tngb* and *hngb* sets.

1) Minimum distance from $h(p_i),\ p_i \in hngb$ to depot and Minimum distance from $t(p_i),\ p_i \in tngb$ to depot;
2) Maximum distance from $h(p_i),\ p_i \in hngb$ to depot and Maximum distance from $t(p_i),\ p_i \in tngb$ to depot;
3) Minimum distance from $h(p_i),\ p_i \in hngb$ to depot and Maximum distance from $t(p_i),\ p_i \in tngb$ to depot;
4) Maximum distance from $h(p_i),\ p_i \in hngb$ to depot and Minimum distance from $t(p_i),\ p_i \in tngb$ to depot;

Step 4. If both *hngb* set and *tngb* set are not empty and $rvc > \alpha \times td / nre$, choose the first arc with smaller demand to serve and add it to current partial tour, and then if $rvc > \alpha \times td / nre$, add another arc in another set with greater demand to current partial tour, else back to the depot (see fig. 1.b). By this idea, we force the vehicle to services just those unserved edges that are incident to the current partial tour. Consequently, more of the required edges are served by the vehicle in its tour. Note that when one arc is selected to receive a service, its inverse must be deleted from unserved arcs.

If *hngb*(*tngb*) set is empty, in other words: there are no unserved edges incident to current partial tour by its head, and the remaining capacity of vehicle *rvc* is greater than $\alpha \times td / nre$, vehicle services the first arc in obtained *tngb*(*hngb*) set in step 2, otherwise it should return to the depot.

Step 5. Update *hngb* set and *tngb* set, in other words subject to new obtained current partial tour, form the *hngb* and *tngb* again.

Step 6. Repeat step 3 to step 5 until vehicle load approaches its capacity or both *hngb* and *tngb*, becoming empty. Then connect the $h(hp_{sel})$ and $t(tp_{sel})$ to the depot by using the shortest path.

Step 7. Repeat step 1 to step 6 until all required edges are served.

Step 8. Merge the constructed tours into less cost tours, subject to vehicle capacity.

Step 9. Repeat steps 1 to 8 for maximum iteration (stopping criterion) determined by decision maker; finally the best solution is selected.

This algorithm is same as explained before if the selected arc in step 1 be in negative direction, and just *p* in all notations is replaced with *n* (e.g., p_{sel} is replaced with n_{sel}). Fig. 2 presents the general structure of the proposed heuristic:

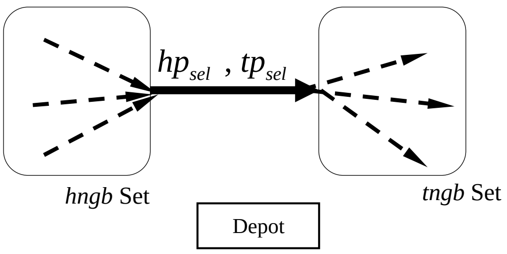

Fig. 1.a. Presentation of forming the *hngb* set and *tngb* set

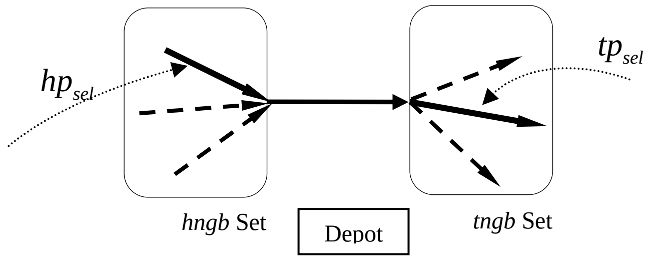

Fig. 1.b. Presentation of determining the (hp_{sel}) and (tp_{sel})

