Computers & Operations Research 36 (2009) 2632 – 2637

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/cor

An improved heuristic for the capacitated arc routing problem

Luís Santos[a,c], João Coutinho-Rodrigues[a,c,*], John R. Current[b,c]

[a]Department of Civil Engineering, Faculty of Sciences and Technology, Polo II, University of Coimbra, 3030-788 Coimbra, Portugal
[b]Department of Management Sciences, The Fisher College of Business, The Ohio State University, 632 Fisher Hall, 2100 Neil Avenue, Columbus. OH 43210-1144, USA
[c]INESC-Coimbra, R. Antero Quental, 199, 3000-033 Coimbra, Portugal

ARTICLE INFO

Available online 19 November 2008

Keywords:
Vehicle routing
Capacitated arc routing problem
Heuristics

ABSTRACT

The capacitated arc routing problem (CARP) is an important and practical problem in the OR literature. In short, the problem is to identify routes to service (e.g., pickup or deliver) demand located along the edges of a network such that the total cost of the routes is minimized. In general, a single route cannot satisfy the entire demand due to capacity constraints on the vehicles. CARP belongs to the set of NP-hard problems; consequently numerous heuristic and metaheuristic solution approaches have been developed to solve it. In this paper an "ellipse rule" based heuristic is proposed for the CARP. This approach is based on the path-scanning heuristic, one of the mostly used greedy-add heuristics for this problem. The innovation consists basically of selecting edges only inside ellipses when the vehicle is near the end of each route. This new approach was implemented and tested on three standard datasets and the solutions are compared against: (i) the original path-scanning heuristic; (ii) two other path-scanning heuristics and (iii) the three best known metaheuristics. The results indicate that the "ellipse rule" approach lead to improvements over the three path-scanning heuristics, reducing the average distance to the lower bound in the test problems by about 44%.

1. Introduction

Vehicle routing is an important activity in both the public and private sectors. In the United States for example, transportation costs account for approximately 6% of the GDP [1]. As a consequence, even small improvements in routing efficiency can result in large cost reductions. Improving routing efficiency will become even more important as a result of recent increases in fuel prices. A common vehicle routing problem involves the pickup or delivery of items located along the arcs of a road network (e.g., trash collection and parcel delivery). Other examples of real applications can be found in Assad and Golden [2], Dror [3], and Santos et al. [4]. Many of these applications can be structured as a capacitated arc routing problem (CARP).

Golden and Wong [5] introduced CARP. The problem may be stated as follows. Let $G=(N,E)$ be an undirected graph, where $N=\{v_0,\dots,v_n\}$ is a node set and $E=\{[v_i,v_j]:v_i,v_j\in N, i<j\}$ is an edge set. Each edge $[v_i,v_j]$ of E has a nonnegative cost or length c_{ij} and nonnegative demand or weight q_{ij}. Node v_0 represents a depot at which λ identical vehicles of capacity w ($w \geqslant \max_{[v_i,v_j]\in E} q_{ij}$) are based. The number of vehicles (λ) is a decision variable. The CARP consists of designing a set of vehicle routes, such that: (1) each positive-demand edge is serviced by exactly one vehicle; (2) each route starts and ends at the depot; (3) the total demand of all edges serviced by any vehicle does not exceed w; and (4) the total routing cost is minimized.

CARP can be formulated as an integer linear programming and solved optimally using a branch-and-bound algorithm (e.g., [6,7]). Given that CARP belongs to the class of NP-hard problems [5] such approaches are limited to small problem instances. For example, Hirabayashi et al. [6] solved problems with up to 30 edges and Longo et al. [7] with up to 87 edges.

As a result of CARP's many practical applications and its computational complexity, considerable research has been devoted to developing heuristic procedures to solve it. Path-scanning [8] is the most commonly used greedy-add solution approach to CARP. At each iteration, these heuristics "scan" the remaining unserved edges to determine which one should be visited next on the current vehicle route. This research introduces a new path-scanning heuristic that includes an "ellipse rule" to solve the problem. In essence, the ellipse rule only considers edges inside an ellipse when the vehicle is near the end of a route (i.e., when the vehicle load is near its capacity). This new heuristic was implemented and tested on three standard

* Corresponding author at: Department of Civil Engineering, Faculty of Sciences and Technology, Polo II, University of Coimbra, 3030-788 Coimbra, Portugal. Tel.: +351 239 797 145; fax: +351 239 797 123.
E-mail addresses: lsantos@dec.uc.pt (L. Santos), coutinho@dec.uc.pt (J. Coutinho-Rodrigues), current.1@osu.edu (J.R. Current).

