Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 165 (2005) 657–667
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

Discrete Optimization

A heuristic algorithm for the truckload and less-than-truckload problem

Ching-Wu Chu *

Department of Shipping and Transportation Management, National Taiwan Ocean University, 2, Pei Ning Road, Keelung, Taiwan

Received 11 December 2002; accepted 22 August 2003
Available online 25 March 2004

Abstract

The delivery of goods from a warehouse to local customers is an important and practical problem of a logistics manager. In reality, we are facing the fluctuation of demand. When the total demand is greater than the whole capacity of owned trucks, the logistics managers may consider using an outsider carrier.

Logistics managers can make a selection between a truckload (a private truck) and a less-than-truckload carrier (an outsider carrier). Selecting the right mode to transport a shipment may bring significant cost savings to the company.

In this paper, we address the problem of routing a fixed number of trucks with limited capacity from a central warehouse to customers with known demand. The objective of this paper is developing a heuristic algorithm to route the private trucks and to make a selection of less-than-truckload carriers by minimizing a total cost function. Both the mathematical model and the heuristic algorithm are developed. Finally, some computational results and suggestions for future research are presented.

Keywords: Vehicle routing; Heuristics; 0–1 integer programming; Logistics

1. Introduction

The delivery of goods from a warehouse to local customers is an important and practical problem of a logistics manager. In many sectors of the economy, transportation costs amount for a fifth or even a quarter (lumber, wood, petroleum, stone, clay, and glass products) of the average sales dollars [1].

Logistics managers can make a selection between a truckload (a private truck) and a less-than-truckload carrier (an outsider carrier). A private truck allows a company to consolidate several shipments, going to different destinations, in a single truck. A less-than-truckload carrier usually assumes the responsibility for routing each shipment from origin to destination. The freight charged by a less-than-truckload carrier is usually much higher than the cost of a private truck. Selecting the right mode to transport a shipment may yield significant cost savings to the company.

Our motivation for this study stems from observations on a local logistics company. This

* Tel.: +886-2-24622192x3407; fax: +886-2-24631903.
E-mail address: cwchu@mail.ntou.edu.tw (C.-W. Chu).

0377-2217/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2003.08.067

company owns different types of trucks and main business of this company is delivering foods and beverages to wholesalers. Since the business hours of the wholesalers are fixed, the delivery time window constraint is not major concern for this company. But, this company is facing the fluctuation of demand within a year. When the demands are greater than the whole capacity of owned trucks during the peak season, there are two ways to deal with this situation. One is asking truck drivers to work overtime; the other is using the outsider carriers. Since the overtime cost is much higher than that of using an outsider carrier, the logistics managers may consider using an outsider carrier.

In this paper, we address the problem of routing a fixed number of trucks with limited capacity from a central warehouse to customers with known demand. The objective of this paper is developing a heuristic algorithm to route the private trucks and to make a selection of less-than-truckload carriers by minimizing a total cost function.

The literature on vehicle routing problem has been concerned almost exclusively with heuristics. Several families of heuristics have been proposed for the VRP. These can be broadly classified into two main classes: classical heuristics developed mostly between 1960 and 1990, and meta-heuristics whose growth has occurred in the last decade [2]. In general, the classical heuristics are of four types: (1) tour building heuristics, (ii) tour improvement heuristics, (iii) two-phase method, (iv) incomplete optimization methods.

The most often mentioned tour building heuristics is the Clarke and Wright method [3]. There have been many modifications to the basic Clarke and Wright method. Gaskell [4] and Yellow [5] independently introduced the concept of a modified savings given by $S_{ij} - \theta C_{ij}$ where is θ a scalar parameter. One can change emphasis on the cost of travel between two nodes by varying θ.

The tour improvement heuristics are based on Lin [6] and Lin–Kernighan [7] heuristics for the traveling salesman problem. Christofides and Eilon [8] have modified this heuristic for vehicle routing problem. Two-phase methods include those of Gillett and Miller [9] and Christofides et al. [10]. The example of a heuristic based on incomplete optimization is the tree-search method reported in [10].

The meta-heuristics, presented below, is restricted to tabu search methods since these have been proved the most successful meta-heuristic approach. Over the past decade, tabu search have been applied to the VRP by several authors. Osman [11], Taillard [12], Gendreau et al. [13], Rochat and Taillard [14], Xu and Kelly [15] and Rego and Roucairol [16] all obtained quite satisfactory results.

Very little research has examined the problem of selecting between a less-than-truckload and truckload carrier. Ball et al. [17] consider a fleet planning problem for long-haul deliveries with fixed delivery locations and an option to use an outside carrier. Agarwal [18] considers the static problem with a fixed fleet size and an option to use an outside carrier. Klincewicz et al. [19] develop a methodology to address the fleet size planning and to route limited trucks from a central warehouse to customers with random daily demands.

In general, our research described here differs from previous fleet planning or vehicle routing in that it modifies the Clarke and Wright method by shifting from distance to cost and also incorporates the fixed cost of different types of trucks into the model; it allows the permutations of the three improvement procedures that will result in best results; it simultaneously considers the determination of routing a heterogeneous fleet vehicles and the selection of less-than-truckload carriers; it also presents a mathematical model for solving the problem.

This paper is organized as follows. Next section formulates the mathematical model for our problem. Section 3 presents the heuristic algorithm. Some computational results are reported in Section 4. Finally some concluding remarks and suggestions for future research are provided in Section 5.

2. Mathematical model

To simplify the analysis, we formulate our mathematical model based on the following assumptions:

1. We consider one warehouse system; all trucks start at the warehouse and return back to the warehouse.
2. The requirements of all the customers are known; the requirement of each customer cannot exceed the truck capacity;
3. Each customer is served by one truck (either by the private truck or the less-than-truckload carrier); the requirements of all the customers must be met.
4. We restrict ourselves to delivery only.
5. The cost of operating the truck fleet consists of fixed cost and variable cost. Principal cost items in fixed cost include personnel, insurance, and truck depreciation. The main item of variable cost is fuel. It is usually proportional to the distance of truck traveled.