While (maximum iteration to be reached){
While (all reqired edges to be served){
Select a required arc to be served
Form *tngb* and *hngb*
While ($rvc < \alpha \times td / nre$ or $hngb \neq \varnothing \ \wedge \ tngb = \varnothing$){
Choose one of four proposed criteria and sort *hngb* and *tngb* based on
If ($rvc > \alpha \times td / nre \ \wedge \ hngb \neq \varnothing \ \wedge \ tngb \neq \varnothing$){
Between the first required arcs of two sets *hng*b and *tngb*,
serve that with less demand
If ($rvc > \alpha \times td / nre$)
Serve another arc of another set
update *hngb* and *tngb*
}
else if ($rvc > \alpha \times td / nre \ \wedge \ hngb = \varnothing \ \wedge \ tngb \neq \varnothing$){
Serve the first arc of *tngb*
update *hngb* and *tngb*
}
else if ($rvc > \alpha \times td / nre \ \wedge \ hngb \neq \varnothing \ \wedge \ tngb = \varnothing$)
Serve the first arc of *hngb*
update *hngb* and *tngb*
}
else connect the constructed partial tour to depot from both of its end
by shortest path
}
}
Merge constructed tours
}

Fig. 2. General structure of the proposed heuristic

4. Computational results

In this section, we show our computational results. The aforementioned algorithm has been coded in C# language and run on a laptop computer with CPU clock frequency 2.66 GHz and 4Gbyte of RAM. In order to evaluate the performance of our heuristic method, we have implemented it on three standard CARP benchmark test sets. The first set contains 23 *gdb* instances introduced by DeArmon (1981) with

7-27 nodes and 11-55 edges, all of which is required. This set contains 25 instances but gdb8 and gdb9 contain inconsistencies, and they have never been used in the literature. The second set consists of 34 *val* problems proposed by Benavent et al. (1992) whose ranges are from 24 to 50 nodes and fro 34 to 97 edges. The last set is bigger which is based on a winter gritting problem (Eglese, 1994) proposed by Belenguer and Benavent (2003) and includes 24 *egl* instances with 77–140 nodes and 98–190 edges and in some instances not all edges are required. All these set of benchmark are available at http://www.uv.es/~belengue/carp.html. In our implementation, we have followed the practice of Santos et al. (2009). Hence, the parameter α is set at 1.5.

The results of our algorithm for three sets of benchmark (*gdb*, *val*, *egl* files) are given in table 1 to 3, respectively. In all these tables, the row named "our algorithm" shows the obtained results by proposed algorithm, over 10 runs for 1000, 10000 and 20000 iterations. Note that due to the large size instances in *egl* files, the row in Table 3 is divided to 10000, 20000 and 25000 iterations. We have compared our computational results with four known heuristics; Path-scanning heuristic (PS) (Golden et al., 1983), Augment-Merge heuristic (AM) (Golden & Wong, 1981), Double Outer Scan heuristic (DOS) (Wøhlk, 2005), and Ellipse Rule based Path Scanning heuristic (with 10000 iteration) (RSE-ER (10000)) (Santos et al. 2009) and also three of the well known metaheuristics including Tabu Search algorithm (CARPET) (Hertz et al., 2000), Memetic algorithm (MA) (Lacomme et al., 2004a) and Ant Colony Optimization algorithm (BACO) (Lacomme et al., 2004b). The columns headed "Ave", "#Opt", "Dev (%)", and "Time" (en second) provide, for each row, average value, number of optimal results, average percentage deviation above lower bound, and running time, respectively. (i.e., deviation above lower bound is equal to $((\cos t - LB)/LB \times 100)$. The details of results can be found in Appendix A.