0305-0548/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2008.11.005

data sets. The results are compared to prior path-scanning heuristics and the three best known metaheuristics for the problem. When compared to the existing path-scanning heuristics, the experimental tests demonstrate that the new heuristic generates higher quality results in comparable CPU time. When compared to the metaheuristics, the experimental tests demonstrate that the new heuristic generates lower quality results, as expected, but in considerably less CPU time. Other reasons for the development of improved heuristics for CARP are presented in Section 2.1.

The remainder of this paper is organized as follows. Existing heuristic solutions for CARP are presented in the next section. The new ellipse-based heuristic is introduced in the third section. Computational results and comparisons are given in the fourth section followed by a summary and conclusions in the last section.

2. Existing heuristic solution procedures for CARP

2.1. General overview

Over the years, various heuristics (e.g., [5,8,9]) and lower bound generating techniques (e.g., [5,7,10,11]) have been proposed for CARP. These heuristics (usually referred to as “greedy-add heuristics”) generally provide good approximate solutions in acceptable CPU time, given the computational complexity of the problem. Descriptions and computational performance comparisons of these heuristics are presented in Coutinho et al. [12] which concluded that the CARP solution heuristic first introduced by Golden et al. [8] is one of the best in terms of solution quality and CPU time required to obtain the solutions.

More recently, metaheuristics have been proposed to solve CARP. These include tabu search [13,14], genetic algorithms [15], and ant colony optimization [16]. In general, these metaheuristics identify better solutions than do the path-scanning heuristic. However, this improvement comes with an increase in solution time

The development of improved greedy-add heuristics for CARP is still an important area of research even though the application of metaheuristics to CARP has led to improved solutions. Several reasons exist for this in addition to reduced solution times required. First, greedy-add heuristics are more intuitive as they mimic the way that people approach many problems. Consequently, they are more likely to be accepted by managers. Second, they are easier to implement (program and encode) and do not require the determination and fine tuning of various metaheuristic parameters such as mutation rate, aspiration levels, tabu list length, number of ants, evaporation coefficient, etc. Third, their simplicity makes them more flexible as they can be more easily modified to accommodate changes in the underlying problem (e.g., prohibited turns and maximum route duration). Finally, they form the starting point for various metaheuristics. For example, the genetic algorithm of Lacomme et al. [15] and Belenguer et al. [17], and the algorithm based on ant colony optimization of Lacomme et al. [16] use the original and a modified path-scanning heuristic to generate one of its “good” starting solutions and the tabu search approach of Greistorfer [14] starts with a greedy-add heuristic solution presented in Greistorfer [18].

2.2. Previous path-scanning heuristics for CARP

In the original path-scanning heuristic [8] each solution is constructed by adding one edge at a time. In determining the edge to add, the heuristic considers variously edge length, edge demand, distance to the depot node, v_0, distance to the next unserved edge, and unused capacity of the vehicle. Specifically, Golden et al. [8] proposed five criteria to determine which edge to service next on a route. These criteria may be stated as follows:

1. minimize the ratio c_{ij}/q_{ij};
2. maximize the ratio c_{ij}/q_{ij};
3. minimize the cost from node v_j back to the node v_0;
4. maximize the cost from node v_j back to the node v_0;
5. if the vehicle is less than half-full use criterion 4, otherwise use criterion 3,

where v_i is the last node visited by the route to date and $[v_i, v_j]$ is a feasible unserved edge (i.e., q_{ij} is less than or equal to the remaining vehicle capacity).

A problem instance is solved five times, using a different selection criterion each time. The best of the five solutions generated is the final solution. Golden et al. [8] did not state which edge should be selected if all edges $[v_i, v_j]$ incident to the last node added (v_i) have no unserved demand. The generally accepted interpretation (e.g., [15,17,19]) in such situations is that the feasible unserved edge nearest to node v_i ($[v_p, v_j]$) is added, where nearest is measured by shortest path distance between node v_i and node v_p. The appropriate selection criterion is used whenever a tie for the nearest edge occurs (in this case, v_i is replaced by v_p in the criteria described above). This interpretation is used in this research.