In the following we present an integer programming model and relevant notations:

$$\min\ z = \sum_{k}^{m} \mathrm{FC}_k + \sum_{i}^{n}\sum_{j}^{n}\sum_{k}^{m} C_{ijk}X_{ijk} + \sum_{i}^{n} \mathrm{CL}_i L_i$$

subject to

$$\sum_{k}^{m} Y_{0k} = m \quad (k = 1,\ldots,m), \tag{1}$$

$$\sum_{k}^{m} Y_{ik} + L_i = 1 \quad (i = 1,\ldots,n), \tag{2}$$

$$\sum_{i}^{n} q_i Y_{ik} \leqslant Q_k \quad (i = 1,\ldots,n;\ k = 1,\ldots,m), \tag{3}$$

$$\sum_{j}^{n} X_{ijk} = Y_{ik} \quad (i = 1,\ldots,n;\ k = 1,\ldots,m), \tag{4}$$

$$\sum_{j}^{n} X_{jik} = Y_{ik} \quad (i = 1,\ldots,n;\ k = 1,\ldots,m), \tag{5}$$

$$\sum_{ij\in s} X_{ijk} \leqslant |S| - 1 \text{ for all } S \subseteq \{2,\ldots,n\}\ (k = 1,\ldots,m), \tag{6}$$

$$X_{ijk} \in \{0,1\};\quad Y_{ik} \in \{0,1\};\quad L_i \in \{0,1\}$$
$$(i = 0,\ldots,n;\ j = 0,\ldots,n;\ k = 1,\ldots,m),$$

i: $\{i = 0,\ldots,n\}$, the index set of customers (let the index 0 denote the warehouse);
j: $\{j = 0,\ldots,n\}$, the index set of customers;
k: $\{k = 1,\ldots,m\}$, the index set of trucks;
n: the number of customers;
m: the number of trucks;

$$X_{ijk} = \begin{cases} 1 & \text{if truck } k \text{ travels from customer } i \text{ to customer } j, \\ 0 & \text{otherwise,} \end{cases}$$

$$L_i = \begin{cases} 1 & \text{if the demand of customer } i \text{ is satisfied by the less-than-truckload carrier,} \\ 0 & \text{otherwise,} \end{cases}$$

$$Y_{ik} = \begin{cases} 1 & \text{if the demand of customer } i \text{ is satisfied by the private vehicle } k, \\ 0 & \text{otherwise,} \end{cases}$$

FC_k: fixed cost of private truck k;
C_{ijk}: the cost of truck k traveling from customer i to customer j;
CL_i: the cost charged by the less-than-truckload carrier for serving customer i;
q_i: the demand of customer i;
Q_i: the capacity of private truck i.

The objective is to route the private trucks and to make a selection of less-than-truckload carriers by minimizing a total cost function.

Constraint (1) ensure that all trucks have been assigned to customers.
Constraint (2) ensure that each customer is served either by the private truck or the less-than-truckload carrier.
Constraint (3) are the truck capacity constraints.
Constraints (4) and (5) ensure that a truck arrives at a customer and also leaves that location.
Constraint (6) serve as subtour-breaking constraints.

3. Heuristic algorithm

In this section we describe an algorithm, called TL–LTL, for solving the vehicle routing and the selection of less-than-truckload carriers problem. The heuristic algorithm can be decomposed into three main steps. In the following we describe algorithm TL–LTL by examining its main steps separately.

3.1. Selection step

The first step of algorithm TL–LTL requires the selection of a group of customers, who will be served by the less-than-truckload carriers. In this step, we will check if the total demand is greater than the whole capacity of owned trucks. If the answer is not, we will skip this step and implement next step directly.

In order to minimize the total cost, we have to design a procedure that can achieve this goal. In reality, the freight charged by the less-than-truckload carrier is usually higher than the cost handled by a private truck. It is obvious that we should order the customers in ascending order based on the freight charged by the less-than-truckload carrier and choose the customers with the lowest cost.

The detail for selecting the customers is described as follows:

(1) Calculate the total demand for all customers.
(2) Calculate the whole capacity of owned trucks.
(3) If the total demand for all customers is greater than the whole capacity of owned trucks, go to step (4) otherwise skip this procedure.
(4) Subtract the whole capacity of own trucks from the total demand for all customers, which is the unsatisfied truck capacity.
(5) Order the customers in ascending order based on the freight charged by the less-than-truckload carrier. Starting at top of the list, do the following.
(6) Sum up the demand of each customer until the total demand is greater than the unsatisfied truck capacity. The corresponding customers will be served by the less-than-truckload carrier; the remaining customers in the list will be served by private trucks and will be used for constructing initial solution.

3.2. Initial solution construction

The Clarke and Wright's savings algorithm is used to solve this problem by making two modifications. The first modification to the algorithm is a shift in criterion from distance to cost. The second modification of the Clarke and Wright formulation is a change in the savings calculation.

The mathematical relationship of the savings of linking two customers is a function of the mix of a less-than-truckload carrier and a private truck that serve customers. There are three possible mixes serving a pair of customers: (1) two less-than-truckload carriers; (2) a private truck and a less-than-truckload carrier; (3) two private trucks.

Before explaining the revised savings calculation, we list the relevant notations as follows:

S_{ij} = savings from consolidating shipments to customer i and j into the same truck.
LTL_i = the total cost charged by the less-than-truckload carrier for serving customer i.
TL_{ij} = the total cost of a private truck that travels from warehouse to customer i, then from customer i to customer j and finally returns back to warehouse.
$\mathrm{FC}(Z)$ = the fixed cost of the smallest truck that can serve a demand of Z.
d_{ij} = the distance from customer i to customer j.
v = the cost of traveling a mile for private truck (\$/per mile).

Fig. 1 illustrates the revised savings calculation from linking two customers under each of the three possible mixes.

The detail for constructing the initial solution is described as follows:

(1) Calculate the savings for all pairs customers based on revised savings scenario 1 in Fig. 1.
(2) Order the savings in descending order. Starting at top of the list, do the following.
(3) Find the feasible link in the list which can be used to extend one of the two ends of the currently constructed route.

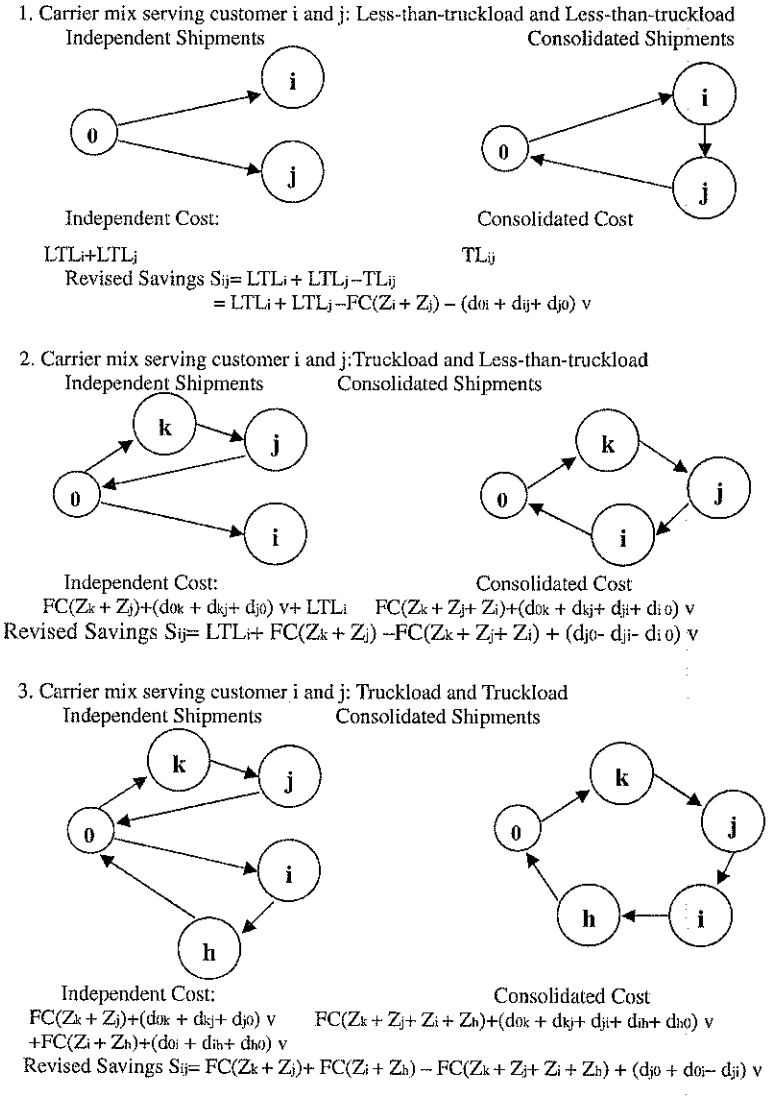


Fig. 1. Savings calculation from consolidating two customers.