Table 1
Computational results for *gdb* files

		Ave			
		Cost	Time(s)	#Opt	Dev (%)
Our algorithm:	1000	257.4	0.09	16	1.16
	10000	256.7	0.84	16	1
	20000	256.4	1.75	16	0.87
Heuristics:	PS	279.6	*	3	8.27
	AM	286.2	*	2	10.92
	DOS	313.6	*	0	24.07
	RSE-ER(10000)	*	1.25	*	1.13
Metaheuristics:	CARPET	255	3.6	18	0.48
	MA	253.9	2.12	21	0.15
	BACO	254.4	7.43	18	0.28

"*" unknown values

Table 2
Computational results for *val* files

		Ave			
		Cost	Time(s)	#Opt	Dev (%)
Our algorithm:	1000	360.2	0.36	7	4.3
	10000	357.4	3.71	7	3.6
	20000	355.6	7.16	7	3.2
Heuristics:	PS	415	*	0	20.35
	AM	402.1	*	0	16.4
	DOS	484.2	*	0	35
	RSE-ER(10000)	*	2.6	*	4.46
Metaheuristics:	CARPET	350.8	25.55	15	1.9
	MA	344.8	15.34	22	0.61
	BACO	346.4	82.9	19	0.89

"*" unknown values

Table 3
Computational results for *egl* files

		Ave		
		Cost	Time(s)	Dev (%)
	10000	10421.5	35.18	8.1
Our algorithm:	20000	10391.4	70.41	7.8
	25000	10375.04	88.15	7.7
	PS	12958.2	*	33.6
Heuristics:	AM	11866.2	*	25.8
	DOS	10633	*	11.4
	RSE-ER(10000)	*	9.216	8.95
	CARPET	10074	*	4.74
Metaheuristics:	MA	9834.1	210.8	2.47
	BACO	10033	702.4	4.1

"*" unknown values

Note that in order to have a fair comparison, the running times are scaled for the 2.66 GHz computer used in this paper, in other words we normalize the running times by multiplying with a CPU speed ratio. Since MA of Lacomme et al. (2004a) and RSE-ER (10000) of Santos et al. (2009) was implemented on 1 GHz Pentium III PC, the running times were multiplied by 0.4, and the running times for CARPET scaled to 1 GHz Pentium III PC by Lacomme et al. (2004a); so they were multiplied by 0.4 too. BACO of Lacomme et al. (2004b) was executed on an 800 MHz Pentium III PC; so here the execution times are multiplied by 0.3. All running times are in seconds. The "*" symbol indicate the values that we do not have any information about them, unfortunately.

4.1. Analysis of experiments

As it can be seen from Table 1, our proposed heuristic algorithm outperforms all Path-Scanning, Augment-Merge, and Double Outer Scan heuristics. In all 1000, 10000, and 20000 iterations for the 16 problem instances, our algorithm reached the optimal solution, whereas PS and AM reached the 3 and 2 optimal solutions respectively, and DOS reached to no optimal solution. In addition, our proposed heuristic algorithm with 10000 and 20000 iterations performs better than ERS-ER (10000) with the routing cost. It is obvious the metaheuristics give the solutions with higher quality rather than heuristics. For example, the ratio of average percentage deviation to LBs of our algorithm (20000) to CARPET, MA and BACO are 1.8, 5.8 and 3.1 respectively, and prove that our approach has significant advantages than other heuristic algorithms with quality of solutions. Table 2 shows the similar results for *val* files. The best lower bound is obtained in 7 problem instances while the number of optimal solution obtained by PS, AM and DOS is zero. Furthermore, average percentage deviation above the LBs for our heuristic is less and is compatible with RSE-ER. Compared to CARPET, MA and BACO, the quality of solutions with our heuristic (20000 iterations) is 1.68, 5.25 and 3.6 times better, respectively. In Table 3, the computational results for *egl* files are reported. The *egl* files are much harder than the previous files (*gdb* and *val* files), and lower bounds are never reached. Hence, we have removed the column headed "#opt" from Table 3. Concerning the average percentage deviation to LBs, one can see that our heuristic algorithm is superior to PS, AM, DOS and RSP-ER. Also, the solutions obtained by our method are near to those obtained from CARPET, MA and BACO with total routing cost and average results.