Pearn [9] modified this approach by selecting one of the five criteria at random, with equal probability, whenever a tie for the nearest edge (incident or not to the last node added to the route) occurs. Each problem was solved k times (results for $k = 30$ were published) and the best solution was selected at the end. One of the main advantages of this approach vis-à-vis the Golden et al. [8] heuristic, is that it generally generates more solutions; consequently, the probability of identifying a better solution increases.

Recently, Belenguer et al. [17] proposed another path-scanning heuristic. At each iteration, this heuristic adds the feasible unserved edge that is nearest to the last node added (v_i), as in the other path-scanning heuristics described above. Each problem is solved k times and the best solution is selected at the end. At any iteration, if there exist more than one feasible unserved edge that are at the minimum distance from node v_i, these edges are referred to as “tied” edges. In such situations, Belenguer et al. [17] randomly select one of the tied edges to be the next edge added to the route. This differs from the Golden et al. [8] and Pearn [9] heuristics in that they would select the next edge to be added in such situations based upon the particular selection criterion in effect at the time.

Belenguer et al. [17] compared their heuristic with the random selection of criteria [9] using three standard data sets (referred to as: *gdb*, *val* and *egl*, and described in Section 4). Belenguer et al. [17] presented results for $k = 20$ and 50. These results suggested that the random selection of tied edges is slightly worse for the *gdb* and *val* problems, and is slightly better for the *egl* problems. However, in all three data sets, the differences are very small. In order to confirm these results, we solved the problems with $k = 1000$, 10 000 and 20 000. Our results, presented in Table 1, confirm the results obtained by Belenguer et al. (2006).

3. A new “ellipse rule” based path-scanning heuristic

The new heuristic is similar to the one proposed by Belenguer et al. [17] in that it randomly selects tied edges and solves each problem k times. It constructs a route by adding a new edge at each iteration. The edge added is the nearest edge to the last one added. In cases where there are ties for the nearest edge, the heuristic randomly

Table 1
Evaluation of the three path-scanning heuristics

	k	PSP (k)	RSE (k)	RSE_ER (k)											
				$\alpha=0.5$		$\alpha=1$		$\alpha=1.5$		$\alpha=2$		$\alpha=2.5$		$\alpha=3$	
		ΔLB	ΔLB	ΔLB	#UE (1)	ΔLB	#UE (1)	ΔLB	#UE (1)	ΔLB	#UE (1)	ΔLB	#UE (1)	ΔLB	#UE (1)
gdb files	1000	3.73	3.94	2.27	7.01	1.76	18.38	1.47	22.20	2.10	32.35	2.05	36.78	5.06	51.47
	10 000	2.75	2.79	1.65	7.64	1.32	18.54	1.13	22.46	1.39	33.21	1.66	36.06	3.69	51.48
	20 000	2.61	2.53	1.49	7.59	1.20	18.21	1.07	22.54	1.27	33.21	1.44	36.06	3.49	51.00
val files	1000	8.83	9.21	7.89	3.15	6.83	6.61	6.28	10.61	5.64	13.07	5.62	17.50	6.53	20.70
	10 000	6.90	6.55	5.90	3.15	5.01	6.44	4.56	9.92	4.10	13.61	4.41	17.26	5.01	21.00
	20 000	5.83	6.19	5.35	2.53	4.65	6.96	3.93	9.45	3.73	14.18	3.83	17.49	4.65	20.89
egl files	1000	18.53	18.30	11.98	12.12	9.95	19.05	10.33	22.94	13.95	28.54	16.20	32.53	21.69	36.33
	10 000	16.91	16.63	10.82	12.20	8.86	17.38	8.95	23.32	12.01	27.88	14.58	33.35	19.88	35.74
	20 000	16.64	16.35	10.65	11.76	8.51	17.35	8.74	22.90	11.49	27.68	13.99	32.87	19.39	36.61
Global average		9.19	9.17	6.44	7.46	5.34	14.32	5.16	18.48	6.19	24.86	7.09	28.88	9.93	36.14
Average for $k=1000$		10.36	10.48	7.38		6.18		6.03		7.23		7.96		11.09	
Average for $k=10\,000$		8.85	8.66	6.12		5.07		4.88		5.83		6.89		9.53	
Average for $k=20\,000$		8.36	8.36	5.83		4.79		4.58		5.50		6.42		9.18	

