

A Tabu Search Algorithm for the Vehicle Routing Problem with Deliveries and Pickups

Niaz Wassan and Gábor Nagy

Centre for Heuristic Optimisation
Kent Business School
University of Kent

1

Centre for Heuristic Optimisation

... we carry out research in various application areas of Combinatorial Optimisation, with a special focus on locational analysis and vehicle routing, using heuristic search techniques



Prof. Saïd Salhi



Dr. Gábor Nagy



Dr. Niaz Wassan



Dr. Paola Scaparra

2

Vehicle Routing with Pickups and Deliveries

Problem Definition

Two types of customers

Linehauls: expect delivery of goods from a depot

Backhauls: wish to send goods to the depot

Objective: find minimum-cost set of vehicle routes

Constraints: Satisfy all customer demand

Respect maximum capacity of vehicles

We assume all goods go to, or come from, the depot – that is, no goods are transported direct from one customer to another.

3

Applications

Delivery of drink bottles to shop, pickup of empty bottles

Delivery and pickup of mail to/from customers or post offices

Delivery of new household appliances and removal of old ones

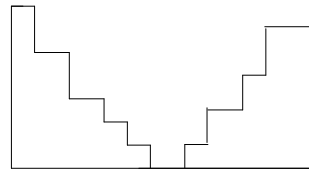
Airport minibus: transporting travellers between home/hotel and airport

4

Problem Versions

1. backhaul (VRPB)

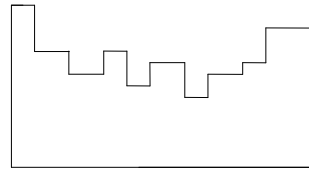
all linehauls must be served before backhauls



backhaul

2. mixed (VRPMDP)

linehauls and backhauls can occur in any order



mixed/simultaneous

3. simultaneous (VRPSDP)

customers can have both demand and supply

4. other

not all pickup and delivery problems are VRPDP

5

Literature Review

Although vehicle routing has a wide subject literature (see *e.g.* Toth and Vigo, book, 2002), vehicle routing with pickups and deliveries attracted relatively little attention, but developed steadily of late...

classical backhaul

Goetschalckx *et al.* (EJOR, 1989)
Toth and Vigo (TranspSci, 1997)
Toth and Vigo (EJOR, 1999)
Osman and Wassan (JSched, 2002)
Wade and Salhi (Omega, 2002)
Wassan (JORS, 2007)

mixed and simultaneous

Dethloff (Spektrum, 2001)
Wade and Salhi (Omega, 2002)
Nagy and Salhi (EJOR, 2005)
Crispim and Brandão (JORS, 2005)
Tang and Galvão (CompOR, 2006)
Chen and Wu (JORS, 2006)

6

Initial Solution Phase

Sweep Method proposed by Gillett and Miller (1974) to generate initial solution

- this approach orders the customers according to their polar coordinates and then partitions the ordering into feasible routes

Modified Sweep

- *customers near the depot are excluded from sweep*
- *they form their own single-customer routes*
- *this gives flexibility to the algorithm later on*

7

Improvement Phase – various improvement routines

- *Shift: moves a customer from one route to another*
- *Swap: swaps two customers that are on different routes*
- *Local-shift: moves a customer to a different place on its route*
- *Reverse: reverses route direction (can be helpful!)*

8

Tabu Search

Proposed by Fred Glover

An extension to neighbourhood search, allowing non-improving moves. To avoid cycling, a “**tabu list**” is used to prevent returning to solutions already visited. The tabu list is updated from time to time with respect to “**Tabu Tenure**” (tt value).

- Fixed tt
- Robust tt
- Reactive tt

Tabu search used by:

Crispim and Brandão (JORS, 2005) – simultaneous/mixed
Tang and Galvão (CompOR, 2006) – simultaneous
Chen and Wu (JORS, 2006) – simultaneous

9

Reactive Tabu Search

Proposed by Battiti and Tecchioli (1994)

A refinement of tabu search where the *length of time* (tt value) record remains in the tabu list is controlled *dynamically*.