5. Conclusion

CARP is a NP-hard problem (Golden & Wong, 1981); consequently many of researchers attempt to develop heuristics and metaheuristics to solve it. In this paper we attempt to incorporate ideas of both, Double Outer Scan heuristic (Wøhlk, 2005) and Ellipse Rule based Path scanning heuristic (Santos et al., 2009), to provide a new heuristic for the CARP that can be effective to generate an initial solution in many kinds of metaheuristics. Our heuristic chooses an unserved edge randomly, and then extends it by both its end points based on four criteria that stated in section 3.2. If the remaining capacity of vehicle is less than predefined value or there is no qualified edge to add the current tour, the vehicle should return to the depot and a new tour is started. We implemented the new heuristic algorithm on three set of standard instances (*gdb*, *val* and *egl* files) and compared our computational results with some known heuristics (Path-scanning, Augment-Merge, Double Outer Scan and Ellipse Rule based Path-Scanning) and metaheuristics(Tabu Search, Memetic Algorithm and Ant Colony Optimization algorithm) that have been presented in the literature. In addition, we compared the results with best lower bounds designed by Belenguer and Benavent (2003). Although the solution times of our heuristic algorithm are rarely long, but subject to the quality of obtained solutions, one can disregards the solution time. Our research is emblematic of that this new approach to solve the CARP is excellent, and when the solution time is important, even it can be replaced metaheuristics methods. The results indicate our approach can be used and developed to solve the other types of CARP in real-world applications.

References

Aragão, M.P., Longo, H., & Uchoa, E. (2006). Solving capacitated arc routing problems using a transformation to the CVRP. *Computers & Operations Research*, 33(6), 1823–1837.

Belenguer, J.M., & Benavent, E. (2003). A cutting plane algorithm for the Capacitated Arc Routing Problem. *Computers & Operations Research* 30(5), 705-728.

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

Benavent, E., Campos, V., Corberan, A., & Mota, E. (1992). The capacitated arc routing problem: Lower bounds. *Networks*, 22, 669-690.

DeArmon, J.S. (1981). *A comparison of heuristics for the capacitated Chinese postman problem*. Dissertation, University of Maryland.

Dijkgraaf, E., & Gradus, R. (2007). Fair competition in the refuse collection market. *Applied Economic Letters*, 14(10), 701–704.

Eglese, R.W. (1994). Routing winter gritting vehicles. *Discrete applied mathematics*, 48(3), 231-244.

Eglese, R.W., & Li, L.Y.O. (1992). Efficient Routing for Winter Gritting. *Journal of Operational Research Society* 43(11), 1031–1034.

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

Golden, B.L., & Wong, R.T. (1981). Capacitated arc routing problems. *Networks*, 11, 305–315.

Grandinetti, L., Guerriero, F., Lagana, D., & Pisacane, O. (2012). An optimization-based heuristic for the Multi-objective Undirected Capacitated Arc Routing Problem. *Computers & Operations Research*, 39(10), 2300-2309.

Han, H.S., Yu, J.J., Park, C.G., & Lee, J.G. (2004). Development of inspection gauge system for gas pipeline. *Korean Society Mechanical Engineering International Journal*, 18(3), 370–378.

Hertz, A., Laporte, G., & Mittaz, M. (2000). A tabu search heuristic for the capacitated arc routing problem. *Operations Research*, 48(1), 129–135.

Hertz, A., Laporte, G., & Nanchen-Hugo, P. (1999). Improvement procedures for the undirected rural postman problem. *INFORMS Journal on Computing*, 11(1), 53–62.

Hirabayashi, H., Saruwatari, Y., & Nishida, N. (1992). Tour construction algorithm for the capacitated arc routing problems. *Asia-Pacific Journal of Operations Research*, 9, 155–175.

Kirlik, G., & Sipahioglu, A. (2012). Capacitated arc routing problem with deadheading demands. *Computers & Operations Research,* 39(10), 2380-2394.