ΔLB—Average deviation to the LB (%). #UE (1)—Average number of unserved edges that satisfy inequality (1) (%).

selects one of the tied edges to add. The new heuristic differs from the Belenguer et al. [17] one in that it utilizes an "ellipse rule" when the vehicle load is near capacity. Intuition suggests that as a vehicle's load approaches its capacity, its route should stay closer to the depot to reduce its cost to return to the depot when full. The "ellipse rule" is designed to implement this intuition. When the vehicle is near the end of a route, this rule forces the vehicle to service only edges near the shortest path between the last serviced edge and the depot (v_0). This rule is similar to the one proposed by Norback and Love [20] for solving the travelling salesman problem.

The "ellipse rule" is defined as follows. Let *ned* be the number of edges with positive demand in the network, *td* the total demand to be collected, *tc* the total cost assigned to edges with positive demand, α a real parameter, and $[v_h, v_i]$ the last serviced edge on the route. If the remaining capacity of the vehicle is less than or equal to $\alpha \times td/ned$ (i.e., the average demand on the arcs), then the next edge to be serviced $[v_p, v_j]$ must be the nearest edge to $[v_h, v_i]$ ($v_i = v_p$, if the edges are adjacent) satisfying the condition:

$$SP(v_i, v_p) + c_{pj} + SP(v_j, v_0) \leqslant tc/ned + SP(v_i, v_0), \tag{1}$$

where for example, $SP(v_i, v_p)$ is defined as the shortest path cost between the nodes v_i and v_p, and v_i and v_0 are the foci of the ellipse. If no feasible edge (i.e., an edge with demand less than or equal to the remaining capacity of the vehicle) satisfies (1) then the vehicle returns directly to the depot. In general, as α increases, the ellipse rule is invoked earlier in the construction of a vehicle route. When enforced, condition 1 limits the addition of arcs to the route to those that are within an "ellipse" such as those shown in Fig. 1. In general, *tc*/*ned* will be less than $SP(v_i, v_0)$ because it represents the average cost (distance) of an arc in the network while $SP(v_i, v_0)$ must include at least one arc and generally will include multiple arcs. However, *tc*/*ned* may be greater than $SP(v_i, v_0)$ if node v_i is near node v_0. In Fig. 1, we present 3 ellipses for various values of *tc*/*ned* where $c_1 + c_2 = tc/ned + SP(v_i, v_0)$. We set $SP(v_i, v_0) = 1$ and set *tc*/*ned* equal to 0.1, 0.5, and 1.0. As *tc*/*ned* increases, the area of the ellipse increases and its shape approaches its limit which is a circle. As this occurs, more unserved arcs are likely to be eligible for inclusion on the current route.

Let *nmit* be the maximum number of iterations, S_{cur} the solution being constructed in the current iteration, S_{best} the best solution obtained so far, CS_{cur} and CS_{best} the total cost of solutions S_{cur} and S_{best}, respectively, *rvc* the remaining vehicle capacity, *w* the vehicle capacity, LB the lower bound for the problem, and *F* the set of feasible unserved edges that are at minimum distance from the last node added to S_{cur}.

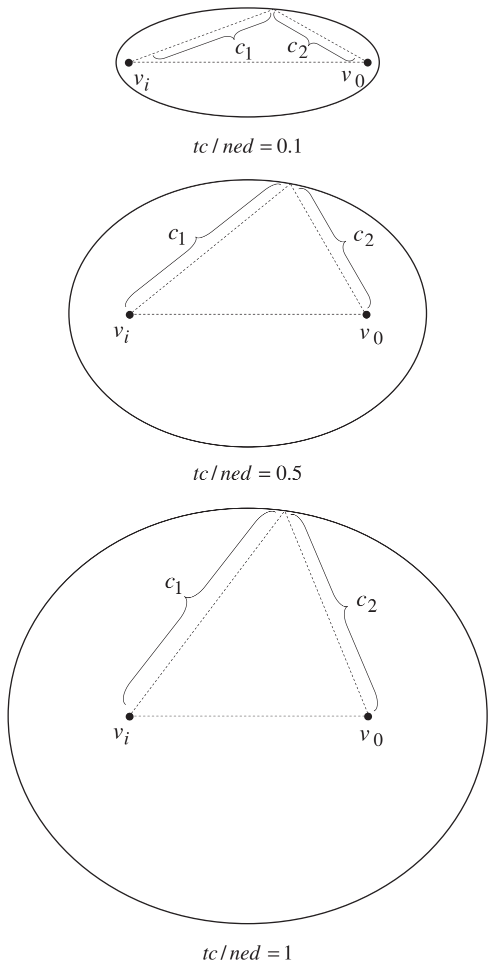

Fig. 1. Comparison of ellipses obtained for $SP(v_i, v_0) = 1$ and several values of *tc*/*ned*.

