Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Operations Research 32 (2005) 1165–1179
computers & operations research
www.elsevier.com/locate/dsw

Very large-scale vehicle routing: new test problems, algorithms, and results

Feiyue Li[a], Bruce Golden[b,*], Edward Wasil[c]

[a] *Department of Mathematics, University of Maryland, College Park, MD 20742, USA*
[b] *R.H. Smith School of Business, University of Maryland, College Park, MD 20742, USA*
[c] *Kogod School of Business, American University, Washington, DC 20016, USA*

Abstract

The standard vehicle routing problem was introduced in the OR/MS literature about 45 years ago. Since then, the vehicle routing problem has attracted an enormous amount of research attention. In the late 1990s, large vehicle routing problem instances with nearly 500 customers were generated and solved using metaheuristics. In this paper, we focus on *very* large vehicle routing problems. Our contributions are threefold. First, we present problem instances with as many as 1200 customers along with estimated solutions. Second, we introduce the variable-length neighbor list as a tool to reduce the number of unproductive computations. Third, we apply record-to-record travel with a variable-length neighbor list to 32 problem instances and obtain high-quality solutions, very quickly.

Keywords: Vehicle routing problem; Metaheuristics

1. Introduction

The standard version of the capacitated vehicle routing problem (VRP) is easy to state and very difficult to solve: Generate a sequence of deliveries for each vehicle in a homogeneous fleet based at a single depot so that all customers are serviced and the total distance traveled by the fleet is minimized. Each vehicle has a fixed capacity and must leave from and return to the depot. Each vehicle might have a route-length restriction that limits the maximum distance it can travel. Each customer has a known demand and is serviced by exactly one visit of a single vehicle.

* Corresponding author. Tel.: +1-301-405-2232; fax: +1-301-405-3364.
E-mail address: bgolden@rhsmith.umd.edu (B. Golden).

0305-0548/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2003.10.002

In the last 5 years, there has been a great deal of computational effort devoted to solving the 20 large-scale vehicle routing problems (denoted by LSVRPs) developed by Golden et al. [1]. These benchmark problems have 200–483 customers. Eight problems have route-length restrictions. Each problem has a geometric symmetry that allows a user to estimate a high-quality solution (eight problems have customers located in concentric circles around the depot, four problems have customers located in concentric squares with the depot located in one corner, four problems have customers located in concentric squares around the depot, and four problems have customers located in a six-pointed star around the depot).

Researchers have applied general-purpose metaheuristics including deterministic annealing and tabu search to the 20 LSVRPs and have generated high-quality solutions. In Section 2, we review six algorithms that have been used by researchers to solve the large-scale problems, develop an improved version of our record-to-record travel algorithm that uses a variable-length neighbor list, and report computational results for all seven procedures.

In Section 3, we develop a new set of 12 very large-scale vehicle routing problems (denoted by VLSVRPs). The problems have 560–1200 customers, route-length restrictions, and exhibit geometric symmetry. We report computational experience with our variable-length neighbor list record-to-record travel algorithm and compare results to the visually estimated solutions.

In Section 4, we summarize our findings and suggest areas for future work.

2. Solving LSVRPs: new algorithms and results

In the last 5 years, a variety of algorithms have been developed to solve the 20 LSVRPs. Researchers have used deterministic variants of simulated annealing (record-to-record travel, back tracking adaptive threshold accepting, and list-based threshold accepting) and variants of tabu search (network flow-based tabu search, adaptive memory-based tabu search, and granular tabu search). We summarize these six algorithms in Table 1.

We set out to develop an improved version of the record-to-record travel algorithm (denoted by RTR) described in Golden et al. [1] that would be accurate, fast, simple, and flexible (this was motivated by the work of Cordeau et al. [2]). The details of our improved algorithm are given in Table 2. Our algorithm (denoted by VRTR) uses a variable-length neighbor list. The idea is to consider only a fixed number of neighbors for each node when making one-point, two-point, and two-opt moves. We start with a traditional fixed-length neighbor list with $k = 40$ nearest neighbors (fixed-length neighbor lists are not new; we used this type of implementation in work on the traveling salesman problem [3]). For each node i, we remove all edges (for nodes in the neighbor list) with length greater than $\alpha \times L$, where L is the maximum length among all edges in i's neighbor list. When α is equal to 1, we consider $k = 40$ edges. When α is less than 1, we consider fewer edges. In general, we expect that as α decreases, so does running time and accuracy suffers. Our variable-length neighbor list is similar to the granular neighborhood developed by Toth and Vigo [6] for the VRP.

We run our record-to-record travel algorithm three times with different starting solutions that have been generated by the modified Clarke and Wright algorithm with parameter λ. The three values for λ that we use in our algorithm (see Step 0 in Table 2) were determined by a search over values