(4) If the route cannot be expanded further, terminate the route. Choose the first feasible link in the list to start a new route.
(5) Repeat Steps (3) and (4) until no more links can be chosen.
(6) Output all the temporary single-customer routes (served by the less-than-truckload carriers) and multi-customer routes.
(7) Calculate the savings for single-customer routes based on revised savings scenario 2 in Fig. 1.
(8) Order the savings in descending order. Starting at top of the list, do the following.
(9) Find the feasible link in the current multi-customer routes which can be used to extend the route.
(10) If the route cannot be expanded further, terminate the route.
(11) Repeat Steps (9) and (10) until no more links can be chosen.
(12) Output all the routes.

3.3. Refining procedure

A refining procedure is applied to the solution obtained through the initial solution step. This procedure is composed of a succession of intra-route and inter-route arc exchanges.

3.3.1. Intra-route improvement

Each route is improved by applying a refining procedure which considers all the feasible exchanges of two arcs belong to the route (the so-called intra-route two-exchanges [20]). The procedure is similar to those described in Christofides and Eilon [8] and Kindervater and Savelsbergh [21]. Given a route, a two-exchange is obtained by replacing arcs (m,n) and (p,q) with arcs (m,p) and (n,q), as illustrated in Fig. 2.

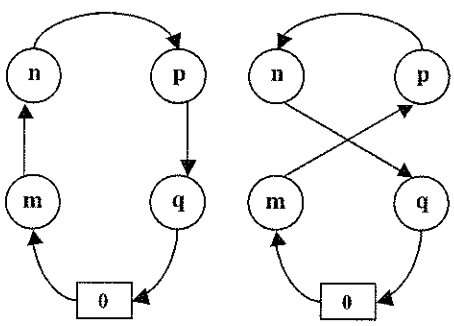


Fig. 2. Example of intra-route two-exchanges.

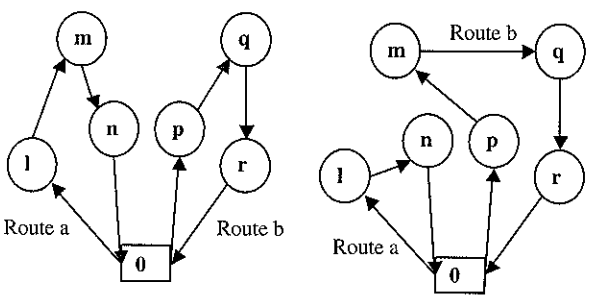


Fig. 3. Example of inter-route one-exchange.

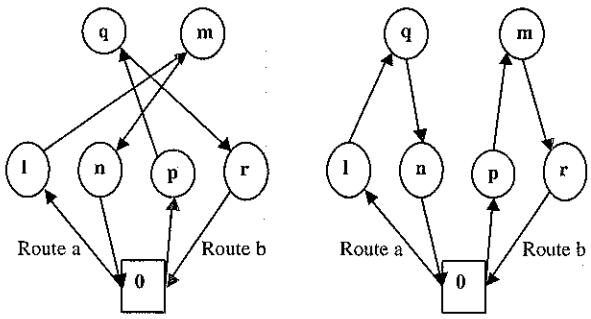


Fig. 4. Example of inter-route two-exchanges.

3.3.2. Inter-route improvement

In this step, a set of routes is obtained by using further local search procedures. These procedures are based on the so-called inter-route one-exchange and two-exchanges, illustrated in Figs. 3 and 4, respectively.

For each node m, belonging to route a, the one-exchange corresponding to its insertion after node p, belonging to route b, is obtained by removing arcs (l,m), (m,n) and (p,q), and replacing them with arcs (l,n), (p,m) and (m,q), as illustrated in Fig. 3.

For each node m, belonging to route a, the two-exchange corresponding to its exchange with node q, belonging to route b, is obtained by removing arcs (l,m), (m,n), (p,q) and (q,r), and replacing them with arcs (l,q), (q,n), (p,m) and (m,r), as illustrated in Fig. 4.

3.3.3. Search procedure

A search procedure is designed in searching for a better solution. From the results of extensive experiments which are not shown here, we know that the implementation sequence of intra-route and inter-route improvement procedure might have impacts on the quality of solution.

The improvement procedures mentioned above include intra-route two-exchanges, inter-route one-exchange and two-exchanges. The possible permutations of three different improvement procedures are only six, so a loop procedure consisting of arranging the possible sequences of intra-route and inter-route improvement is applied on the solution obtained in the initial solution construction phase. The purpose of this loop proce-

Table 1
Results for five test problems

Test problem		Routes	Total cost	CPU[a] time	% larger than the best solution
1	Heuristic algorithm	1-3-5-4-1 1-6-1 Customer 2 is served by LTL	387.5	3.14	0
	Mathematical model	1-3-5-4-1 1-6-1 Customer 2 is served by LTL	387.5	1.1	
2	Heuristic algorithm	1-9-8-7-5-11-1 1-2-4-3-10-1 Customer 6 is served by LTL	631	4.58	7.67
	Mathematical model	1-4-3-10-11-5-1 1-2-9-8-7-1 Customer 6 is served by LTL	586	34.5	
3	Heuristic algorithm	1-3-2-4-11-10-1 1-12-15-8-13-1 1-16-6-14-9-7-1 Customer 5 is served by LTL	900	5.88	0
	Mathematical model	1-8-12-4-2-3-1 1-7-13-10-11-1 1-9-15-14-16-6-1 Customer 5 is served by LTL	900	578	
4	Heuristic algorithm	1-17-16-4-3-2-7-14-10-6-5-9-1 1-22-20-19-21-15-18-23-12-8-13-1 Customer 11 is served by LTL	1681.5	8.42	1.81
	Mathematical model	1-20-23-21-19-15-18-16-17-4-3-2-8-10-13-1 1-7-14-12-6-5-9-22-1 Customer 11 is served by LTL	1651.5	830	
5	Heuristic algorithm	1-19-24-9-15-30-28-27-29-1 1-11-12-13-10-18-8-14-17-16-1 1-20-21-23-7-26-25-2-6-5-4-1 Customers 3 and 22 are served by LTL	1917	11.06	0.86
	Mathematical model	1-3-5-2-1 1-22-18-17-27-29-28-26-25-23-21-1 1-4-6-7-30-16-14-8-10-15-9-13-12-11-1 Customers 19, 20 and 24 are served by LTL	1900.5	2406	

[a] All times are in seconds; the results were obtained on a PC running at 2000 MHz.

dure is in a sense to similar to the tabu search method to escape from a local minimum. Once a better solution is found after finishing improvement phase, the best solution record is updated. We repeat the above improvement processes until all possible permutations of three different improvement procedures have been implemented.