Labelle, A., Langevin, A., & Campbell, J.F. (2002). Sector design for snow removal and disposal in urban areas. *Socio-Economic Planning Sciences*, 36(3), 183–202.

Lacomme, P., Prins, C., & Ramdane-Cherif, W. (2004a). Competitive memetic algorithms for arc routing problems. *Annals of Operation Research,* 131, 159–185.

Lacomme, P., Prins, C., & Tanguy, A. (2004b). First competitive ant colony scheme for the CARP. *Lecture Notes in Computer Science*, 3172, 426-427.

Pearn, W.L. (1989). Approximate solutions for the capacitated arc routing problem. *Computers & Operations Research*, 16, 589-600.

Salazar-Aguilar, M.A., Langevin, A., & Laporte, G. (2012). Synchronized arc routing for snow plowing operations. *Computers & Operations Research*, 39(7), 1432-1440.

Santos, L., Coutinho-Rodrigues, J.R., & Current, J.R. (2009). An improved heuristic for the capacitated arc routing problem. *Computers & Operations Research*, 36(9), 2632–2637.

Stern, H.I., & Dror, M. (1979). Routing Electric Meter Readers. *Computers & Operations Research* 6(4), 209–223.

Tobin, G.A., & Brinkmann, R. (2002). The effectiveness of street sweepers in removing pollutants from road surfaces in Florida. *Journal of Environmental Science and Health (Part A)* 37(9), 1687–1700.

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

Wøhlk, S. (2008). *A decade of capacitated arc routing. In: Golden B, Raghavan S, Wasil E, editors. The vehicle routing problem: latest advances and new challenges*. New York, USA: Springer, 29–48.

Wøhlk, S. (2005). *Contributing to arc routing*. PhD thesis, University of Southern Denmark.

Wøhlk, S. (2006). New lower bound for the capacitated arc routing problem. *Computers & Operations Research*, 33(12), 3458-3472.

Appendix A. Details of results

Tables A.1 to A.3 show a comparison between the obtained results by proposed heuristic and the obtained results by some heuristic and metaheuristic algorithms reported in literature. In all these tables the first column gives the name of instance. Columns labeled "|V|"and "$|E_R|$" stands for the number of vertices and required edges, respectively. The column headed "LB" shows the best lower bonds that proposed by Belenguer and Benavent (2003). Note that the best obtained solution over 10 runs by our algorithm has been selected as final solution. The times reported for CARPET and MA are those given by Lacomme et al. (2004a), and for BACO are those presented by Lacomme et al. (2004b). These times are scaled in tables 1 to 3 as described in section 4. Unfortunately, we do not have any information about details of results of RSE-ER, so subject to Santos et al. (2009), we have only provided the average of percentage deviation to LBs and average of running time in section 4.