Reactive tabu search used by:

Osman and Wassan (JSched, 2002) – backhaul
Wassan (JORS, to appear) – backhaul
Wassan, Nagy and Wassan (in progress) – simultaneous/mixed

There are other ways of enhancing the tabu search framework:

Using both long- and short-term memory (Crispim and Brandão, JORS, 2005)
Combining it with *adaptive memory programming* (Wassan, JORS, to appear)

10

RTS Algorithm for VRPSDP

[Step-1] Initialisation

- Start with the constructed initial solution by the Modified Sweep method as a S_{best} .
- Set stopping rule
- Initialise data structures for the tabu search components
- Initialise RTS parameters

[Step 2] Start search

- Perform *shift and swap* neighbourhood moves and complete a cycle of search
- Scan for the best neighbouring solution, S'
- Apply *local shift* and *reverse* post-optimizer procedures to get any further improvement on the current set of routes
- Update route configurations by setting $S = S'$ as a current solution

11

[Step 3] RTS mechanism

- Check if the current solution is a repetition
- If repetition was found, then increase tt by 10%
- Else if repetition was not found, then decrease tt by 10%

[Step 4] Update the search parameters and the best solution

- $iter = iter + 1$
- If $C(S) < C(S_{best})$ then set $S_{best} = S$
- Check for stopping rule; STOP
- Else go to [Step 2]

12

1. Computational Results – Backhaul Problems

Class 1 dataset – Goetschalckx & Jacobs-Blecha (EJOR, 1989) – 62 instances
25 to 150 customers; proportion of backhauls between 20% and 50%

Class 2 dataset – Toth and Vigo (TranspSci, 1997) – 33 instances
21 to 100 customers; proportion of backhauls is either 1/2, 1/3 or 1/5

reactive tabu adaptive memory programming (Wassan, JORS, to appear)

21 new best results for dataset 1 (real values)
improvement over Toth & Vigo (1996) **1.13%** – optimality gap **0.02%**

15 new best results for dataset 1 (integer values)
improvement over Toth & Vigo (1999) **1.00%**

9 new best results for dataset 2
improvement over Toth & Vigo (1999) **0.77%** – optimality gap **0.76%**

13

Computational Results – Mixed Problems

Class 1 dataset – Goetschalckx *et al* (EJOR, 1989); Halse (PhD, 1992)
25 to 150 customers; proportion of backhauls between 20% and 50%

Class 2 dataset – Toth and Vigo (TranspSci, 1997); Wade (PhD, 2002)
21 to 100 customers; proportion of backhauls is either 1/2, 1/3 or 1/5

Class 3 dataset – Salhi and Nagy (JORS, 1999) – 28 instances
50 to 199 customers; proportion of backhauls is either a 1/10, 1/4 or 1/2
time constraints apply to half the instances

reactive tabu search (Wassan and Nagy, in review)

31 new best for dataset 1 – improved Halse by **0.98%**

15 new best for dataset 2 – improved Salhi *et al* by **0.55%**

30 new best for dataset 3 – improved Crispim and Brandão by **4.37%**

14

Computational Results – Simultaneous Problems

only one dataset: Salhi and Nagy (JORS, 1999); 28 instances; 50 to 199
customers; time constraints apply to half the instances

10 new best for nonconstrained – improved best known by **1.32%**
(Crispim and Brandão, Chen and Wu, Tang and Galvão)

[Average gap from LB]

Nagy&Salhi	Dethloff	Crispim&Brandao	Chen&Wu	Tang&Galvao	RTS
41.28%	38.18%	27.15%	18.03%	19.10%	15.72%

(for constrained cases comparison is problematic...)

15

Suggestions for Future Research

Technical enhancements

Combining reactive tabu search
with adaptive memory programming

Combining reactive tabu with
variable neighbourhood search

Allowing search to traverse
infeasible solutions

Model extensions

Multiple depots

Heterogeneous fleet

Allowing mixture of linehaul and
backhaul goods only subject to
extra free space

Airport minibus: a stochastic-
deterministic problem

16

Table 6.1: Comparison between different algorithms for the CMT problems (without maximum distance constraints).