Table 1
Six algorithms for solving the large-scale vehicle routing problem

Authors	Algorithm	Comments
Golden et al. [1]	Record-to-record travel	An initial solution is generated by the Clarke and Wright algorithm. Feasible one-point moves are made using record-to-record travel (uphill moves allowed). Points are exchanged on different routes (two-point exchange) while feasibility is maintained (uphill moves allowed). Routes are cleaned up (only downhill moves allowed). A local reinitialization allows individual routes to be resequenced and the process of one-point moves, two-point exchanges, and clean-up is repeated. At the end, global reinitialization perturbs the best solution and the process of one-point moves, two-point exchanges, and clean-up is repeated.
Golden et al. [1]	Network flow-based tabu search	The authors used the algorithm of Xu and Kelly [4] where insertion and swap moves are controlled by a network flow model. Infeasible solutions that violate capacity are allowed.
Tarantilis and Kiranoudis [5]	Adaptive memory-based tabu search	The key idea is to extract a sequence of points (called bones) from a set of solutions and generate a route using adaptive memory. If a large number of routes in the set of solutions contains a specific bone, then the authors argue that this bone should be included in a route that appears in a high-quality solution. The BoneRoute algorithm has two phases. In Phase I, a set of initial solutions is generated using weighted savings. The solutions are improved using a standard tabu search algorithm. In Phase II, promising bones are extracted, a solution is generated and improved using tabu search, and the set of solutions is updated.
Toth and Vigo [6]	Granular tabu search	The authors define a granular neighborhood for the VRP by considering short edges (these are edges whose lengths are less than a threshold value) and by typically not considering long edges. Granular neighborhoods are similar in concept to neighbor list strategies. A granular tabu search algorithm is developed. The algorithm starts with a Clarke and Wright solution (infeasible solutions are allowed) and uses granularity-based intensification and diversification during the search.
Tarantilis [7]	Backtracking adaptive threshold accepting List-based threshold accepting	Threshold accepting is a deterministic variant of simulated annealing in which a threshold value T is specified as the upper bound on the amount of objective function increase allowed (uphill moves can be made). In the backtracking algorithm, T is allowed to increase during the search. In the list-based algorithm, a list of values for T is used during the search.

Table 2
Variable-length neighbor list record-to-record travel algorithm

Step 0: Initialization.
Parameters are I, K, and λ.
Set $I = 30$, $K = 5$, and $\lambda \in \{0.6, 1.4, 1.6\}$.
Step 1: Starting solution.
Generate an initial feasible solution using the modified
Clarke and Wright algorithm with parameter λ.
Set Record = objective function value of the current solution.
Set Deviation = 0.01 × Record.
Step 2: Improve the current solution.
For $i = 1$ to I (I loop)
Do One-Point Move with record-to-record travel, Two-Point Move
with record-to-record travel between routes, and Two-opt Move
with record-to-record travel. Feasibility must be maintained.
If no feasible record-to-record travel move is made, go to Step 3.
If a new record is produced, update Record and Deviation.
End I loop
Step 3: For the current solution, apply One-Point Move (within and between routes),
Two-Point Move (between routes), Two-opt Move (between routes), and
Two-opt Move (within and between routes). Only downhill moves are allowed.
If a new record is produced, update Record and Deviation.
Step 4: Repeat for K consecutive iterations.
If no new record is produced, go to Step 5.
Otherwise, go to Step 2.
Step 5: Perturb the solution.
Compare the solution generated after perturbation to the best solution
generated so far and keep the better solution.
Step 6: Keep the best solution generated so far.
Go to Step 1 and select a new value for λ.

One-point Move with record-to-record travel
For each node i in the current solution (I loop)
For each edge j in the current solution whose one end is in node i's neighbor list (J loop)
Calculate the savings of inserting node i between edge j
if such a move is feasible.
If the savings is greater than or equal to zero, then make
the move and continue with the I loop.
Store the value of the largest savings so far.
If the tour length − largest savings from the J loop ⩽ Record + Deviation,
then make the move and continue with the I loop.
End J loop
End I loop

Two-point Move with record-to-record travel
For each node i in the current solution (I loop)
For each node j from the neighbor list of node i (J loop)
Calculate the savings of exchanging i and j.
If the savings is greater than or equal to zero, then
make the exchange and go to the I loop.
Store the value of the largest savings so far.

Table 2 (*continued*)

```
            If the tour length − largest savings from the J loop ⩽ Record + Deviation,
            then make the exchange and continue with the I loop.
            End J loop
End I loop

Two-opt Move with record-to-record travel
For each edge i in the current solution (I loop)
            For each node j from the neighbor list of node i (J loop)
                        Calculate the savings of the two-opt move with i and j
                        if the move is feasible, that is, both capacity and
                        distance constraints are satisfied.
                        If the savings is greater than or equal to zero, then
                        make the move and go to the I loop.
                        Store the value of the largest savings so far.
            If the tour length − largest savings from the J loop ⩽ Record + Deviation,
            then make the move and continue with the I loop.

Perturb a feasible solution
For each node i, define r(i) = d(i)/s(i), where d(i) is the demand of customer i and
s(i) = dist (prior (i), i) + dist (i, next(i)) − dist (prior (i), next (i)), where dist (i, j) is the
distance from node i to node j, prior (i) is the node serviced before node i, and next (i) is
the node serviced after node i.
Sort the r(i) values in ascending order and select the first M nodes where M = min(20, n/10),
where n is the number of nodes. Try to insert these nodes, one by one, into a new location on
a tour using least-cost insertion while maintaining feasibility.
```

ranging from 0.6 to 2.0 that we conducted in preliminary computational experiments. The values 0.6, 1.4, and 1.6 generated reasonably good results.

There are two key differences between VRTR and RTR. First, VRTR considers two-opt moves between and within routes; RTR considers two-opt moves only within routes. Second, VRTR uses a variable-length neighbor list that should help focus the algorithm on promising moves and speed up the search procedure; RTR does not use a neighbor list.

In Table 3, we present the estimated solution values and solution values generated by seven algorithms on the 20 LSVRPs (we maintain the same ordering of the problems as given in Golden et al. [1]). The first eight problems (problems 1–8) have route-length restrictions.

By exploiting the geometric symmetry of a problem, a high-quality estimated solution can be produced visually. In the column labeled ESTG, we give the estimated solutions produced by Golden et al. [1]. Tarantilis and Kiranoudis [5] improved the estimated solutions for the first eight problems and these are given in the column labeled ESTTK.

The results for six algorithms (RTR, NFTS, BR, GTS, BATA, and LBTA) are taken from the literature and represent solutions generated using a single set of parameter values. For example, there are seven parameters whose values need to be set in the BoneRoute algorithm (BR) and the authors use one "standard setting" for each parameter to produce the results shown in Table 3.