Given these definitions, the new path-scanning heuristic may be stated as follows.

```
$CS_{best} \leftarrow +\infty$
$iter \leftarrow 1$
**while** (($iter \leqslant nmit$) and ($CS_{best} > LB$)) **do** //if both the maximum nr. of iterations and the LB are not attained
    $S_{cur} \leftarrow v_0$
    $rvc \leftarrow w$
    **for** $i = 1$ **to** $ned$ **do** //for all the edges with positive demand
        **if** ($rvc > \alpha \times td/ned$) **then**
            Determine $F$
        **else**
            Determine $F$ satisfying equation (1)
        **end if**
        **if** ($F = \emptyset$) **then**
            $S_{cur} \leftarrow S_{cur} \cup \{v_0\}$ //the vehicle returns to the depot
            $rvc \leftarrow w$
        **else**
            Select $[v_p, v_j]$ randomly from $F$ //an additional edge is serviced in the solution being constructed
            $S_{cur} \leftarrow S_{cur} \cup \{[v_p, v_j]\} \cup \{v_j\}$
            $rvc \leftarrow rvc - q_{pj}$
        **end if**
    **end for**
    **if** ($CS_{cur} < CS_{best}$) **then** //if the current solution is better than the best so far
        $S_{best} \leftarrow S_{cur}$
    **end if**
    $iter \leftarrow iter + 1$
**end while.**
```

4. Computational results

The new "ellipse rule" based heuristic, and the path-scanning heuristics of Golden et al. [8], Pearn [9], and Belenguer et al. [17] were implemented and tested on the 23 problems proposed by DeArmon [21] (*gdb* files), the 34 problems proposed by Benavent et al. [10] (*val* files) and the 24 problems proposed by Belenguer and Benavent [11] (*egl* files), based on real road networks used by Li and Eglese [22]. The first set of problems ranges from 7 to 27 nodes and from 11 to 55 edges. In keeping with general practice, problems 8 and 9 were removed from the 25 originally proposed because they contain inconsistencies. The second set of problems ranges from 24 to 50 nodes and from 34 to 97 edges. The third set of problems ranges from 77 to 140 nodes and from 98 to 190 edges. These data sets were used because they are the standard data sets for testing CARP solution heuristics (data can be downloaded at http://www.uv.es/~belengue/carp.html). These problems were also solved with three of the best known metaheuristics. These include the tabu search procedure of Hertz et al. [13], the genetic algorithm-based procedure of Lacomme et al. [15], and the ant colony approach of Lacomme et al. [16]. The solutions generated by the various heuristics and metaheuristics were compared to each other and to the lower bounds (LB) identified by Belenguer and Benavent [11]. To facilitate comparisons, the heuristics were run on a 1 GHz Pentium III with 368 K RAM which is similar to the computers used by the three metaheuristics.

The following notation will be used in presenting the results of these tests.

PSG: original path-scanning heuristic with 5 criteria [8] (as implemented by Lacomme et al. [15], Belenguer et al. [17], and Evans and Minieka [19]);

PSP (k): path-scanning heuristic modification of Pearn [9] with 5 random selection criteria in each iteration (uniform distribution used) and k runs for each problem (as implemented by Lacomme et al. [15], Belenguer et al. [17], and Evans and Minieka [19]);

RSE (k): path-scanning heuristic with random selection of tied edges and k runs for each problem as proposed and implemented by Belenguer et al. [17];

RSE_ER (k): "ellipse rule" based path-scanning heuristic, with k runs for each problem, introduced in this article;

CARPET: metaheuristic proposed by Hertz et al. [13] based on tabu search (CPU times were scaled to 1 GHz Pentium III PC by Lacomme et al. [15]);

MA: metaheuristic proposed by Lacomme et al. [15] based on genetic algorithms (CPU times were obtained by 1 GHz Pentium III PC);

BACO: metaheuristic proposed by Lacomme et al. [16] based on ant colony optimization (CPU times were obtained by 800 MHz Pentium III PC).

