Journal of the Operational Research Society (2006) 57, 111–116
www.palgrave-journals.com/jors

A reactive tabu search for the vehicle routing problem

NA Wassan*

University of Kent, Canterbury, UK

The classical vehicle routing problem (VRP) involves determining a fleet of homogeneous size vehicles and designing an associated set of routes that minimizes the total cost. Our tabu search (TS) algorithm to solve the VRP is based on reactive tabu search (RTS) with a new *escape mechanism*, which manipulates different neighbourhood schemes in a very sophisticated way in order to get a balanced intensification and diversification continuously during the search process. We compare our algorithm with the best methods in the literature using different data sets and report results including new best known solutions for several well-known benchmark problems.
Journal of the Operational Research Society (2006) **57,** 111–116. doi:10.1057/palgrave.jors.2601957
Published online 6 July 2005

Keywords: vehicle routing meta-heuristic; tabu search

Introduction

In the vehicle routing problem (VRP), we are given a set of customers, a set of homogeneous vehicles and a depot. The problem is to design least-cost vehicle routes originating and terminating at the depot to service customers with known demands, subject to vehicle capacity and other relevant constraints. For more details and information of VRPs, see Bodin *et al*[1] and Laporte and Osman.[2]

A review of the VRP literature reveals that the problem has been studied extensively. A number of exact and approximate algorithms exist. While exact algorithms can only solve relatively small-size problems, several heuristic algorithms have proved very successful. Tabu search (TS) is declared to be the best meta-heuristic for the VRP by Cordeau *et al*[3] and Laporte *et al.*[4] In the following sections, we describe briefly the most successful TS algorithms for the VRP, some other relevant studies, and provide details of our implementation. At the end, the computational results are compared and analysed.

Osman[5] developed a standard classical TS algorithm that used nodes involved in a move to prevent reversals. The tabu tenure (tt) value is systematically varied between three predefined values at every $2tt$ iterations. A rather more sophisticated TS implementation is given by Gendreau *et al*[6] that considers 1-insert type moves with the GENI (Generalized Insertion Procedure) algorithm. To prevent cycles, the procedure uses nodes involved in moves and the indices of routes, while the tt value is randomly varied between predefined limits. Taillard[7] partitions the large problems into several subproblems, solving each of these using a robust TS scheme, with the tt value being randomly varied between $0.4n$ and $0.6n$. It is worth to mention two ejection-chain model-based TS implementations by Rego and Roucairol,[8] and Xu and Kelly.[9] Among other VRP studies that we will be comparing in this paper are the *decomposition algorithm* of Noon *et al*,[10] the *minimum K-trees algorithm* of Fisher[11] and a *probabilistic technique* of Rochat and Taillard.[12]

Some studies suggest that TS works well with good diversification and intensification strategies. However, during the experiments in Wassan and Osman,[13] it was noticed that too much diversification and/or intensification could bring deterioration to the search process and this may cause late convergence. The current implementation is designed to achieve the maximum amount of correct balance between these two vital factors of TS search process.

Details of the TS Algorithm

Initial solution generation

The initial solution in our case is generated through a hybrid operation of the modified savings method of Clarke and Wright[14] and some improvement heuristic modules developed in Salhi and Rand.[15,16] For each problem we generate a set of 20 savings solutions using different values of the route shape parameter γ. Each solution is then further refined by a series of the improvement modules. These modules are:

- *Combine*: builds larger routes by combining pairs of routes.

*Correspondence: NA Wassan, Kent Business School, University of Kent, Canterbury CT2 7PE, UK.
E-mail: N.A.Wassan@kent.ac.uk

- *Shift*: involves transferring a customer from one route to another.
- *Swap*: exchanges two nodes between a pair of routes at a time.
- *Perturb*: extends the idea of *shift* by considering three routes at a time.
- *Idle*: removes empty routes from the system.

The routes are then harmonized by applying well-known *2-opt and 3-opt* procedures.