4. Computational results

In this section, we summarize our computational results on five test problems. The detailed data associated with five examples are given in Appendix A. The solutions produced by the heuristic algorithm are compared with the optimal results from the mathematical model. The heuristic algorithm was written in FORTRAN language and the mathematical model was solved using the software LINDO version 6.1. Both of them were implemented on a PC with a 2000 MHz processor. Computational results on five test problems are reported in Table 1.

For the first and the third test problems, our heuristic algorithm obtains the optimal solution. As shown in Table 1, both the mathematical model and the heuristic algorithm yield the same total cost \$387.5 and \$900, respectively. The only difference between two approaches in the third test problem is in that each approach arranges customers in different routes and in different sequences.

Computationally, exact algorithm for the VRP is restricted to solving problems of only up to about 25 customers. For five test problems, the solution time of mathematical model increased quickly with problem size. On the other side, our heuristic algorithm required very little time to solve the problem. Every problem took only a few seconds. The CPU time of test problems is not very sensitive to problem size.

In order to test whether the solution time of our algorithm is not sensitive to larger size of problem, we have solved additional three test problems with the customer size of 51, 76 and 101, respectively. Because the VRP is very difficult to solve with mathematical model even for relatively small size instances, only the average computation times to run the heuristic are reported. These results are presented in Table 2. Though the solution's time increased with problem size, it is obvious that the solution's time increase gradually without rapid growth.

Table 2
Results for larger size of test problems

Test problem[a]	CPU[b] time
1 [E-n51-K5]	27.84
2 [E-n76-K7]	84.48
3 [E-n101-K8]	192.48

[a] Test problem 1 and 3 can be found in Christofides and Eilon [8]; test problem 2 can be found in Gillett and Miller [9].
[b] All times are in seconds; the results were obtained on a PC running at 2000 MHz.

5. Conclusions

The delivery of goods from a warehouse to local customers is an important and practical problem of a logistics manager. In this paper, we develop both the mathematical model and the heuristic algorithm for solving the less-than-truckload and truckload problem. Some computational results are presented. Our heuristic algorithm obtains the optimal or near-optimal solutions in an efficient way in terms of time and accuracy.

As for further research, a wide range of test problems should be performed. It would be interesting to see if other intelligent optimization techniques, such as tabu search, genetic algorithms, simulated annealing and neural networks, can be modified to solve this problem and even provide better results.

Acknowledgements

The author wishes to thank a referee for his valuable comments. This work was partially supported by the National Science Council of Taiwan under grant NSC 89-2416-H-019-004.

Appendix A

Details of problem 1

No.	x	y	Q	LTL
1	35	35	0	0
2	41	49	10	90
3	35	17	7	108
4	55	45	13	132
5	55	20	19	150
6	15	30	26	120

Warehouse co-ordinates(35,35);
Customer demands (q) in cwt.

Vehicle	Capacity	Fixed Cost
1	40 cwt	60
2	30 cwt	50

The variable cost for private Vehicles is $1.5/per mile

Details of problem 2

No.	x	y	q	LTL
1	30	40	0	0
2	37	52	7	78
3	49	49	30	126
4	52	64	16	192
5	20	26	9	102
6	40	30	21	84
7	21	47	15	66
8	17	63	19	156
9	31	62	23	132
10	52	33	11	138
11	51	21	5	168

Warehouse co-ordinates(30,40);
Customer demands (q) in cwt.

Vehicle	Capacity	Fixed Cost
1	75 cwt	120
2	65 cwt	100

The variable cost for private vehicles is $1.5/per mile

Details of problem 3

No.	x	y	Q	LTL
1	40	40	0	0
2	22	22	18	150
3	36	26	26	84
4	21	45	11	114
5	45	35	30	42
6	55	20	21	150
7	33	34	19	54
8	50	50	15	84
9	55	45	16	90
10	26	59	29	138
11	40	66	26	156
12	55	65	37	174
13	35	51	16	72
14	62	35	12	132
15	62	57	31	162
16	62	24	8	162

Warehouse co-ordinates(40,40);
Customer demands (q) in cwt.

Vehicle	Capacity	Fixed Cost
1	110 cwt	150
2	100 cwt	140
3	90 cwt	130

The variable cost for private vehicles is $1.5/per mile

Details of problem 4

No.	x	y	Q	LTL
1	266	235	0	0
2	295	272	125	282
3	301	258	84	246
4	309	260	60	294
5	217	274	500	372
6	218	278	300	384
7	282	267	175	210
8	242	249	350	162
9	230	262	150	270
10	249	268	1100	222
11	256	267	4100	198
12	265	257	225	132
13	267	242	300	42
14	259	265	250	180
15	315	233	500	294
16	329	252	150	390
17	318	252	100	324
18	329	224	250	378
19	267	213	120	132
20	275	192	600	258
21	303	201	500	300
22	208	217	175	360
23	326	181	75	480

Warehouse co-ordinates (266,235);
Customer demands (q) in cwt

Vehicle	Capacity	Fixed Cost
1	4500 cwt	250
2	4000 cwt	200

The variable cost for private Vehicles is $1.5/per mile

Details of problem 5

No.	x	y	q	LTL
1	162	354	0	0
2	218	382	300	372
3	218	358	3100	336
4	201	370	125	252
5	214	371	100	324
6	224	370	200	384
7	210	382	150	330
8	104	354	150	348
9	126	338	450	234
10	119	340	300	270
11	129	349	100	198
12	126	347	950	216
13	125	346	125	222
14	116	355	150	276
15	126	335	150	240
16	125	355	550	222
17	119	357	150	258
18	115	341	100	288
19	153	351	150	54
20	175	363	400	90
21	180	360	300	114
22	159	331	1500	138
23	188	357	100	156
24	152	349	300	66
25	215	389	500	378
26	212	394	800	384
27	188	393	300	276
28	207	406	100	408
29	184	410	150	360
30	207	392	1000	348

Warehouse co-ordinates (162,354);
Customer demands (q) in cwt

Vehicle	Capacity	Fixed Cost
1	4500 cwt	250
2	4000 cwt	200
3	3500 cwt	180

The variable cost for private Vehicles is $1.5/per mile

References

[1] L.M. Schneider, New era in transportation strategy, Transportation Strategy (1985) 118–126.