We experimented with a range of values from 0.40 to 1 for the α parameter in VRTR. In Table 3, we show results for the default value of 1 (use all edges in the neighbor list with $k = 40$; we

Table 3
Estimated solution values and solution values generated by seven algorithms on 20 LSVRPs

Problem	n	ESTG	ESTTK	RTR	NFTS	BR	GTS	BATA	LBTA	VRTR
1	240	5859.62	5676.97	5834.60		5676.97	5736.15	5683.63	5680.16	**5666.42**
2	320	8566.04	**8447.92**	9002.26	8570.28	8512.64	8553.03	8528.80	8512.64	8469.32
3	400	11649.06	**11036.23**	11879.95	11880.37	11199.72	11402.75	11199.72	11190.38	11145.80
4	480	15108.68	**13624.53**	14639.32	15250.78	13637.53	14910.62	13661.16	13706.78	13758.08
5	200	8631.64	**6460.98**	6702.73	7361.29	**6460.98**	6697.53	6466.68	**6460.98**	6478.09
6	280	9843.01	**8412.88**	9016.93	9088.66	8429.28	8963.32	8429.28	8427.72	8539.61
7	360	11047.69	**10195.56**	11213.31	11411.85	10216.50	10547.44	10297.27	10274.19	10289.72
8	440	12250.06	**11828.78**	12514.20	12825.00	11936.16	12036.24	11953.93	11968.93	11920.52
9	255	678.82		**587.09**	589.10		593.35	596.92	595.35	588.25
10	323	865.50		749.15	**746.56**		751.66	765.03	764.88	749.49
11	399	1074.80		934.33	932.68		936.04	945.20	945.09	**925.91**
12	483	1306.73		1137.18	1140.72		1147.14	1143.39	1143.74	**1128.03**
13	252	956.04		881.04	881.07		868.80	872.66	871.97	**865.20**
14	320	1220.27		1103.69	1118.09		**1096.18**	1102.40	1102.66	1097.78
15	396	1516.48		1364.23	1377.79		1369.44	1384.04	1385.59	**1361.41**
16	480	1844.67		1657.93	1656.66		1652.32	1679.50	1677.25	**1635.58**
17	240	796.13		720.44			**711.07**	718.16	717.40	711.74
18	300	1133.25		1029.21			1016.83	1030.54	1032.07	**1010.32**
19	360	1554.64		1403.05			1400.96	1408.62	1406.47	**1382.59**
20	420	2081.38		1875.17			1915.83	1872.23	1872.87	**1850.92**

ESTG, estimated solution from Golden et al. [1]. ESTTK, estimated solution from Tarantilis and Kiranoudis [5]. RTR, record-to-record travel solution from Golden et al. [1]. NFTS, network flow-based tabu search solution of Xu and Kelly from Golden et al. [1]. BR, BoneRoute solution from Tarantilis and Kiranoudis [5]. GTS, granular tabu search solution from Toth and Vigo [6]. BATA, backtracking adaptive threshold accepting solution from Tarantilis [7]. LBTA, list-based threshold accepting solution from Tarantilis [7]. VRTR, variable-length neighbor list record-to-record travel solution ($\alpha=1$). **Bold**, Best solution.

point out that, when $\alpha = 0.40$, we use about 50–60% of the edges and when $\alpha = 0.60$, we use about 80–90% of the edges).

In Table 3, among the seven algorithms, VRTR performs best—it generates the best solution to nine problems. GTS is second best—it generates the best solution to two problems. The estimated solutions to problems 2–8 given by Tarantilis and Kiranoudis [5] are very good—no algorithm produced a better solution to these seven problems.

In Table 4, we provide additional computational results on the 20 LSVRPs. In the columns marked Best known and Source, we provide the best-known solution to each problem and the source of the solution. Seven of the best-known solutions are visually estimated solutions (ESTTK, EST), three are generated by VRTR with two different values $(0.60, 0.65)$ for the α parameter, and 10 are generated by RTR (these solutions were generated during the overall computational testing with a variety of settings for the parameters). We compute the percent above the best-known solution for each solution generated by two visual procedures (ESTG and ESTTK) and seven algorithms (RTR, NFTS, BR, GTS, BATA, LBTA, and VRTR with $\alpha = 1$, 0.60, and 0.40), and for the best solutions found by granular tabu search (these are given in the column marked BGTS; recall that GTS gives results

Table 4
Computational results on 20 LSVRPs: percent above best-known solution and computing time