Table A.1
Computational results for *gdb* filse

				Our algorithm														
				Iteration=1000		10000		20000		Heuristics			Metaheuristic					
Instances	|V|	|ER|	LB	Cost	Time	Cost	Time	Cost	Time	PS	AM	DOS	CARPET	Time	MA	Time	BACO	Time
gdb1	12	22	316	**316**	0.05	**316**	0.45	**316**	0.9	345	351	370	**316**	3.15	**316**	0	**316**	0.5
gdb2	12	26	339	345	0.06	345	0.53	345	1.06	369	394	414	**339**	5.17	**339**	0.44	**339**	1.8
gdb3	12	22	275	**275**	0.05	**275**	0.5	**275**	1.01	284	338	354	**275**	0.07	**275**	0.06	**275**	0.5
gdb4	11	19	287	**287**	0.04	**287**	0.43	**287**	0.87	321	342	372	**287**	0.09	**287**	0	**287**	0.1
gdb5	13	26	377	383	0.06	383	0.54	383	1.09	429	383	501	**377**	5.59	**377**	0.11	**377**	2.2
gdb6	12	22	298	**298**	0.04	**298**	0.41	**298**	0.84	332	354	370	**298**	0.85	**298**	0.17	**298**	1.1
gdb7	12	22	325	**325**	0.05	**325**	0.48	**325**	0.96	359	359	368	**325**	0	**325**	0.05	**325**	0.1
gdb8	27	46	344	365	0.21	360	2.02	358	4.07	402	399	400	352	61	350	0.66	350	130.6
gdb9	27	51	303	331	0.21	327	2.07	322	4.12	374	369	375	317	53.91	**303**	7.09	306	330.1
gdb10	12	25	275	**275**	0.06	**275**	0.56	**275**	1.13	307	319	371	**275**	1.55	**275**	0.06	**275**	0.7
gdb11	22	45	395	**395**	0.2	**395**	2.03	**395**	4.11	451	457	515	**395**	2.29	**395**	1.26	**395**	7.3
gdb12	13	23	450	468	0.05	468	0.55	468	1.1	550	577	594	458	20.63	458	0.06	458	2.8
gdb13	10	28	536	548	0.07	544	0.68	544	1.37	562	586	641	544	2.42	**536**	7.42	542	26.6
gdb14	7	21	100	**100**	0.04	**100**	0.41	**100**	2.37	112	108	146	**100**	0.48	**100**	0.05	**100**	0.4
gdb15	7	21	58	**58**	0.04	**58**	0.4	**58**	0.8	**58**	**58**	74	**58**	0	**58**	0	**58**	0.2
gdb16	8	28	127	**127**	0.07	**127**	0.68	**127**	1.37	131	137	143	**127**	1.7	**127**	0.06	**127**	6.5
gdb17	8	28	91	**91**	0.07	**91**	0.65	**91**	1.32	**91**	**91**	109	**91**	0	**91**	0.05	**91**	0.2
gdb18	9	36	164	**164**	0.09	**164**	0.97	**164**	1.89	168	170	202	**164**	0.28	**164**	0.11	**164**	1.1
gdb19	8	11	55	**55**	0.02	**55**	0.2	**55**	0.41	**55**	63	73	**55**	0.2	**55**	0	**55**	0.2
gdb20	11	22	121	**121**	0.05	**121**	0.46	**121**	0.9	123	123	147	**121**	9.5	**121**	0.33	**121**	22.3
gdb21	11	33	156	**156**	0.09	**156**	0.9	**156**	1.8	162	160	181	**156**	1.13	**156**	0.17	**156**	8
gdb22	11	44	200	**200**	0.14	**200**	1.34	**200**	2.76	202	204	224	**200**	3.38	**200**	3.35	**200**	19.6
gdb23	11	55	233	237	0.2	235	2	235	4.07	243	241	269	235	34.37	**233**	51.19	235	7