[2] G. Laporte, M. Gendreau, J.-Y. Potvin, F. Semet, Classical and modern heuristics for the vehicle routing problem, International Transactions in Operational Research 7 (2000) 285–300.

[3] G. Clarke, J. Wright, Scheduling of vehicles from a central depot to a number of delivery points, Operational Research 12 (1964) 568–581.

[4] T.J. Gaskell, Basis for vehicle fleet scheduling, Operational Research Quarterly 18 (1967) 281.

[5] P. Yellow, A computational modification to the savings method of vehicle scheduling, Operational Research Quarterly 21 (1970) 281.

[6] S. Lin, Computer solution of the traveling salesman problem, Bell System Technology Journal 44 (1965) 2245–2269.

[7] S. Lin, B. Kernighan, An effective heuristic algorithm for the traveling salesman problem, Operational Research 21 (1973) 498–516.

[8] N. Christofides, S. Eilon, An algorithm for the vehicle dispatching problems, Operational Research Quarterly 20 (1969) 309–318.

[9] B. Gillett, L. Miller, A heuristic algorithm for the vehicle dispatch problem, Operational Research 22 (1974) 340–349.

[10] N. Christofides, A. Mingozzi, P. Toth, The Vehicle Routing Problem, Combinatorial Optimization, Wiley, Chichester, 1979, pp. 315–338.

[11] I.H. Osman, Metastrategy simulated annealing and tabu search algorithm for the vehicle routing problem, Annals of Operations Research 41 (1993) 421–451.

[12] É.D. Taillard, Parallel iterative search methods for vehicle routing problem, Networks 23 (1993) 661–673.

[13] M. Gendreau, A. Hertz, G. Laporte, A tabu search heuristic for the vehicle routing problem, Management Science 40 (1994) 1276–1290.

[14] Y. Rochat, É.D. Taillard, Probabilistic diversification and intensification in local search for vehicle routing, Journal of Heuristics 1 (1995) 147–167.

[15] J. Xu, J.P. Kelly, A network flow-based tabu search heuristic for the vehicle routing problem, Transportation Science 30 (1996) 379–393.

[16] C. Rego, C. Roucairol, A parallel tabu search algorithm using ejection chains for vehicle routing problem, in: I.H. Osman, J.P. Kelly (Eds.), Meta-Heuristics: Theory and Applications, Kluwer, Boston, 1996.

[17] M.O. Ball, A. Golden, A. Assad, L.D. Bodin, Planning for truck fleet size in the presence of a common-carrier option, Decision Sciences 14 (1983) 103–120.

[18] Y.K. Agarwal, Vehicle routing with limited fleet and common carrier option, presented at TIMS/ORSA Joint National Meeting, Boston, 1985.

[19] J.G. Klincewicz, H. Luss, M.G. Pilcher, Fleet size planning when outside carrier service are available, Transportation Science 24 (1990) 169–182.

[20] P. Toth, D. Vigo, A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls, European Journal of Operational Research 113 (1999) 528–543.

[21] G.A.P. Kindervater, M.W.P. Savelsbergh, Vehicle routing: Handling edge exchanges, in: E.H. Aarts, J.K. Lenstra (Eds.), Local Search in Combinatorial Optimization, Wiley, Chichester, 1997.

第三章　模式建構

3.1 最佳解－整數規劃模式

考慮單一場站，不同性質的車隊，每個顧客的需求已知。假設每個顧客需求皆小於車輛容量，每位顧客僅能被一輛車服務，每輛車皆由場站出發再回到場站，在不違反車輛容量限制及時窗限制的情況下，求以最少之車輛數及最小路線距離，在路線最大的時限內服務所有顧客。

其中，時窗限制可分為「硬性時窗」及「軟性時窗」兩種。硬性時窗意指不可違反時窗限制，即必須在顧客要求之時間上下界之內開始服務該顧客，否則不為可行解，但是允許車輛可在時窗下界之前到達該顧客點，惟須等到時窗下界始可進行服務（停等時間），軟性時窗則可以違反時窗限制（上下界），但違反時會給予一懲罰成本。此外每條路線之總時間不可違反路線時限限制。本研究針對硬性時窗車輛途程問題進行求解。

以下是根據 Solomon.（1983）之硬性時窗車輛路線問題之數學模式加以修改，第一目標為求總車輛數最少，第二目標為給定總車輛數求總路線距離最短。

Object 1　min　K　〈1〉求總車輛數 K 最少

Object 2　min　$\sum_{k} FC_k + \sum_{i}\sum_{j}\sum_{k} C_{ijk} X_{ijk}$　〈2〉給定總車輛數 K 求總路線距離最小

Subject to：

顧客 i 由第 k 輛車服務

↑ 第 k 輛車之容量

$$\sum_i q_i y_{ik} \le V_k \quad k = 1, \ldots, K \qquad \langle 3 \rangle \text{ 容量限制}$$

$$\sum_k y_{ik} = \begin{cases} K & i = 0 \\ 1 & i = 1, \ldots, n \end{cases}$$ 從 depot 出發的車為 K 輛 〈4〉 每點恰經過一次

每位顧客僅由一輛車拜訪

流入 $\sum_i X_{ijk} = y_{jk}$ $j = 0, \ldots, n \quad k = 1, \ldots, K$ 〈5〉 流量守恆

流出 $\sum_j X_{ijk} = y_{ik}$ $i = 0, \ldots, n \quad k = 1, \ldots, K$ 〈6〉 流量守恆

$$b_j \ge b_i + s_i + t_{ij} - (1 - X_{ijk})T \quad i, j = 1, \ldots, n \quad k = 1, \ldots, K$$

(1)

$j \to n \to j$

〈7〉 時間順序

$$t^k_{0,p} \ge b_i + s_i + t_{i0} - (1 - X_{i0k})T \quad i = 1, \ldots, n \quad k = 1, \ldots, K$$

回場站之時間

$n \to 0$ 回到

〈8〉 時間順序

$$b_j \ge t^k_{0,s} + t_{0j} - (1 - X_{0jk})T \quad j = 1, \ldots, n \quad k = 1, \ldots, K$$