Neighbourhood search procedure

Any TS implementation needs some kind of method (usually neighbourhood structure) that can be guided or/and manipulated by the TS framework to find solutions that are normally beyond the reach of a hill climbing heuristic. We have used a neighbourhood heuristic that is based on exchanging nodes between a given set of initial vehicle routes. These exchanges are called 1-exchange and 2-exchange schemes.

The *1-exchange* scheme involves shifting a node from one route to another and swapping two nodes between two given routes.

The *2-exchange* is an extension to the 1-exchange processes that involves shifting two consecutive nodes from one route to another and swapping four nodes between two given routes by taking two consecutive nodes from each of the routes.

Reactive TS implementation

TS is a meta-strategy that employs computer memory structures to avoid phenomena like *local minima* and *limit cycles*. From the TS literature, it is clear that the success of a TS algorithm is highly dependent on the settings of its components, specially the *tt* value. If the *tt* is set at a high value, then the search may be restricted to certain regions, and if it is set to a low value then the search may cycle itself. Therefore, a balanced *tt* value is needed to control and run the search process smoothly. Battiti and Tecchiolli[17] developed an approach that dynamically determines the *tt* value during the search process. Their version of TS known as *reactive tabu search* (RTS) employs two mechanisms: the first mechanism builds an automated tabu tenure that is maintained throughout the search process by the dynamic reaction to the repetitions. The second mechanism is called escape diversification strategy that takes the search process out from its current position if it appears to be repeating itself excessively. Battiti and Tecchiolli performed random repositioning of customers for this purpose. In our case, the escape mechanism is modified, instead of random repositioning of customers the diversification is obtained by using *2-exchanges*, while keeping *1-exchanges* for the first mechanism. However, we found that there was no need to change the first mechanism parameter values drastically, hence only small changes are made.

Hashing functions

As mentioned above, the RTS method works mainly by reacting to the repetitions of the solutions. The repetitions are detected using a computer science technique called *Hashing Function Search* (HFS), a pattern described in Juliff.[18] The method generates a code, S_r, for each solution. The S_r is set to the sum (overall routes) of the product of the customer index x_j and the total number of customers in each corresponding route $|R_p|$:

$$S_r = \sum_{p=1}^{v} \sum_{x_j \in R_p} x_j |R_p|$$

For each solution, the S_r value is computed and then divided by a largest prime number smaller than the total number of iterations, *iter*, to be performed. HFS uses a matrix of dimensions, *iter* $\times$ 10, where the first column stores a *key* for each solution and other columns are to store S_r values. More than one solution may have the same key, hence collisions might happen. A small number of collisions occur in our case, which are dealt with using the extra columns of the matrix.

The RTS Algorithm

[Step 1] Initial solution phase

- Choose the best initial solution S_{best} from the generated set.
- Set stopping rule

[Step 2] TS phase

- Initialize data structures for the tabu list and hashing arrays
- Initialize RTS parameters as follows:
  - $tt = 1$
  - Repetition = 0
  - Chaos = 9
  - Increase = 1.1
  - Decrease = 0.9
  - Moving Average, MovAvg = 0
  - Number of iterations since last change in *tt* value, LastChange = 0
  - Gap between repetitions, GapRpt = 0
  - Maximum gap allowed, GapMax = 50
  - Iteration number when last time repetition occurred, LasTimeRpt = 0
  - Iteration number of the most recent repetition, CurTimeRpt = 0

[Step 3] Start search

- Perform *1-exchange* neighbourhood moves and complete a cycle of search
- Scan for the best neighbouring solution, S'
- Update route configurations by setting $S = S'$ as a current solution
- Apply *2-opt* and *3-opt* post-optimizer procedures to harmonize the new routes

[Step 4] RTS mechanism-1

- Generate a record (*Sr*) for the solution S through hashing, and store it into *hashing data structures*
- Search for a possible repetition, using *Hash Function Search*
- LastChange = LastChange + 1
- If repetition was found, then goto [Step 5]
- Else add the solution record *Sr* to the *Hash Table*, and goto [Step 6]