Problem	n	Best known	Source	ESTG	ESTTK	RTR	NFTS	BR	GTS	BGTS	BATA	LBTA	VRTR		
													$\alpha = 1$	$\alpha = 0.6$	$\alpha = 0.4$
1	240	5639.36	VRTR, $\alpha = 0.65$	3.91	0.67	3.46		0.67	1.72	1.72	0.79	0.72	0.48	0.65	0.60
2	320	8447.92	ESTTK	1.40	0.00	6.56	1.45	0.77	1.24	1.24	0.96	0.77	0.25	0.39	0.39
3	400	11036.23	ESTTK	5.55	0.00	7.65	7.65	1.48	3.32	3.32	1.48	1.40	0.99	0.14	1.42
4	480	13624.53	ESTTK	10.89	0.00	7.45	11.94	0.10	9.44	5.22	0.27	0.60	0.98	1.28	1.03
5	200	6460.98	ESTTK	33.60	0.00	3.74	13.93	0.00	3.66	3.40	0.09	0.00	0.26	0.26	0.35
6	280	8412.88	ESTTK	17.00	0.00	7.18	8.03	0.19	6.54	3.56	0.19	0.18	1.51	1.48	1.50
7	360	10195.56	ESTTK	8.36	0.00	9.98	11.93	0.21	3.45	3.14	1.00	0.77	0.92	0.98	0.52
8	440	11696.55	ORTR	4.73	1.13	6.99	9.65	2.05	2.90	2.90	2.20	2.33	1.91	2.13	0.65
9	255	585.54	ORTR	15.93		0.26	0.61		1.33	1.33	1.94	1.68	0.46	0.80	0.55
10	323	743.17	ORTR	16.46		0.80	0.46		1.14	1.14	2.94	2.92	0.85	0.78	0.78
11	399	923.11	ORTR	16.43		1.22	1.04		1.40	1.40	2.39	2.38	0.30	0.31	1.11
12	483	1116.22	VRTR, $\alpha = 0.65$	17.07		1.88	2.19		2.77	1.78	2.43	2.47	1.06	0.86	1.07
13	252	862.38	ORTR	10.86		2.16	2.17		0.74	0.73	1.19	1.11	0.33	0.98	1.41
14	320	1083.65	EST	12.61		1.85	3.18		1.16	1.16	1.73	1.75	1.30	0.98	1.11
15	396	1351.35	ORTR	12.22		0.95	1.96		1.34	0.89	2.42	2.53	0.74	0.24	0.24
16	480	1632.47	ORTR	13.00		1.56	1.48		1.22	1.10	2.88	2.74	0.19	0.51	0.66
17	240	708.63	ORTR	12.35		1.67			0.34	0.18	1.34	1.24	0.44	0.44	0.59
18	300	1007.86	ORTR	12.44		2.12			0.89	0.69	2.25	2.40	0.24	0.24	0.24
19	360	1378.20	ORTR	12.80		1.80			1.65	0.79	2.21	2.05	0.32	0.32	0.41
20	420	1840.66	VRTR, $\alpha = 0.60$	13.08		1.87			4.08	4.08	1.72	1.75	0.56	0.00	0.80
Average percent above best-known solution				12.53	0.22	3.56	5.18	0.68	2.52	1.99	1.62	1.59	0.70	0.69	0.77
Average computing time (min)						37.15	1825.59	42.05	17.55		18.41	17.81	1.13	0.97	0.68

EST, estimated solution by Li, Golden, and Wasil. ESTG, estimated solution from Golden et al. [1]. ESTTK, estimated solution from Tarantilis and Kiranoudis [5]. RTR, record-to-record travel solution from Golden et al. [1]; Pentium 100 MHz CPU. NFTS, network flow-based tabu search solution of Xu and Kelly from Golden et al. [1]; DEC Alpha Workstation. BR, BoneRoute solution from Tarantilis and Kiranoudis [5]; Pentium 400 MHz CPU. GTS, granular tabu search solution from Toth and Vigo [6]; Pentium 200 MHz CPU. BGTS, best solution found by granular tabu search over all testing from Toth and Vigo [6]. BATA, backtracking adaptive threshold accepting solution from Tarantilis [7]; Pentium 233 MHz CPU. LBTA, list-based threshold accepting solution from Tarantilis [7]; Pentium 233 MHz CPU. VRTR, variable-length neighbor list record-to-record travel solution; Athlon 1 GHz CPU. ORTR, record-to-record travel solution from other experiments by Li, Golden, and Wasil.

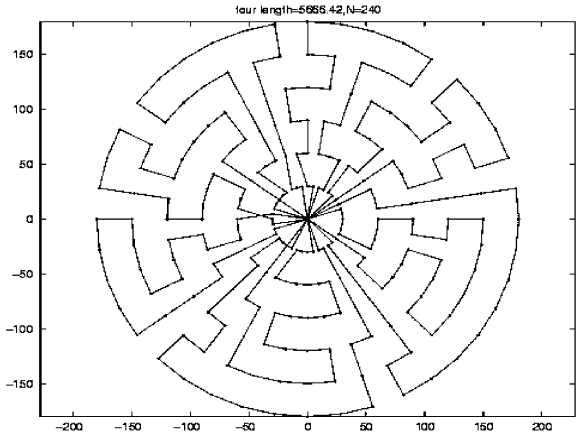

Fig. 1. Best-known solution produced by VRTR with $\alpha = 1$ for the large-scale vehicle routing problem with 240 customers.

using the *same* set of parameter values for each problem, while the parameter settings in BGTS can vary).

In Table 4, over all 20 problems, we see that VRTR with the default value of $\alpha = 1$ generates very high-quality solutions (on average within 0.70% of the best-known solution) in a small amount of computing time (on average 1.13 min per problem) and clearly outperforms RTR, GTS, BGTS, BATA, and LBTA. VRTR with smaller values of the α parameter (0.60 and 0.40) also produces very high-quality solutions (on average within 0.69 and 0.77 of the best-known solution, respectively) very quickly (average of 0.97 min and 0.68 min per problem, respectively).

In Fig. 1, we show the best-known solution produced by VRTR with $\alpha = 1$ for the problem with 240 customers.

3. Very large-scale vehicle routing: new problems and results

We have generated a new set of 12 very large-scale vehicle routing problems with route-length restrictions. These problems have 560–1200 customers. The problem generator is given in the Appendix. Each problem exhibits a geometric symmetry that allows us to visually estimate a high-quality solution. In Fig. 2, we show our estimated solutions for the 12 VLSVRPs.

In Table 5, we present the estimated solution values and the solution values generated by VRTR for three values of the α parameter $(1, 0.60, 0.40)$. In addition, we show the percent above the best-known solution, and the average computing time.