顧客 j 開始服務時間

$0 - j$

$0 - \text{某地} - j$

直接由 depot 到達

〈9〉 時間順序

時窗上限

時窗下限 $e_i \le b_i \le l_i$ $i = 1, \ldots, n$ 〈10〉 時窗限制

路線上限

路線下限 $e_0 \le t^k_{0,p} \le l_0$ $p = s, f \quad k = 1, \ldots, K$ 〈11〉 路線時限限制

場站時間

$b_i \ge 0$ $i = 0, \ldots, n$ 開始服務顧客 i 的時間 〈12〉 非負限制式

$y_{ik} = (0, 1)$ $i = 1, \ldots, n \quad k = 1, \ldots, K$ 〈13〉 二元變數

$X_{ijk} = (0, 1)$ $i, j = 0, \ldots, n \quad k = 1, \ldots, K$ 〈14〉 二元變數

其中各變數定義如下：

FC_k：車輛 k 之固定成本

K ：代表所有車輛之集合，下標為 k

q_i ：顧客 i 之需求量

V_k：車輛 k 之容量

c_{ijk}：代表車輛 k 從顧客 i 到 j 之行駛成本

y_{ik}：0-1 整數變數；當車輛 k 服務 i 時其為 1，否則為 0

x_{ijk}：0-1 整數變數；當車輛 k 從顧客 i 到顧客 j 時其為 1，否則為 0

b_i：代表開始服務顧客 i 時間

s_i：代表顧客 i 所需服務時間

t_{ij}：代表顧客 i 到顧客 j 所需的旅行時間

T：為一大的常數，通常以所有路線中最長的路線之服務時間為代表

e_i：代表顧客 i 時窗之下界

l_i：代表顧客 i 時窗之上界

$t^k_{o,s}$： 代表車輛 k 從場站出發之時間

$t^k_{0,p}$：代表車輛 k 抵達場站之時間

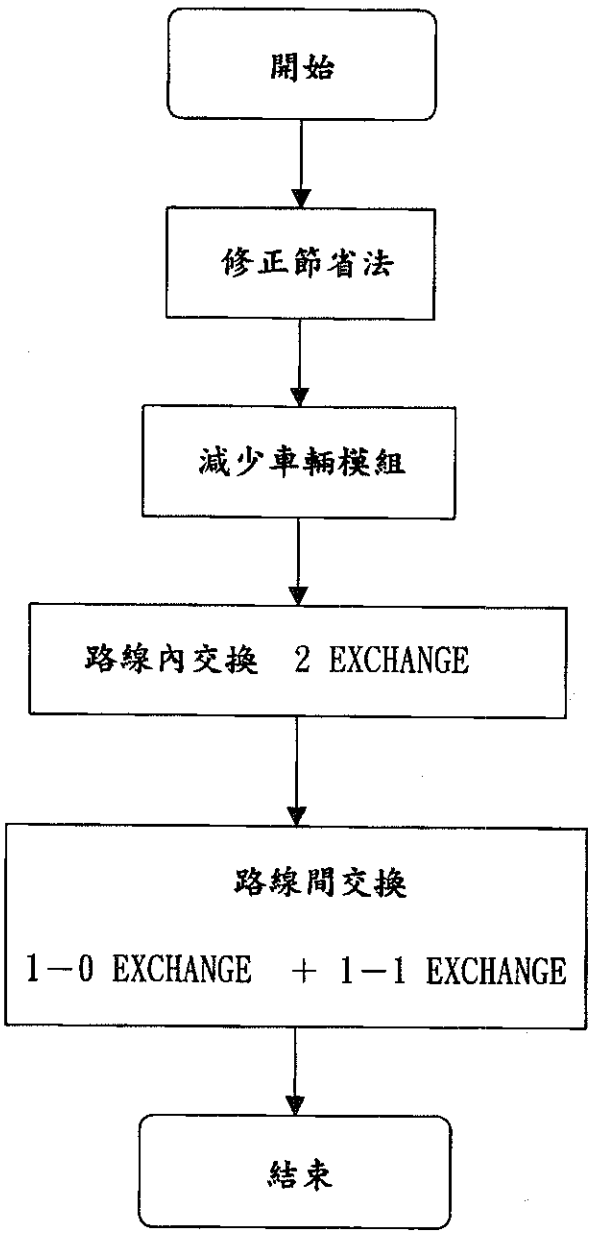
e_0：路線時限之下界

l_0：路線時限之上界

(1)(2) 式為 VRPTW 之目標函數；(3) 式保證每條路線的需求不會超過車輛容量；(4) 式說明每條路線起迄點皆為場站 (depot)，且每個顧客點恰被一輛車服務一次；(5)(6) 式限制每個顧客點恰被一條路線進入與離開一次；(7)－(9) 保證抵達任意兩顧客的時間不會矛盾，且 (7) 式為破除子迴路限制式；(10) 式說明運達顧客點時間不能違反時窗限制；(11) 式說明每條路線不能違反路線時限。

3.2 啟發式解－節省修正法

本研究為解決硬性時窗限制（VRPHTW）之車輛途程問題，其啟發式求解方法架構如圖 3.1 所示



結合 + Tabu Search 未合理想

圖 3.1　VRPHTW 啟發式求解方法架構

3.2.1 起初解構建

在起初解構建，本研究採用節省法。節省法屬於循序構建式啟發式解法。Solomon (1983) 延伸 Clarke 及 Wright (1964) 原運用在 VRP 之節省法，將其運用在 VRPTW 問題上。此法根據成本節省值大小依序將兩顧客點之節線加入路線中，直到所有顧客都分配到路線中為止。開始構建路線時，將第一條節線加入路線後，選擇時間上界較早的顧客點為第一點，以決定路線方向，節省值公式為 $S_{ij} = C_{io} + C_{oj} - QC_{ij}$ 其中 Q 為路線形狀參數（設為 1 或 2），分為 Gaskell (1967) 與 Yellow (1970) 所提出。每一節線加入的步驟皆須檢查是否違反容量及時間可行性。此外，由於時窗使得路線之合併需考慮順序的問題，如圖 3.2 所示：

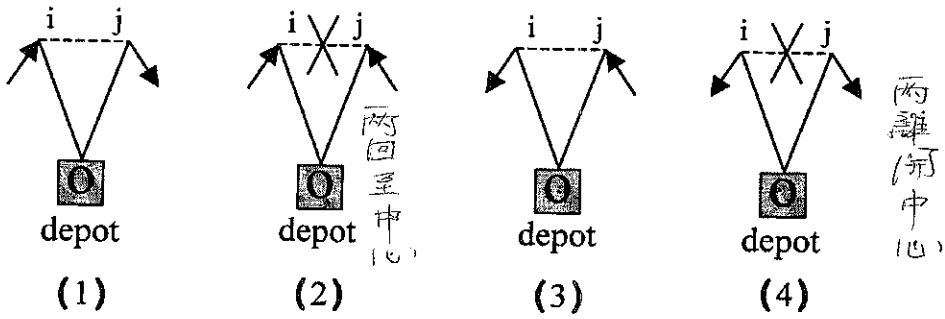


圖 3.2　節省法的解題概念

i、j 分別位於兩條路線的端點，由於路線有方向，若欲將 (i, j) 加入路線中，必須兩條路線的方向不衝突始可加以合併，如上圖之 (1)、(3) 兩種情形可行，(2)、(4) 兩種情形因為路線方向衝突，故不可行。除此之外兩條子路線必須符合下列條件才可合併：