[Step 5] Calculate the gap between two repetitions and increase tt

- GapRpt = CurTimeRep − lasTimeRpt
- Repetition = Repetition + 1
- If (Repetition > Chaos) then,
- Repetition = 0, call *mechanism-2* in [Step*]
- If (GapRpt < GapMax) then
- MovAvg = 0.1*GapRep + 0.9*MovAvg
- *tt* = *tt**Increase
- LastChange = 0, and goto [Step 7]

[Step 6] Decrease tt

- If (LastChange > MovAvg) then
- *tt* = *tt**Decrease
- LastChange = 0, goto [Step 7]

[Step 7] Update iteration counter and the best solution,

- iter = iter + 1
- If $C(S) < C(S_{best})$ then set $S_{best} = S$
- Check for stopping rule, if continue, goto [Step 3]

*[Step *]* RTS mechanism-2: run a mini RTS algorithm

- Set the current and best escape solutions to S (i.e., $S_{ec} = S$ and $S_{eb} = S$)
- Set *escape* iterations
- Perform *2-exchange* neighbourhood moves
- Scan for the best solution, S'_{ec}, from neighbouring solutions of S_{ec}
- If $C(S'_{ec}) < C(S_{eb})$, then update $S_{eb} = S'_{ec}$ and $S_{ec} = S'_{ec}$
- Else $S_{ec} = S'_{ec}$
- Repeat until the allowed number of iterations
- Return to the main algorithm by setting its current solution S to S_{eb} (if found) or to S_{ec}

Computational experience

The RTS algorithm is coded in Fortran 77/90, and run on a Sun sparc server 1000 with 50 MHz processor under solaris 2.3 system. The algorithm is tested on different sets of test problems: Christofides *et al*[19] C1–C14 (in short CMT), Fisher[11] F1–F3 and Russell[20] E1–E4. The distances in these data sets are Euclidean distances and therefore real numbers. However, some researchers have rounded the data into integers for C1–C14 and E1–E4. In order to provide consistent results, we have run our algorithm on both real and integer data of these sets.

The RTS algorithm results report

Table 1 shows results of a single run of the standard RTS algorithm for various data sets. Clearly, it has performed equally well on all the data sets. The main achievement of the algorithm can be seen in its consistency in terms of both quality and speed. A significant overall improvement 4.04% is gained over the initial solution, while taking a relatively moderate amount of computer time.

Comparison of the best results

Table 2 shows the comparison of the best results of the TS implementations listed in the introduction, for the 14 CMT test problems with real data. While the solution value of all other implementations is chosen from several runs, we report RTS solution of five runs. Statistical figures of 0.195 ARPD above the best known solution with 0.307 SD shows that our algorithm is quite competitive.

The results in Table 3 are obtained using integer data of the CMT problems. The RTS algorithm seems to be working even better for the integer data. It has produced equal or better solutions for most of the problems including the five new best known, with significantly lower statistical figures of 0.048 ARPD and 0.105 SD.

The results in Table 4 further illustrate our algorithm's consistency in producing high-quality solution. It has produced new best known solution for all four test problems at a low computer time using real as well as integer data.

Finally, we compare our algorithm for Fisher's test problems in Table 5. The RTS algorithm appears to be competitive in terms of both the solution quality and speed. Note that the results presented in Tables 3–5 are from only two runs.