Over all 12 problems, we see that VRTR with the default value of $\alpha = 1$ generates very high-quality solutions (on average within 1.10% of the best-known solution) in a small amount of computing time (on average 3.16 min per problem). VRTR with a value of $\alpha = 0.60$ also produces very high-quality solutions (on average within 1.20% of the best-known solution) very quickly (on average 2.94 min

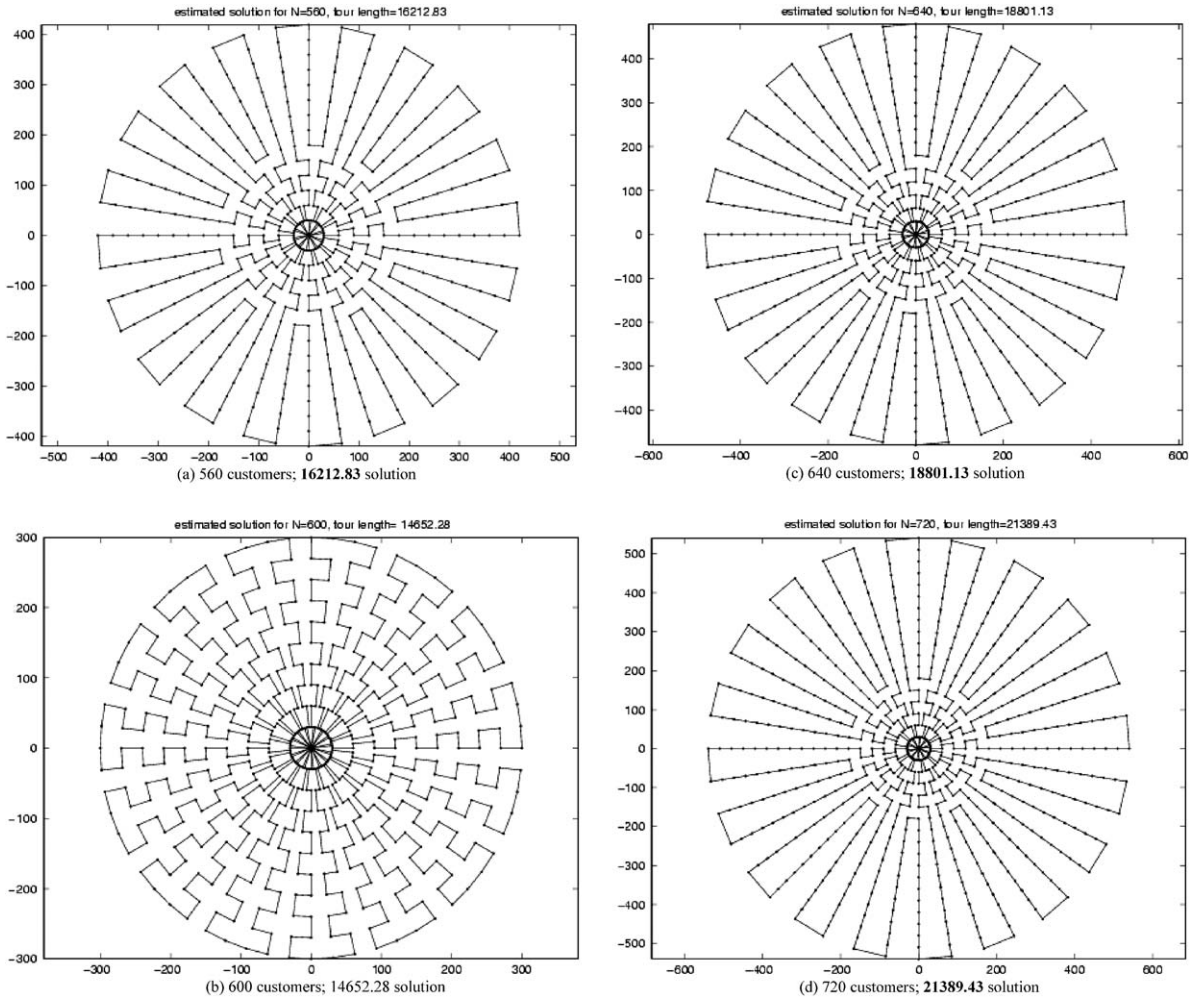

Fig. 2. Estimated solutions for the 12 VLSVRPs where **bold** denotes a best-known solution. (a) 560 customers; **16212.83** solution. (b) 600 customers; 14652.28 solution. (c) 640 customers; **18801.13** solution. (d) 720 customers; **21389.43** solution. (e) 760 customers; **17053.26** solution. (f) 800 customers; **23977.74** solution. (g) 840 customers; 18253.55 solution. (h) 880 customers; **26566.04** solution. (i) 960 customers; **29154.34** solution. (j) 1040 customers; **31742.64** solution. (k) 1120 customers; **34330.94** solution. (l) 1200 customers; **36919.24** solution.

per problem). VRTR with a value of $\alpha = 0.40$ is very fast (on average 2.08 min per problem), but not highly accurate (on average within 2.28% of the best-known solution) especially on the three largest problems.

In Fig. 3, we show the solutions produced by VRTR for the problem with 640 customers. All three values of α produce very good solutions with $\alpha = 0.40$ generating a solution that is only 0.20% above the best-known solution.