Problem Size	Nagy & Salhi			Dethloff			Crispim&Brandao			Chen & Wu			Tang & Galvao			RTS-VRPSD			LB	
	Sol	v	T	Sol	v	T	Sol	v	T	Sol	v	T	Sol	v	T	Sol	v	TB		TT
CMT1X 50	525	5	0.3	501	3		477	3	11.2	478.59	3	7.74	472	3	3.7	468.30	3	0.3	48	454.68
CMT1Y 50	525	5	0.3	501	3		485	3	9.1	480.78	3	7.81	470	3	4.37	458.96	3	2	69	455.52
CMT2X 75	841	10	0.8	782	7		710	6	24.3	688.51	6	24.86	695	7	6.91	668.77	6	5	94	617.01
CMT2Y 75	839	10	0.7	782	7		715	6	26.4	679.44	6	12.02	700	7	7.61	663.25	6	6	102	617.64
CMT3X 100	829	8	1.1	847	5		744	5	37.3	744.77	5	94.06	721	5	11.04	729.63	4	65	294	646.73
CMT3Y 100	829	8	1.5	847	5		742	5	33.5	723.88	5	120.66	719	5	12.01	745.46	4	15	285	648.04
CMT12X 100	820	10	1.3	804	6		731	5	37.1	678.46	6	46.83	675	6	12.23	644.70	5	22	242	568.79
CMT12Y 100	825	10	1.4	825	5		860	5	33.7	676.23	6	56.35	689	6	12.80	659.52	6	12	254	573.53
CMT11X 120	1087	7	2.4	959	4		944	4	32.4	858.57	4	321.08	900	4	18.17	861.97	4	10	504	663.38
CMT11Y 120	1075	7	2.6	1070	4		1035	4	29.6	859.77	5	230.72	910	5	18.08	830.39	4	129	325	662.84
CMT4X 150	1053	12	3.8	1050	7		915	7	58.1	887.00	7	501.95	880	7	24.60	876.50	7	69	558	714.18
CMT4Y 150	1047	12	3.3	1050	7		996	7	47.6	852.35	7	406.32	878	7	29.09	870.44	7	73	405	715.67
CMT5X 199	1334	16	7.4	1348	11		1136	10	89.4	1089.22	10	1055.83	1098	11	51.50	1044.51	9	16	483	858.14
CMT5Y 199	1334	16	7.2	1348	11		1129	10	77.1	1084.27	10	771.71	1083	10	56.21	1054.46	9	132	533	856.59
Average gap	41.28%			38.18%			27.15%			18.03%			19.10%			15.72%				

Sol: Solution value; **V:** number of vehicles used in the solution, **LB:** lower bound
T: CPU time in seconds; **TB:** CPU time to best; **TT:** Total CPU time (for 3000 iterations)
 Best solutions are highlighted in **bold**.
 Average gap is with respect to the lower bound (found by Tang and Galvão)

Table 6.2.: Comparison between different algorithms for the CMT problems (with maximum distance constraints).

Problem Size	Dethloff			Nagy & Salhi			RTS (with δ)			Tang & Galvao			RTS-VRPSD (without δ)			LB		
	Sol	v	T	Sol	v	T	Sol	v	TB	TT	Sol	v	T	Sol	v		TB	TT
CMT6X 50	584	6		555	6	0.3	556.06	6	0.30	32	476	3	3.7	471.89	3	3	65	436.63
CMT6Y 50	584	6		555	6	0.3	558.17	6	0.49	33	474	3	4.37	467.70	3	9	60	437.11
CMT7X 75	961	11		910	11	0.7	903.05	11	2	49	695	7	6.91	663.95	6	6	86	607.97
CMT7Y 75	961	11		910	11	0.7	903.36	11	2	47	700	6	7.61	662.50	6	6	80	606.38
CMT8X 100	928	9		873	9	1.8	879.60	9	40	131	720	5	11.04	726.88	5	4	254	649.25
CMT8Y 100	936	9		867	9	1.6	917.42	10	2	113	721	5	12.01	741.96	5	15	315	655.53
CMT14X 100	871	10		879	11	1.5	823.95	10	28	134	675	6	12.23	644.70	5	22	244	558.86
CMT14Y 100	871	10		879	11	1.5	823.34	10	4	122	689	6	12.80	659.52	6	12	255	568.48
CMT13X 120	1576	11		1557	11	2.3	1647.51	12	10	163	918	5	18.17	858.48	5	156	319	—
CMT13Y 120	1576	11		1546	11	2.3	1647.04	11	29	173	910	5	18.08	880.56	4	224	546	—
CMT9X 150	1299	15		1188	14	3.5	1220.00	15	48	201	885	7	24.60	880.61	7	532	561	—
CMT9Y 150	1299	15		1188	14	3.4	1213.11	15	79	171	900	8	29.09	886.84	7	87	562	—
CMT10X 199	1571	19		1420	18	7.5	1464.58	19	157	301	1100	11	51.50	1079.99	10	19	529	—
CMT10Y 199	1571	19		1420	18	7.5	1419.79	18	80	318	1083	11	56.21	1058.09	10	271	524	—