Table 1 RTS single run solution for various data sets with real distances data

Problem	*Size*	*Q*	*ST/TL*	*Initial solution*	*RTS*	*Relative % improvement on initial solution*	*CPU Time (in min)*
C1	50	160	0/∞	549.47	**524.61**	4.52	0.35
C2	75	140	0/∞	872.07	**835.26**	4.22	3.25
C3	100	200	0/∞	848.63	829.44	2.26	3.46
C4	150	200	0/∞	1086.34	1038.1	4.44	10.13
C5	199	200	0/∞	1398.92	1308.53	6.46	33.06
C6	50	160	10/200	561.26	**555.43**	1.03	0.46
C7	75	140	10/160	924.27	**909.68**	1.57	5.56
C8	100	200	10/230	926.86	**865.92**	6.57	3.23
C9	150	200	10/200	1214.45	1164.08	4.08	3.78
C10	199	200	10/200	1515.02	1408.65	7.02	7.63
C11	120	200	0/∞	1103.27	1044.76	5.3	2.41
C12	100	200	0/∞	827.32	**819.56**	0.93	0.43
C13	120	200	50/720	1576.51	1542.86	2.13	0.76
C14	100	200	90/1040	882.85	**866.37**	1.86	1.73
F1	44	2010	0/∞	723.54	**723.41**	0.01	0.01
F2	71	3000	0/∞	260.16	**241.97**	8.05	0.81
F3	134	2210	0/∞	1433.71	1166.95	18.6	18.35
E1	75	100	0/∞	1042.96	**1026.65**	1.56	1.01
E2	75	180	0/∞	777.18	**740.65**	4.7	0.7
E3	75	220	0/∞	703.72	**692.71**	1.56	0.3
E4	100	112	0/∞	1120.3	**1097.93**	2.06	3.85
NMP	32	38000		2006.34	**2006.34**	0	0.01
Average						4.04	4.55

ST/TL: service time/total length; Q: Vehicle capacity; NMP: 32 node problem proposed in Noon *et al.*[10]
Bold figures indicate the best values.

Table 2 Comparison of the best known results for the CMT problems with real distances data

Problem	*Osman*[5]	*Gendreau et al*[6]	*Rego and Roucairol*[8]	*Xu and Kelly*[9]	*RTS*	*Taillard*[7]*/Rochat and Taillard*[12]
C1	**524.61**	**524.61**	**524.61**	**524.61**	**524.61**	**524.61**
C2	838.62	**835.26**	**835.32**	**835.26**	**835.26**	**835.26**
C3	829.18	**826.14**	827.53	**826.14**	827.5	**826.14**
C4	1044.35	1031.07	1044.35	1029.56	1034.78	**1028.42**
C5	1334.55	1311.35	1334.55	1298.58	1304.37	**1291.45**[RT]
C6	**555.43**	**555.43**	**555.43**	—	**555.43**	**555.43**
C7	**909.68**	**909.68**	**909.68**	—	**909.68**	**909.68**
C8	866.75	**865.95**	866.75	—	**865.94**	**865.94**
C9	1164.52	1162.89	1164.12	—	1164.08	**1162.55**
C10	1417.85	1404.75	1420.84	—	1404.87	**1395.85**[RT]
C11	**1042.11**	**1042.11**	**1042.11**	—	1042.77	**1042.11**
C12	**819.59**	**819.56**	**819.56**	**819.56**	**819.56**	**819.56**
C13	1545.98	1545.93	1550.17	1042.11	1542.77	**1541.14**
C14	**866.37**	**866.37**	**866.37**	—	**866.37**	**866.37**
ARPD	0.529	0.198	0.547	—	0.195	
SD	0.94	0.413	0.961	—	0.307	

ARPD: Average of relative percentage deviation; SD: standard deviation; RT: solution values obtained in Rochat and Taillard.[12]
Bold figures indicate the best values.

Performance analysis of the TS algorithms

Laporte *et al*[4] and Cordeau *et al*[3] describe four major attribute criteria against which good heuristics, especially those designed for the VRP can be measured. These are: *accuracy*, *speed*, *simplicity* and *flexibility*. We think *consistency* should also be included in the above criteria. Unfortunately, not all of the algorithms described in the Introduction are able to handle single-handedly all the benchmark problems while keeping all these attributes. For instance, Osman's[5] algorithm lacks accuracy as compared to Taillard[7] and the Rochat and Taillard[12] algorithms, which