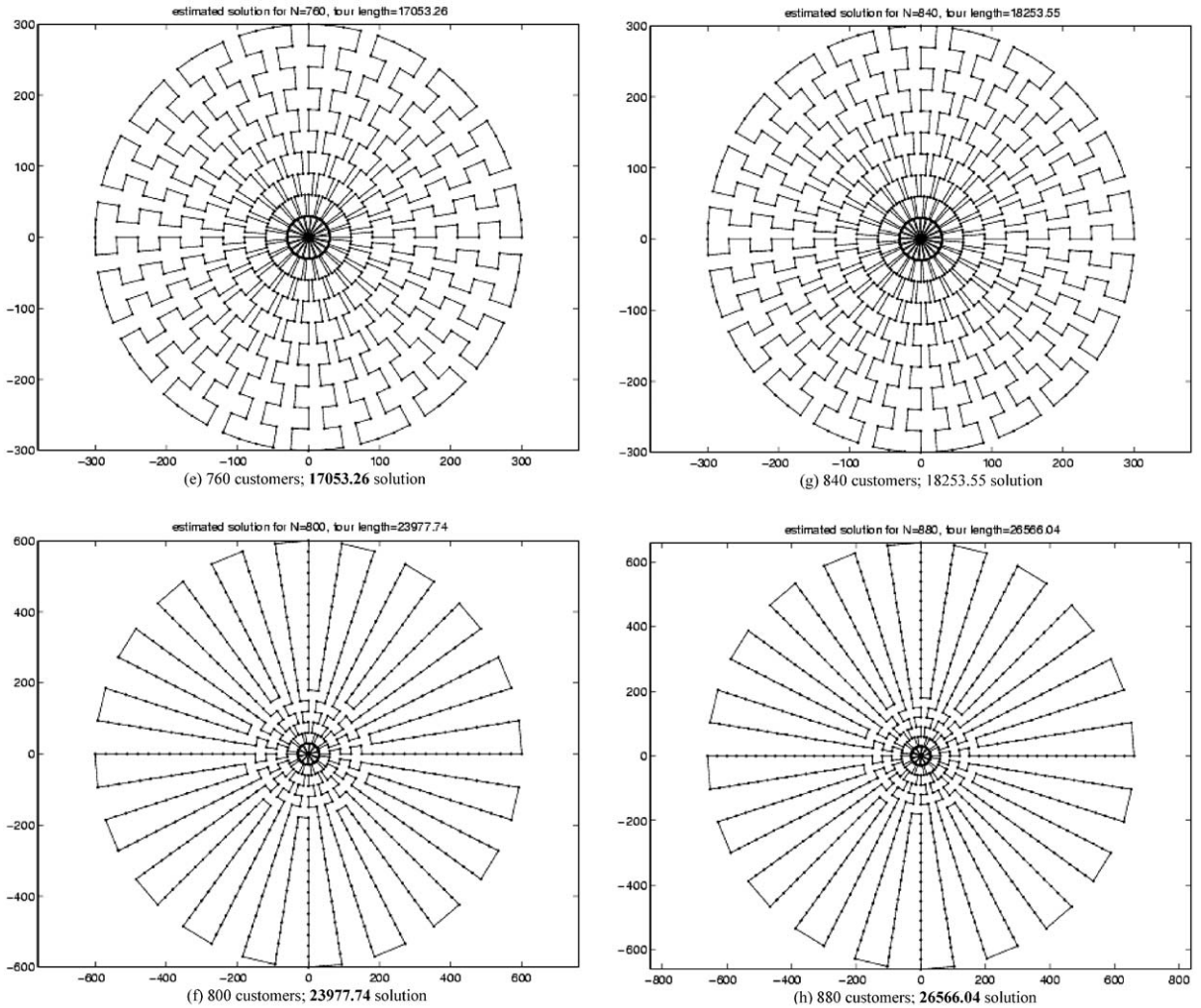

Fig. 2. *continued.*

4. Conclusions

In this paper, we reviewed procedures for solving large-scale vehicle routing problems and developed a record-to-record travel algorithm that uses a variable-length neighbor list. Our algorithm was very fast and highly accurate in solving 20 LSVRPs.

We generated a new set of 12 very large-scale VRPs with 560–1200 customers that are some of the largest test instances in the literature. A solution to each problem can be estimated visually. We applied our record-to-record travel algorithm to these 12 VLSVRPs and found that it produced solutions quickly and accurately.