The deviation percentage between the obtained solution cost (SC) and the lower bound (LB), for each problem, are used to evaluate the solution quality (i.e., deviation percentage is equal to $((SC-LB)/LB) \times 100\%$).

Table 1 presents the results obtained by two of the existing path-scanning heuristics, PSP and RSE, and the new approach, RSE_ER, for $k = 1000$, 10 000 and 20 000. The table also includes the results of the RSE_ER heuristic for six values of α (0.5, 1.0, 1.5, 2.0, 2.5, and 3.0). These results indicate that RSE_ER generated better solutions than did either PSP or RSE on every data set for every value of k and for every value of α except for $\alpha = 3$ in the *gdb* and *egl* datasets. For example, the "global average" for the average deviations to the LBs for the various problem sets were 9.19% and 9.17% for PSP and RSE, respectively, versus 6.44%, 5.34%, 5.16%, 6.19%, 7.09%, and 9.93% for the RSE_ER for the six values of α.

Overall, the best results for the RSE_ER heuristic occurred when $\alpha = 1.5$. As reported in the "global average" row, the global averages of the deviations to the LBs for the RSE_ER solutions decreased as α went from 0.5 to 1.5 and then increased as α went from 1.5 to 3.0. In addition, these global averages were best for $\alpha = 1.5$ for all three values of k.

These results are not surprising as increasing α invokes the ellipse rule earlier in the construction of a route. If it is invoked too early (e.g., $\alpha = 2.5$ or 3.0), condition 1 is not restrictive enough and keeps the route close to the depot too early in the route construction. If α is too small (e.g., 0.5), it is overly restrictive on arc selection and keeps the route close to the depot too late in the route construction. The relative effects that α may have on arc selection options can be seen in the "#UE" columns of Table 1. The entries in these columns are the average number of unserved edges that satisfy condition (1) when invoked. As the "global" averages for these values reported in the last row of Table 1 indicate, the number of unserved edges that satisfy condition 1 are 7.46%, 14.32%, 18.48%, 24.86%, 28.88%, and 36.14% for $\alpha = 0.5$, 1.0, 1.5, 2.0, 2.5, and 3.0, respectively.

Further analysis of the average global results ("global average" row of Table 1) for the 81 problems indicates that the results attained by the RSE heuristic are similar to those attained by the PSP heuristic. The RSE results are slightly worse than those for PSP for the *gdb* and *val* files and are slightly better for *egl* files (for all three values of k). These results indicate that random selection among tied edges (e.g., RSE) is as effective as selection based upon "improvement criteria" (e.g., PSP). This is further supported by the results presented in Table 2 comparing the average deviations to the LBs for the PSG heuristic which uses five "improvement criteria" and RSE for $k = 5$. These results indicate that heuristics based upon "greedy" selection criteria should be compared to ones based upon "random" selection as well as others based upon "greedy" criteria. However as discussed earlier, and as shown in Table 1, the introduction of the "ellipse rule"

Table 2
Comparison between heuristics and metaheuristics approaches

	Av. deviation to the LB (%)			Av. CPU time (seconds)		
	gdb	*val*	*egl*	*gdb*	*val*	*egl*
Heuristics						
PSG	10.62	16.34	26.05	0.00	0.01	0.02
RSE (5)	10.59	18.51	25.54	0.00	0.01	0.02
RSE_ER (5)	7.86	15.77	15.84	0.01	0.02	0.03
PSP (10 000)	2.75	6.90	16.91	2.59	8.51	22.15
RSE (10 000)	2.79	6.55	16.63	2.41	8.45	21.85
RSE_ER (10 000)	1.13	4.56	8.95	3.12	6.51	23.04
Metaheuristics						
CARPET	0.48	1.96	4.74	9.03	63.86	[a]
MA	0.15	0.61	2.47	5.29	38.35	526.99
BACO	0.28	0.90	4.11	24.77	276.32	2341.3

[a]Unknown CPU time.

(RSE_ER) greatly improves the quality of the RSE solutions. For example, the global average deviation to the LBs was 9.17% for RSE and 5.16% for RSE_ER with $\alpha = 1.5$ (i.e., the average deviation was globally reduced by about 44%).