produce together all of the best known solutions for the real data in Table 2; however, the latter two cannot compete at the speed level. The Gendreau *et al*[6] algorithm, however, appears to meet most of the above criteria, but falls somewhat behind on accuracy. The results of Rego and Roucairol[8] algorithm seems faster than the rest but do not compete on accuracy. The Xu and Kelly[9] algorithm finds similar quality results to that of Taillard[7] for the CMT problems but only for those without tour length constraints. The RTS algorithm on the other hand shows a very good performance, covering all of the above criteria by achieving a good accuracy with reasonable speed and remarkable consistency. It is also simple to implement because it uses very basic neighbourhood structures and is flexible in the sense that the model has been extended to some VRP variants, see Osman and Wassan,[22] Wassan and Osman.[13]

Table 3 Comparison of the best known results for CMT problems with integer distances data

Problem	*Gendreau et al*[21]	*Xu and Kelly*[9]	*Time*	*RTS*	*Time*	*Best known*
C1	**521**	**521**	3.5	**521**	1.7	**521**
C2	832	**830**	34.35	831	3.23	**830**
C3	**815**	822	35.84	818	20.1	**815**
C4	1024	1024	49.87	**1015**	50.93	**1015**
C5	1316	1296	228.77	**1289**	95.76	**1289**
C6	**548**	—	—	**548**	1.08	**548**
C7	**907**	—	—	**907**	2.61	**907**
C8	**856**	—	—	**856**	4.63	**856**
C9	1180	—	—	**1147**	21.51	**1147**
C10	1404	—	—	**1392**	9.06	**1392**
C11	1035	**1034**	2.35	**1034**	22.53	**1034**
C12	824	**820**	104.71	**820**	4.43	**820**
C13	**1529**	—	—	1532	7.55	**1529**
C14	866	—	—	**865**	0.55	**865**
ARPD	0.547	—	—	0.048		
SD	0.858	—	—	0.105		

ARPD: average of relative percentage deviation; SD: standard deviation.
Bold figures indicate the best values.

Conclusion

Inspired by the fact that exact methods can only solve small size problems, Laporte[23] rightly pointed out TS as one of the most promising avenues of research, and exhorted researchers to explore it. These insightful remarks have attracted tremendous response from the OR/MS research community. As a result, several good works have been reported including those described in the Introduction. These investigations have provided undoubtedly a better understanding of the power of TS that will be useful in its use to other related problems. To explore the power of TS further, the current work is another attempt to provide some valuable information on the importance of the correct *balance* of intensification and diversification within TS to obtain accuracy with consistency in solution results for various types of benchmark problems. It also reveals how different neighbourhood heuristics can be manipulated through an efficient use of TS components. Since the success of the implementation is entirely attributed to the above factors while using very simple neighbourhood structures, the algorithm can be

Table 4 Comparison of the best known solution for Russell[20] test problems

Problem	*Noon et al*[10]	*Rego and Roucairol*[8]	*Xu and Kelly*[9]	*Time*	*RTS*	*Time*	*Best known*
Real data							
E1	—	1041.84	—	—	**1026.65**	1.1	**1026.65**
E2	746	746.48	—	—	**740.65**	0.71	**740.65**
E3	695.94	694.64	—	—	**690.2**	0.75	**690.2**
E4	—	1103.88	—	—	**1092.91**	2.65	**1092.91**
Integer data							
E1	—	1034	1023	4.15	**1021**	2.73	**1021**
E2	739	739	736	18.59	**735**	21.03	**735**
E3	690	688	685	1.4	**682**	9.88	**682**
E4	—	1093	1080	34.91	**1079**	11.6	**1079**

Bold figures indicate the best values.

Table 5 Comparison of the best known solution for Fisher[11] test problems

Problem	*Fisher*[11]	*Time*	*Rego and Roucairol*[8]	*Time*	*Xu and Kelly*[9]	*Time*	*RTS*	*Time*	*Best known*
F1	**723.54**	49.73	727.74	14	**723.54**	1.41	**723.54**	0.11	**723.54**
F2	**241.97**	105.01	244.54	380	244.54	86.65	**241.97**	0.81	**241.97**
F3	1163.6	253.83	1167.22	768	1176.72	191.45	1165.93	160	**1162.96**

Bold figures indicate the best values.

regarded as distinctive from the others on the grounds that it does not totally rely on the power of a underlying local or neighbourhood search method.