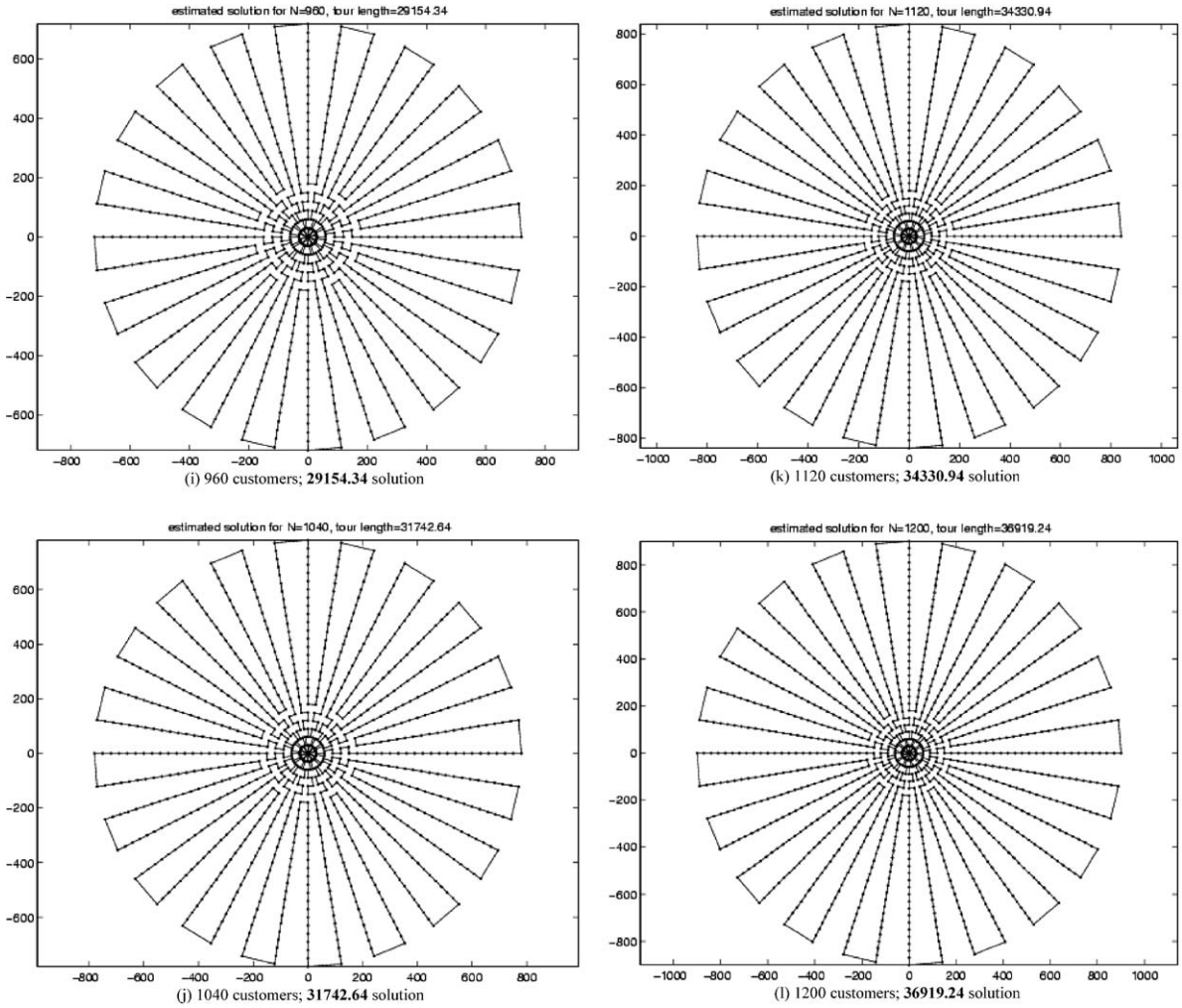

Fig. 2. *continued.*

Finally, at the suggestion of a referee, we applied VRTR without any additional fine-tuning using the three values of the α parameter $(1, 0.60, 0.40)$ to the seven, small-scale benchmark VRPs of Christofides et al. [9]. These seven problems do not have service times for customers. In Table 6, we present the solution values generated by VRTR and the granular tabu search procedure (GTS) of Toth and Vigo [6]. In addition, we show the average percent above the best-known solution, and the average computing time.

Over all seven small-scale problems, we see that VRTR with the default value of $\alpha = 1$ generates very high-quality solutions (on average within 0.62% of the best-known solution) in a small amount of computing time (on average 0.41 min per problem). All three versions of VRTR are quick and accurate, and very competitive with GTS.

Table 5
Computational results on 12 VLSVRPs: solution value, percent above best-known solution, and computing time

Problem	n	Best known	Source	Solution value				Percent above best-known solution			
					VRTR				VRTR		
				EST	$\alpha = 1$	$\alpha = 0.6$	$\alpha = 0.4$	EST	$\alpha = 1$	$\alpha = 0.6$	$\alpha = 0.4$
21	560	16212.83	EST	16212.83	16602.99	16627.22	16739.25	0.00	2.41	2.56	3.25
22	600	14641.64	ORTR	14652.28	14651.27	14655.60	14669.39	0.07	0.07	0.10	0.19
23	640	18801.13	EST	18801.13	19005.37	19094.98	18838.62	0.00	1.09	1.56	0.20
24	720	21389.43	EST	21389.43	21784.43	21616.25	21932.57	0.00	1.85	1.06	2.54
25	760	17053.26	EST	17053.26	17151.43	17163.31	17146.41	0.00	0.58	0.65	0.55
26	800	23977.74	EST	23977.74	24189.66	24200.27	24009.74	0.00	0.88	0.93	0.13
27	840	17651.60	ORTR	18253.55	17823.40	17936.25	17901.56	3.41	0.97	1.61	1.42
28	880	26566.04	EST	26566.04	26606.11	26784.38	26787.38	0.00	0.15	0.82	0.83
29	960	29154.34	EST	29154.34	29181.21	29183.58	29401.90	0.00	0.09	0.10	0.85
30	1040	31742.64	EST	31742.64	31976.73	31961.58	33246.60	0.00	0.74	0.69	4.74
31	1120	34330.94	EST	34330.94	35369.17	35355.50	36339.65	0.00	3.02	2.98	5.85
32	1200	36919.24	EST	36919.24	37421.44	37410.84	39415.85	0.00	1.36	1.33	6.76
Average percent above best-known solution								0.29	1.10	1.20	2.28
Average computing time (min)									3.16	2.94	2.08

EST, estimated solution by Li, Golden, and Wasil. VRTR, variable-length neighbor list record-to-record travel solution; Athlon 1 GHz CPU. ORTR, record-to-record travel solution from other experiments by Li, Golden, and Wasil.