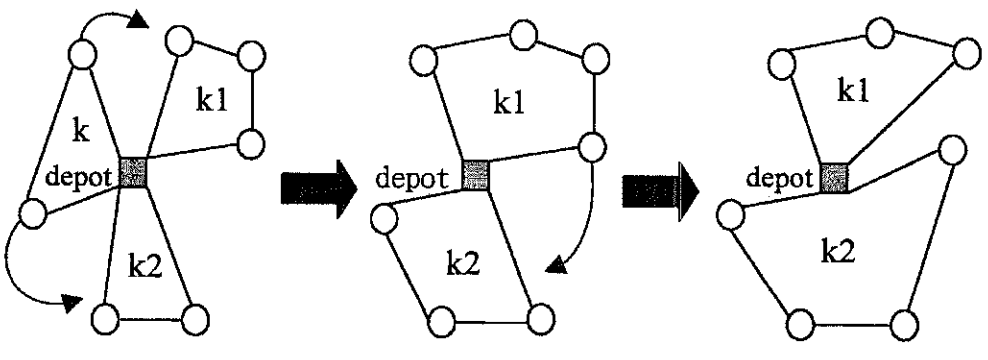
1. i 和 j 在不同子路線中。
2. i 和 j 皆非路線的內部點 (interior point)，即須為路線之端點。
3. 兩子路線合併後的需求量不可超出車輛容量。
4. 兩子路線之方向需可行（如上圖所示），且合併後各顧客點到達時間需符合時窗限制。

5. 加入（i，j）後若造成在 j 點之等待時間（W_j）過長則不採用（i，j），即 $W_j \le W$，W 為事先設定之最大等待時間。

自行設計定

3.2.2 減少車輛模組

減少車輛（Reduction）模組之概念來自 Salhi 及 Rand 之擾動程序（Perturbation procedure, SRP）中的 Reduction 模組，其主要概念為將某條路線上的顧客點全部打散，然後分別插入到其他路線內。但因 VRPTW 問題中時窗特性，將顧客點插入其他路線中有時會違反時窗限制，因此當路線 k 上某點 i 插入 k1 而違反時窗限制時，再將 k1 路線中的另一顧客點 j 由路線 k1 中取出而插入路線 k2 中，使得路線 k1、k2 上各點均滿足時窗限制。如圖 3.3 所示。



explain this figure

圖 3.3 Reduction 的解題概念

將整條路線上之顧客點打散插入其他路線中，必須滿足容量限制及時窗限制，若再限制成本改善值小於零將使可行性降低，故 Reduction 模組之接受法則採用滿足容量限制及時窗限制之可行解即可，若有一個以上之可行解，則鄰解選擇法則以可行解中成本最低者為優先選取對象。Reduction 模組分成兩個階段，第一階段先針對對每一條路線 k，檢查該路線上之所有點是否可以插入其他路線中而符合容量限制及時窗限制，當有一條以上的路線符合此條件（可行

解），本研究選取成本最小的可行解；若無任何路線符合此條件時，進入第二階段，針對某條路線 k，將此路線上在第一階段插入其他路線會違反容量限制或時窗限制的所有點 i，以兩次插入的方式，先插入路線 k1 中，再將 k1 上某點 j 由 k1 中拿出插入路線 k2 中，若可以符合容量限制及時窗限制則執行之，若不能符合限制則檢查下一條路線，若所有路線皆檢查完畢則停止。

本研究僅採取 Reduction 模組之第一階段，即在符合容量及時窗限制前提下，針對每一條途程檢查途程內所有點是否可插入其他途程中，只要滿足限制之可行解即接受，當整個配送順序結構沒有改變時，即停止。本研究採用 Reduction 模組之目的在縮減起始解車數。解的改善交由鄰域改善模組執行。不希望在 Reduction 模組執行時間過長。

3.2.3 鄰域改善模組

本研究所採用的鄰域改善模組可分為：(1) 路線內改善模組，與 (2) 路線間改善模組。

在執行鄰域搜尋，尋找可以交換的鄰近解時，決定選擇哪一個鄰近解進行交換的準則稱為「選擇策略 (selection strategy)」。根據 Osman 的文獻指出，選擇策略一般有兩種：(1) 最佳改善 (best-improve) 策略，即從所有搜尋範圍內的鄰近解中，選擇一個改善最多的解進行交換；及 (2) 首先改善 (first-improve) 策略，則是在搜尋過程中，只要鄰近解有改善就進行交換。此兩種策略並無法證明孰優孰劣，但從執行效率來看，使用「首先改善」選擇策略略優於「最佳改善」選擇策略，因此在後續各種交換改善模組執行架構中，一律採用「首先改善」選擇策略。

此外在實際執行交換法時，依考慮交換範圍之不同可分為「全套」與「半套」之交換法，其間的差異分別如表 3.1 之虛擬碼 (pseudo codes) 所示。由表 3.1 可知，全套交換法的兩個迴圈皆是由第 1 個

第四章 啟發式演算法

針對圖書配送問題所構建之車輛排程問題為混合整數規劃問題(mixed integer programming)，隨著問題大型化(large scale)利用軟體LINDO使用的分支界限(branch and bound)求解方法越顯困難，因其限制式將依決策變數數目增加而呈指數增加，尤其是避免子迴路(subtours)產生的限制式，故大型VRP不易取得正確解，屬於NP-hard之問題。因此，本研究將尋找一啟發式求解方法，以期在可容忍的時間內取得一可行解。

禁忌搜尋演算法(Tabu Search)是一種高階的萬用啟發式方法，(Meta-Heuristic)，專門用來處理組合最佳化的問題。此方法透過彈性記憶體之應用，故常常能跳脫區域最佳解(Local Optimal)，且能在一合理的時間之內求得一近似(或最佳)解。

綜合上述，本研究提出以禁忌搜尋演算法(Tabu Search)來求解圖書配送之車輛排程問題。本章將於4-1、4-2節說明禁忌搜尋演算法概念及本研究演算法之程式設計，4-3節依範例測試演算法之可行性，並於4-4節研提出本章之小結。

4-1 禁忌搜尋法則(Tabu Search)

禁忌搜尋法則為Fred Glover於1977年所提出，目前已經應用的領域[Glover,1990]有排程、通訊、字元辨識、巡迴推銷員問題(TSP)、二次指派問題(QAP)、圖形著色、積體電路設計、時間表設計與類神經網路等組合最佳化之問題，它使用一個非常有彈性的記憶體結構，

因此比起嚴苛的記憶體系統或無記憶體系統，在允許資訊的利用上要來的更具有彈性。

禁忌搜尋法可分為短期與長期二階段。在短期階段，利用禁忌限制式(Tabu Constraints)來限制搜尋的狀態，以避免搜尋的重複與反覆，而免禁準則(Aspiration Level)用來釋放禁忌限制，避免搜尋停滯不前。短期的重點在加速達到區域最佳化；在長期階段則使用加強性(Intensification)與多樣性(Diversification)將搜尋帶入新的區域以求得更佳的解，使得搜尋得以跳脫區域最佳解，進而搜尋總體最佳解。禁忌搜尋法主要由(1)移步；(2)禁忌名單；(3)免禁準則；(4)候選名單；(5)搜尋停止準則，五大模組所組成。以下針對此五大模組加以說明。

(1)移步(Move)