Sol: Solution value; **V:** number of vehicles used in the solution
T: CPU time in seconds; **TB:** CPU time to best; **TT:** Total CPU time
 δ : drop time (used for Dethloff and Nagy & Salhi, but not used for Tang & Galvão). Lower bounds (**LB**) also assume no drop time.
 Best results are highlighted in **bold** only for the case of zero drop time

Table 6.3. Comparison between single runs and best results over 5 runs

Problem Size	Previous best result	RTS-VRPSD (single-run, normal sweep)			RTS (single-run, modified sweep)			RTS-VRPSD (best of 5 runs, modified sweep)					
	Solution v	Solution v	TB	TT	Solution v	TB	TT	Solution v	TB	TT			
CMT1X 50	472 ^T	479.75	3	0.36	80	468.30	3	0.22	48	468.30	3	0.3	48
CMT1Y 50	470 ^T	473.72	3	1	103	465.47	3	2	59	458.96	3	2	69
CMT2X 75	688.51 ^C	668.77	6	49	95	680.34	6	1	82	668.77	6	5	94
CMT2Y 75	679.44 ^C	663.25	6	29	102	675.95	6	1	79	663.25	6	6	102
CMT3X 100	721 ^T	762.96	5	340	343	729.63	4	4	294	729.63	4	65	294
CMT3Y 100	719 ^T	746.56	5	338	341	745.46	4	11	285	745.46	4	15	285
CMT12X 100	675 ^T	655.03	6	69	254	649.14	5	3	220	644.70	5	22	242
CMT12Y 100	676.23 ^C	662.99	6	186	245	668.43	6	33	182	659.52	6	12	254
CMT11X 120	858.57 ^C	934.43	4	727	730	861.97	4	69	504	861.97	4	10	504
CMT11Y 120	859.77 ^C	870.93	4	719	722	869.59	4	14	583	830.39	4	129	325
CMT4X 150	880 ^T	876.50	7	502	560	883.87	7	95	474	876.50	7	69	558
CMT4Y 150	852.35 ^C	893.34	7	304	595	870.44	7	25	448	870.44	7	73	405
CMT5X 199	1089.22 ^C	1071.90	10	596	681	1061.69	9	62	487	1044.51	9	16	483
CMT5Y 199	1084.27 ^C	1063.11	11	637	686	1054.46	9	40	533	1054.46	9	132	533
CMT6X 50	584 ^D	558.81	6	7	44	556.06	6	0.30	32	556.06	6	0.30	32
CMT6Y 50	584 ^D	558.81	6	9	39	558.17	6	0.49	33	558.17	6	0.49	33
CMT7X 75	961 ^D	903.51	11	2	45	906.57	11	31	42	903.05	11	2	49
CMT7Y 75	961 ^D	903.36	11	2	53	905.52	11	12	52	903.36	11	2	47
CMT8X 100	928 ^D	913.16	10	80	119	889.10	9	39	139	879.60	9	40	131
CMT8Y 100	936 ^D	919.60	10	50	119	925.08	10	123	129	917.42	10	2	113
CMT14X 100	871 ^D	847.74	11	17	111	831.77	10	2	111	823.95	10	28	134
CMT14Y 100	871 ^D	841.32	10	9	109	823.34	10	4	122	823.34	10	4	122
CMT13X 120	1576 ^D	1675.24	11	54	158	1647.51	12	10	163	1647.51	12	10	163
CMT13Y 120	1576 ^D	1655.16	11	43	163	1647.04	11	29	173	1647.04	11	29	173
CMT9X 150	1299 ^D	1272.46	16	88	170	1220.00	15	48	201	1220.00	15	48	201
CMT9Y 150	1299 ^D	1213.11	15	79	171	1250.86	16	11	185	1213.11	15	79	171
CMT10X 199	1571 ^D	1517.77	20	93	269	1466.33	19	157	301	1464.58	19	157	301
CMT10Y 199	1571 ^D	1427.00	18	184	286	1419.79	18	80	318	1419.79	18	80	318

Previous best result is from Chen & Wu (C), Tang & Galvao (T) or Dethloff (D).

V: number of vehicles used in the solution
TB: CPU time (in seconds) to best; **TT:** Total CPU time
 Single-run solutions that are better than previously known results are highlighted in **bold**. (This is not done for the instances with maximum distance constraints, as these are always better than the results of Dethloff.)