Table 2 presents the results obtained by the three path-scanning heuristics, PSG, PSP and RSE, by the new approach, RSE_ER ($\alpha = 1.5$), and also by the best known metaheuristics (CARPET, MA and BACO). RSE and RSE_ER were tested for five iterations in order to attain results comparable with PSG (which is executed five times, one time for each of the five criteria). Conclusions are similar to those obtained from the results presented in Table 1. For example, comparing the ratios of average deviation to the LBs one finds that the ratio of PSG to RSE_ER(5) solutions ranged from 1.04 to 1.64 and the ratio of RSE(5) to RSE_ER(5) ranged from 1.17 to 1.61, for the three problem sets. Similar ratios for the PSP(10 000) to RSE_ER(10 000) ranged from 1.51 to 2.43 and RSE(10 000) to RSE_ER(10 000) ranged from 1.44 to 2.47. The new heuristic generated better solutions on average than did the existing path-scanning heuristics in all three problem sets. The times reported in Table 2 indicate that these improved results were attained with little or no additional CPU time.

As expected, the metaheuristics generated better solutions, on average, than did the path-scanning heuristics. For example, the average ratios of average deviation to LBs of RSE_ER(10 000) to CARPET, MA, and BACO over all data sets were 2.19, 6.21, and 3.76, respectively. These improvements came at a large cost in increased solution times as shown by the ratios of average solution times for the problem sets for the metaheuristics to those for RSE_ER(10 000) which varied from 2.89 to 9.81 for CARPET, 1.70 to 22.87 for MA, and 7.94 to 101.62 for BACO. The importance of these increased solution times is mentioned in the next section.

5. Conclusions

Due to its practicality and computational complexity, the capacitated arc routing problem (CARP) has received considerable attention in the Operations Research literature. Several heuristics and metaheuristics have been developed to solve it. This article introduces a new path-scanning heuristic for the CARP. The heuristic constructs a route by adding feasible edges (i.e., with demand less than or equal to the remaining vehicle capacity) one at a time based on a nearest neighbor strategy. If there is a tie for the nearest neighbor, one of the tied edges is selected at random. As the vehicle approaches the end of a route, an ellipse rule is invoked which constrains the options for the next edge added to those near the shortest path between the last serviced edged and the depot. The new heuristic was tested on three well-known data sets and the solutions were compared to lower bounds obtained by Belenguer and Benavent [11] and to the solutions generated by the three previous best path-scanning heuristics for CARP [8,9,17] and the three best metaheuristics for the problem [13,15,16].

The results of these tests indicate that a random selection of tied edges performs as well as the multi-criterion selection procedures proposed by Golden et al. [8] and Pearn [9]. This supports the findings of Belenguer et al. [17] and suggests that greedy-add heuristics should be compared to random-add heuristics during their testing and evaluation for other problems as well as for CARP. The results also indicate that the new path-scanning heuristic (RSE_ER) generates better quality solutions than do the existing path-scanning heuristics over all three standard problem sets. This improvement (about a 44% reduction in overall average gap to the lower bounds) comes with little or no increase in solution time.

As expected, metaheuristics in general generate better solutions for the CARP than do path-scanning heuristics. However, these improvements come with a cost of increased solution time. For example, on average the metaheuristics increased solution times by factors ranging from 1.70 for the smallest problem instances to 101.62 for the largest instances.

The development of more efficient and effective greedy heuristics for CARP like RSE_ER is an important area of research. This is true for several reasons.

First, solution time is an important criterion in many applications of CARP; especially in those for which edge demand is uncertain and/or varies while the route is being served. In such dynamic and/or stochastic applications, CARP may need to be solved numerous times in “real-time” to incorporate the actual demand encountered as the routes progress. As Table 2 indicates, solution times for the CARP metaheuristics can become quite large for even relatively small (i.e., maximum of 140 nodes and 190 edges) problems like those in the data sets. Real world problems are often much larger. For example, a CARP application to trash collection [4] to part of Coimbra, Portugal (total population of about 120 000 inhabitants), included 756 nodes and 1051 edges. For large networks, metaheuristic solution times may be too great for even static, deterministic applications of CARP.