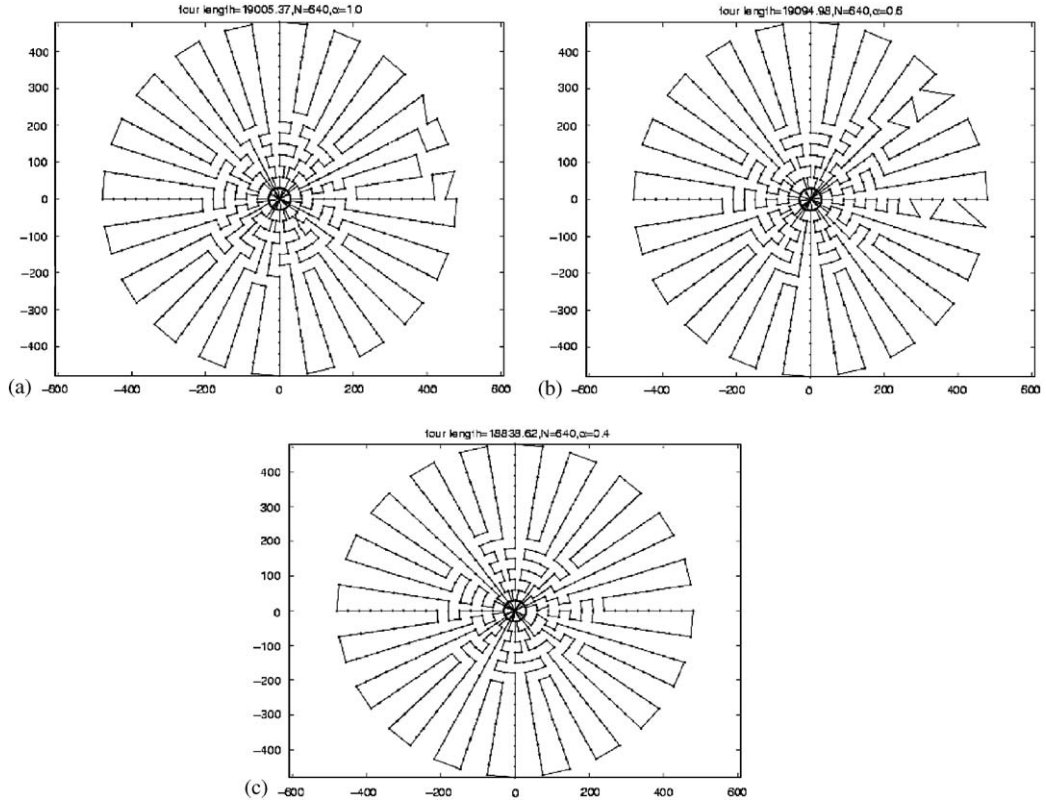

Fig. 3. Solutions produced by VRTR with (a) $\alpha = 1$, (b) $\alpha = 0.60$, and (c) $\alpha = 0.40$ for the very large-scale vehicle routing problem with 640 customers.

Table 6
Solutions generated by VRTR and GTS on seven small-scale VRPs

Problem	n	Best known	VRTR			
			$\alpha = 1$	$\alpha = 0.6$	$\alpha = 0.4$	GTS
1	50	524.61	**524.61**	**524.61**	**524.61**	**524.61**
2	75	835.26	846.64	838.60	836.18	838.60
3	100	826.14	**826.14**	**826.14**	827.39	828.56
4	150	1028.42	1036.00	1044.36	1045.36	1033.21
5	199	1291.45	1318.70	1322.56	1303.47	1318.25
11	120	1042.11	1043.50	**1042.11**	**1042.11**	1042.87
12	100	819.56	**819.56**	**819.56**	**819.56**	**819.56**
Average percent above best-known solution			0.62	0.62	0.41	0.47
Average computing time (min)			0.41	0.35	0.32	3.10

Best known, Best-known solutions from Golden et al. [1] and Gendreau et al. [8]. VRTR, variable-length neighbor list record-to-record travel solution; Athlon 1 GHz CPU. GTS, granular tabu search solution from Toth and Vigo [6]; Pentium 200 MHz CPU. **Bold**, best-known solution.

Appendix A. VLSVRP generator

$(x(i), y(i))$ is the coordinate of customer i, where $i = 0$ is the depot
$q(i)$ is the demand of customer i
A and B are parameters that determine the number of customers n, where $n = A \times B$
All data recorded to four decimal places

```
begin
  ω = 0
  x(ω) = 0, y(ω) = 0, q(ω) = 0
  for k: = 1 to B do
    begin
      γ = 30k
      for i: = 1 to A do
        begin
          ω = ω + 1
          x(ω) = γ cos[2(i − 1)π/A]
          y(ω) = γ sin[2(i − 1)π/A]
          if mod(i, 4) = 2 or 3
            then q(ω) = 30
            else q(ω) = 10
        end
    end
end
```

Problem	A	B	n	Vehicle capacity	Maximum route length
21	40	14	560	1200	1800
22	60	10	600	900	1000
23	40	16	640	1400	2200
24	40	18	720	1500	2400
25	76	10	760	900	900
26	40	20	800	1700	2500
27	84	10	840	900	900
28	40	22	880	1800	2800
29	40	24	960	2000	3000
30	40	26	1040	2100	3200
31	40	28	1120	2300	3500
32	40	30	1200	2500	3600

References

[1] Golden BL, Wasil EA, Kelly JP, Chao I-M. The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In: Crainic T, Laporte G, editors. Fleet management and logistics. Boston, MA: Kluwer; 1998. p. 33–56.

[2] Cordeau J-F, Gendreau M, Laporte G, Potvin J-Y, Semet F. A guide to vehicle routing heuristics. Journal of the Operational Research Society 2002;53:512–22.
[3] Coy SP, Golden BL, Runger GC, Wasil EA. See the forest before the trees: fine-tuned learning and its application to the traveling salesman problem. IEEE Transactions on Systems, Man, and Cybernetics 1998;28(4):454–64.
[4] Xu J, Kelly J. A network flow-based tabu search heuristic for the vehicle routing problem. Transportation Science 1996;30:379–93.
[5] Tarantilis CD, Kiranoudis CT. BoneRoute: an adaptive memory-based method for effective fleet management. Annals of Operations Research 2002;115:227–41.
[6] Toth P, Vigo D. The granular tabu search and its application to the vehicle-routing problem, INFORMS Journal on Computing, 2003, forthcoming in vol. 15, No. 4.
[7] Tarantilis CD. Threshold accepting-based heuristics for solving large scale vehicle routing problems. Working Paper, National Technical University of Athens, Athens, Greece, 2003.
[8] Gendreau M, Laporte G, Potvin J-Y. Metaheuristics for the capacitated VRP. In: Toth P, Vigo D, editors. The vehicle routing problem. Philadelphia, PA: SIAM; 2002. p. 129–54.
[9] Christofides N, Mingozzi A, Toth P. The vehicle routing problem. In: Christofides N, Mingozzi A, Toth P, Sandi C, editors. Combinatorial optimization. UK: Chichester: Wiley; 1979. p. 315–38.