Table A.2
Computational results for *val* filse

Instances	|V|	|ER|	LB	Our algorithm Iteration=1000 Cost	Time	10000 Cost	Time	20000 Cost	Time	Heuristics PS	AM	DOS	Metaheuristic CARPET	Time	MA	Time	BACO	Time
val1a	24	39	173	**173**	0.14	**173**	1.39	**173**	2.91	197	194	240	**173**	0.02	**173**	0	**173**	0.1
val1b	24	39	173	179	0.12	177	1.32	177	2.57	199	200	243	**173**	9.26	**173**	8.02	**173**	120.6
val1c	24	39	235	260	0.13	258	1.39	256	2.74	321	298	284	245	93.2	245	0.27	245	13.1
val2a	24	34	227	**227**	0.1	**227**	1.05	**227**	2.06	258	263	317	**227**	0.17	**227**	0.05	**227**	2
val2b	24	34	259	260	0.09	260	0.97	260	1.96	296	311	363	260	13.02	**259**	0.22	**259**	8.4
val2c	24	34	455	482	0.1	476	1.07	463	2.11	538	533	533	494	31.66	457	8.08	457	135.1
val3a	24	35	81	**81**	0.1	**81**	1.05	**81**	2.1	92	84	102	**81**	0.77	**81**	0.05	**81**	1.2
val3b	24	35	87	88	0.09	88	0.94	88	1.95	107	90	115	**87**	2.79	**87**	0	**87**	3.6
val3c	24	35	137	146	0.1	143	1.02	142	1.99	155	160	157	138	41.66	138	0.49	138	10.6
val4a	41	69	400	**400**	0.36	**400**	3.3	**400**	6.87	490	435	577	**400**	28.32	**400**	0.72	**400**	15.3
val4b	41	69	412	432	0.34	422	3.43	422	6.65	478	641	596	416	75.66	**412**	1.21	**412**	117.1
val4c	41	69	428	461	0.33	456	3.03	448	6.35	518	491	593	453	70.06	**428**	19.11	430	285.4
val4d	41	69	520	578	0.36	572	3.55	574	7.02	662	653	660	556	233.56	530	6.37	539	315.9
val5a	34	65	423	433	0.29	433	2.9	433	5.68	498	502	637	**423**	3.8	**423**	1.86	**423**	49.5
val5b	34	65	446	460	0.25	453	2.77	451	5.26	509	487	588	448	41.4	**446**	1.04	**446**	24.3
val5c	34	65	469	492	0.24	483	2.72	483	5.1	600	550	680	476	53.27	474	0.44	474	200.3
val5d	34	65	571	647	0.29	636	2.87	624	5.41	821	726	791	607	224.11	581	11.32	597	193.8
val6a	31	50	223	**223**	0.19	**223**	1.91	**223**	4.26	243	252	294	**223**	3.89	**223**	0.17	**223**	3.8
val6b	31	50	231	242	0.18	242	1.93	241	3.42	282	258	314	241	26.94	233	6.48	233	78.4
val6c	31	50	311	334	0.19	334	2.14	327	4.04	391	370	364	329	85.18	317	52.23	317	91.6
val7a	40	66	279	**279**	0.45	**279**	4.43	**279**	9.05	358	329	393	**279**	6.59	**279**	4.66	**279**	11.2
val7b	40	66	283	287	0.44	293	4.41	286	8.28	345	335	397	**283**	0.02	**283**	0.44	**283**	6.6
val7c	40	66	333	352	0.36	348	3.59	344	7.52	417	405	409	343	121.44	334	60.53	334	569.3
val8a	30	63	386	**386**	0.29	**386**	2.9	**386**	6.07	445	411	556	**386**	3.84	**386**	0.66	**386**	15.4
val8b	30	63	395	404	0.26	403	2.68	403	5.57	499	425	572	401	81.46	**395**	9.95	**395**	259.5
val8c	30	63	517	588	0.27	578	2.75	579	5.28	613	645	660	533	147.4	528	62.83	534	358.1
val9a	50	92	323	325	0.9	324	9.51	324	18.48	388	367	458	**323**	28.51	**323**	18.29	**323**	969
val9b	50	92	326	329	0.83	327	8.83	327	17.1	388	373	467	329	59.89	**326**	29.39	**326**	1076.2
val9c	50	92	332	341	0.81	338	8.12	338	15.68	407	385	473	**332**	56.44	**332**	71.19	**332**	1368.5
val9d	50	92	382	431	0.72	420	7.24	424	13.57	503	457	507	409	353.28	391	211.13	404	634
val10a	50	97	428	434	0.83	433	8.44	432	15.57	471	471	587	**428**	5.52	**428**	25.48	**428**	341.8
val10b	50	97	436	448	0.78	448	7.81	448	14.4	471	471	598	**436**	18.43	**436**	4.67	437	683.4
val10c	50	97	446	470	0.73	466	7.74	464	13.43	509	497	601	451	93.47	**446**	17.3	448	515.8
val10d	50	97	524	575	0.71	570	7.14	562	13.11	641	603	652	544	156.31	528	215.04	536	916.1