References

1 Bodin L, Golden B, Assad A and Ball M (1983). Routing and scheduling of vehicles and crews: the state of the art. *Comput Opns Res* **10**: 63–211.
2 Laporte G and Osman I H (1995). Routing problems. A bibliography. *Ann Opns Res* **61**: 227–262.
3 Cordeau J-F, Gendreau M, Laporte G, Potvin J-Y and Semet F (2002). A guide to vehicle routing heuristics. *J Opl Res Soc* **53**: 512–522.
4 Laporte G, Gendreau M, Potvin J-Y and Semet F (2000). Classical and modern heuristics for the vehicle routing problem. *Int Trans Opl Res* **7**: 285–300.
5 Osman IH (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Ann Opns Res* **41**: 421–451.
6 Gendreau M, Hertz A and Laporte G (1994). A tabu search heuristic for the vehicle routing problem. *Mngt Sci* **40**: 1276–1290.
7 Taillard ED (1993). Parallel iterative search methods for the vehicle routing problems. *Networks* **23**: 661–676.
8 Rego C and Roucairol C (1996). A parallel tabu search algorithm using ejection chains for the vehicle routing problem. In: Osman IH and Kelly JP (eds). *Metaheuristics: Theory & Applications*. Kluwer, Boston, MA, pp 661–675.
9 Xu J and Kelly JP (1996). A network flow-based tabu search heuristic for the vehicle routing problem. *Transp Sci* **30**: 379–393.
10 Noon CE, Mittenthal J and Pillai R (1994). A TSSP + 1 decomposition strategy for the vehicle routing problem. *Eur J Opl Res* **79**: 524–536.
11 Fisher ML (1994). Optimal solution of vehicle routing problems using minimum *k*-trees. *Opns Res* **42**: 626–646.
12 Rochat Y and Taillard ED (1995). Probabilistic diversification and intensification in local search vehicle routing. *J Heuristics* **1**: 147–167.
13 Wassan NA and Osman IH (2002). Tabu search variants for the mix fleet vehicle routing problem. *J Opl Res Soc* **53**: 768–782.
14 Clarke G and Wright JW (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Opns Res* **12**: 568–581.
15 Salhi S and Rand GK (1987). Improvements to vehicle routing heuristics. *J Opl Res Soc* **38**: 293–295.
16 Salhi S and Rand GK (1993). Incorporating vehicle routing into the vehicle fleet composition problem. *Eur J Opl Res* **66**: 313–330.
17 Battiti R and Tecchiolli G (1994). The reactive tabu search. *ORSA J Comput* **6**: 126–140.
18 Juliff P (1990). *Program Design*, 3rd edn. Prentice-Hall: Australia.
19 Christofides N, Mingozzi A and Toth P (1979). The vehicle routing problem. In: Christofides N, Mingozzi A, Toth P and Sandi C (eds). *Combinatorial Optimization*. Wiley, Chichester, pp 431–448.
20 Russell A (1977). An effective heuristic for the M-tour travelling salesman problem with some side constraints. *Opns Res* **25**: 517–524.
21 Gendreau M, Hertz A and Laporte G (1991). *A tabu search heuristic for the vehicle routing problem*. Working paper, CRT-777, Centre de recherche sur les transports, Université de Montréal.
22 Osman IH and Wassan NA (2002). A reactive tabu search meta-heuristic for the vehicle routing problem with back-hauls. *J Schedul* **5**: 263–285.
23 Laporte G (1992). The vehicle routing problem: an overview of exact and approximate algorithms. *Eur J Opl Res* **59**: 345–358.

Received December 2003;
accepted December 2004 after two revisions