Second, they are often used to identify “starting” solutions for metaheuristics. Consequently, their solution time (efficiency) and quality (effectiveness) have an important impact on the efficiency and effectiveness of the metaheuristic. Third, the simplicity of greedy heuristics like RSE_ER makes them easier to encode and to modify to accommodate various real-world specifics of the underlying problem. Fourth, path scanning heuristics do not require the determination of various input parameters associated with metaheuristics. Finally, decision makers often find greedy-add heuristics intuitive. As a consequence, they may be more willing to accept them and implement their results.

Our research indicates that the new greedy heuristic presented in this article is the best path-scanning based heuristic to solve CARP. It can be used to either solve instances of CARP directly or to generate starting solutions for various metaheuristics.

References

[1] MacroSys Research and Technology. Logistics costs and U.S. gross domestic product. Federal Highway Administration Department of Transportation; 2005, August 25 ⟨http://ops.fhwa.dot.gov/freight/freight_analysis/econ_methods/lcdp_rep/index.htm⟩.

[2] Assad AA, Golden BL. Arc routing methods and applications. In: Ball MO et al., editor. Handbook in operations research and management science, vol. 8, Elsevier; 1995. p. 375–483.

[3] Dror M. Arc routing: theory, solutions and applications. MA: Kluwer Academic Publishers; 2000.

[4] Santos L, Coutinho-Rodrigues J, Current JR. Implementing a multi-vehicle multi-route spatial decision support system for efficient trash collection in Portugal. Transportation Research Part A: Policy and Practice 2008;42(6):922–34.

[5] Golden BL, Wong RT. Capacitated arc routing problems. Networks 1981;11: 305–15.

[6] Hirabayashi R, Saruwatari Y, Nishida N. Tour construction algorithm for the capacitated arc routing problems. Asia-Pacific Journal Operations Research 1992;9:155–75.

[7] Longo H, Aragão MP, Uchoa E. Solving capacitated arc routing problems using a transformation to the CVRP. Computers and Operations Research 2006;33: 1823–37.

[8] Golden BL, DeArmon J, Baker EK. Computational experiments with algorithms for a class of routing problems. Computers and Operations Research 1983;10: 47–59.

[9] Pearn WL. Approximate solutions for the capacitated arc routing problem. Computers and Operations Research 1989;16:589–600.

[10] Benavent E, Campos V, Corberán A, Mota E. The capacitated arc routing problem: lower bounds. Networks 1992;22:669–90.

[11] Belenguer JM, Benavent E. A cutting plane algorithm for the capacitated arc routing problem. Computers and Operations Research 2003;30(5):705–28.

[12] Coutinho-Rodrigues J, Rodrigues N, Clímaco J. Solving an urban routing problem using heuristics: a successful case study. Belgian Journal of Operations Research, Statistics and Computer Science, Combinatorial Optimization in Applications 1993;33:19–32 [special issue].

[13] Hertz A, Laporte G, Mittaz M. A tabu search heuristic for the capacitated arc routing problem. Operations Research 2000;48:129–35.

[14] Greistorfer P. A tabu scatter search metaheuristic for the capacitated arc routing problem. Computers and Industrial Engineering 2003;44(2):249–66.

[15] Lacomme P, Prins C, Ramdane-chérif W. Competitive memetic algorithms for arc routing problems. Annals of Operations Research 2004;131:159–85.

[16] Lacomme P, Prins C, Tanguy A. First competitive ant colony scheme for the CARP. Research Report LIMOS/RR-04-21; 2004 ⟨http://www.isima.fr/limos/publi/RR-04-21.pdf⟩.

[17] Belenguer JM, Benavent E, Lacomme P, Prins C. Lower and upper bounds for the mixed capacitated arc routing problem. Computers and Operations Research 2006;33:3363–83.

[18] Greistorfer P. Computational experiments with heuristics for a capacitated arc routing problem. In: Derigs U, Bachem A, Drexl A, editors. Operations research proceedings, 1994. Berlin: Springer; 1995. p. 185–90.

[19] Evans J, Minieka E. Optimization algorithms for networks and graphs. second ed., New York: Marcel Dekker; 1992.

[20] Norback JP, Love RF. Geometric approaches to solving the traveling salesman problem. Management Science 1977;23(11):1208–23.

[21] DeArmon JS. A comparison of heuristics for the capacitated Chinese postman problem. Master's thesis, University of Maryland at College Park; 1981.

[22] Li LYO, Eglese RW. An interactive algorithm for vehicle routeing for winter-gritting. Journal of the Operational Research Society 1996;47:217–28.