Table A.3
Computational results for *egl* filse

Instances	|V|	|ER|	LB	Our algorithm Iteration=10000 Cost	Time	20000 Cost	Time	25000 Cost	Time	Heuristics PS	AM	DOS	Metaheuristic CARPET	MA	Time	BACO	Time
e1-a	77	51	3515	3779	4.08	3779	8.2	3779	10.33	3885	4939	4414	3625	3548	1.48	3548	70.7
e1-b	77	51	4436	4716	4.16	4715	8.4	4716	10.45	6601	5371	4770	4532	4498	48.39	4534	307.5
e1-c	77	51	5453	5884	4.09	5835	8.15	5855	10.17	6719	6827	6063	5663	5595	39.98	5647	159.1
e2-a	77	72	4994	5275	10.86	5302	21.7	5271	27.22	6199	6596	5778	5233	5018	20.6	5018	470.4
e2-b	77	72	6249	6670	10.64	6668	21.37	6652	26.83	7451	8372	6735	6422	6340	22.19	6401	406.4
e2-c	77	72	8114	8691	10.84	8700	21.5	8687	27.08	9532	10590	8934	8603	8395	27.52	8498	707.4
e3-a	77	87	5869	6136	20.81	6132	41.84	6117	52.37	6169	7643	6442	5907	5898	24.44	5934	609.8
e3-b	77	87	7646	8199	18.81	8144	37.54	8115	46.98	8510	9441	8107	7921	7816	173.18	7915	781.9
e3-c	77	87	10019	10775	17.48	10725	35.27	10695	44.06	12175	12657	11084	10805	10369	111.5	10402	226.7
e4-a	77	98	6372	6716	17.7	6702	35.32	6716	44.23	7410	8116	7322	6489	6461	275.5	6520	616.8
e4-b	77	98	8809	9505	16.84	9516	33.87	9460	42.58	9916	10302	9681	9216	9021	291.49	9234	839.8
e4-c	77	98	11276	12286	16.59	12276	33.18	12219	41.65	68226	13692	12404	11824	11779	77.83	11883	799.3
s1-a	140	75	4992	5256	6.99	5252	13.74	5252	17.6	5345	6512	5529	5149	5018	15.88	5049	1010.5
s1-b	140	75	6201	6706	6.94	6682	13.79	6684	17.26	6296	8552	6806	6641	6435	21.42	6541	2899.8
s1-c	140	75	8310	9001	6.42	8932	12.75	8904	15.77	8692	10608	9053	8687	8518	160.38	8561	2388.9
s2-a	140	147	9780	10685	80.91	10645	162.19	10716	203.16	10217	12097	11111	10373	9995	795.1	10368	4108
s2-b	140	147	12886	14015	68.46	13908	137.48	13848	172.7	14773	15249	14242	13495	13174	641.58	13676	5377.6
s2-c	140	147	16221	17732	60.67	17728	121.2	17755	151.51	17517	19767	17890	17121	16715	743.69	17115	3099.3
s3-a	140	159	10025	10939	84.31	10871	168.9	10857	211.14	11931	12544	11471	10541	10296	651.03	10619	1392.1
s3-b	140	159	13554	14645	70.26	14601	139.95	14657	176.02	13916	16116	14962	14291	14028	1043.6	14264	6568.6
s3-c	140	159	16969	18744	64.34	18639	131.13	18600	161.25	17740	20070	18563	17789	17297	622.58	17797	3160
s4-a	140	190	12027	13343	90.11	13356	179.15	13269	224.65	13596	14989	13962	13036	12442	1529.6	12868	8919.2
s4-b	140	190	15933	17817	78.2	17731	155.07	17675	195.52	16830	19249	17723	16924	16531	1184.5	17090	6360
s4-c	140	190	20179	22601	73.9	22554	148.03	22502	184.99	21351	24493	22142	21486	20832	1464.3	21314	4911.4