目前禁忌搜尋法最常使用的移步方式為K-OPT法所採用之K邊互換方式(K-edge Exchange)，即刪除原本不相鄰之K條節線，加入新的K條節線。通常K越大則最後所獲得之解品質亦較佳，但因K-OPT法之複雜度為$O(n^k)$，K越大時，其所需耗費的時間也越久。一般而言，K為2或3時最有效率，為一般使用者所採用。

(2)禁忌名單(Tabu List)

用來記錄過去搜尋中每次發生移步(Move)時的屬性，為一個提供禁忌限制的記憶體結構。一般而言，禁忌名單越大則陷入區域最佳解的發生機率將越低，但所需的記憶體空間也將越大，且電腦每次所須的偵測時間將越長，相對之下，可提供移動的空間亦將縮小，這些現象將降低求解的效率。雖然如此，現階段並無一套固定方法來解決禁忌名單尺寸的大小，通常要根據問題本身的特性來決定之。Glover[1990]建議使用魔術數字7作為禁忌名單尺寸。Knox[1994]發

現在 TSP 問題裡，當城市數目小於 100 時，禁忌名單大小與城市數目成線性關係。 Taillard[1991]以一隨機變動的禁忌名單尺寸來求解 QAP 問題，其變動的範圍為 $0.9n \sim 1.1n$ (n 為問題的大小)。

(3)免禁準則(Aspiration Level)

用來釋放一個被列為禁忌之移步，但此禁忌移步若被允許，將可獲得比目前最佳目標函數值還好的目標函數值。

(4)候選名單(Candidate List)

為所有合法移步屬性之集合，即所有可能的移步集合扣除所有禁忌移步集合再加上免禁準則集合所成的集合。

(5)搜尋停止準則(Stopping Criterion)

搜尋停止準則指的是用來終止搜尋進行的條件，較常見的大約有下列四種方式：(1)預設可允許之最大迭代次數；(2)預設目標函數值持續未改善之最大迭代次數；(3)預設 CPU 之最長允許處理時間；(4)預設可接受之目標函數值。一但搜尋達到這些預設值，則停止搜尋，當時最佳解即為最終解。大部份使用禁忌搜尋法的研究者所使用的終止條件為第(1)種，因為使用此種方法保證在一段迭次後會終止搜尋且不因所使用電腦系統不同而有所影響。

禁忌搜尋法之演算法[Glover, 1990,1989]可分為起始與搜尋二階段。起始階段主要目的在輸入禁忌參數、起始解、目標函數係數與搜尋終止條件等項目。而搜尋階段之目的在從一可行的起始解 (x_0) 開始，針對 x_0 之所有鄰近解(Neighborhood solution) $N(x_0)$ 進行評估，然後挑選出非禁忌(或符合免禁準則)之最佳鄰近解進行移步，此移步不一定要對解的品質有所增進，此為禁忌搜尋法與傳統只允許比目前最

佳解還佳的解進行移動之求解方法最大的不同點之一。例如在圖 4-1 之最小化問題中，我們假設起始解為 x_0，最佳鄰域解為 x_{local}，由於在 x_{local} 之所有鄰域解 $N(x_{local})$ 中無法找出比目標函數值 $f(x_{local})$ 還佳的解，但禁忌搜尋法允許比目前最佳解還差的解進行移動，故有可能跳脫區域最佳解 x_{local} 而繼續搜尋的進行，甚至有可能達到整體最佳解 x_{global}。又為預防移步的循環(即 $x_0 \rightarrow x_1, x_1 \rightarrow x_0$)，在每次移步之後必須更新禁忌名單之紀錄，其更新的方法為移去最舊的紀錄，加入新的移步資訊。

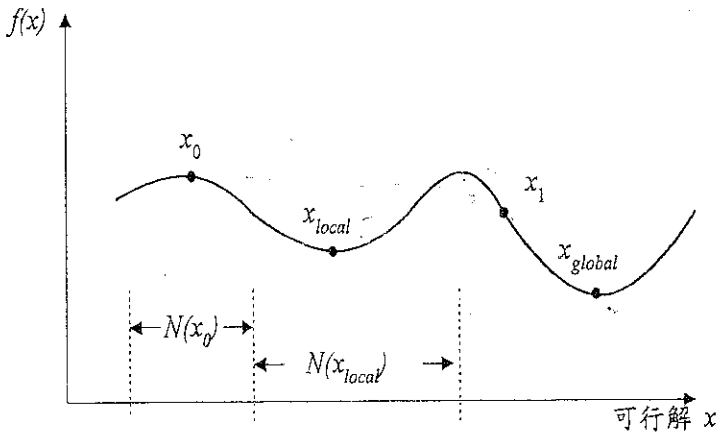


圖 4-1 禁忌搜尋法之基本原理

4-2 VRP 禁忌搜尋演算法

禁忌搜尋法為屬於改善式(Improvement)的啟發式解法，在求解動作開始前，必須先產生一組起始解。本研究所採用的演算法為兩階段法，第一階段為路線建構階段，第二階段為路線改善階段，以下分別就此兩階段演算法進行詳細之說明。4-2-1 節說明路線建構階段之起始解產生法，4-2-2 節則介紹路線改善階段之禁忌搜尋法。

4-2-1 路線建構階段

本研究以節省法(Savings Method)為 VRP 禁忌搜尋法之起始解產生方法，茲針對節省法之演算步驟說明如下：

步驟一：

輸入圖書倉儲中心(depot)與各書店(需求點)的相關資料，如書店數、圖書需求量、退書量、相對距離矩陣、車輛數量及其容量限制。

步驟二：

根據距離資料，求出兩兩節點合併後之成本節省量 $S_{ij} = c_{oi} + c_{oj} - c_{ij}$ 。

步驟三：

將成本節省量 S_{ij} 按大小排序，而後依照 S_{ij} 由大至小排序，根據連結原則與車輛容量限制式滿足與否，逐一構建路線。。

步驟四：

重複步驟三，直到所有 S_{ij} 均已考慮完畢。

步驟五：

將尚未與其他路線連結或合併之書店，直接與圖書倉儲中心連結成單一路線。

步驟六：

分別計算出每條路線之運輸變動成本，並加總算出總運輸成本。

步驟七：

輸出排程路線結果與成本資料。

4-2-2 路線改善階段

路線改善階段是以路線建構階段之起始解為輸入，以禁忌搜尋法為演算法的骨幹，並利用插入法、路線間節點交換法與路線內節點交換法為改善之依據；輸出則為一個較佳之車輛路線。而此禁忌搜尋演算法之運作架構乃參考 Osman[1993]求解典型車輛路線問題所使用的求解架構修改而成。以下介紹本演算法所使用之移步及如何評估一個移步，以作為選取時之依據。

一、移步之方式

移步可表示成一個函數 $M(R, s)$，其中 R 表示目前的路線解，s 表示移步屬性，這個函數的意義為：當路線 R 發生移步 s 時，會將路線 R 轉變為 R′，也就是 $R' = M(R, s)$。

本研究使用插入法或節點交換法來作為路線間、路線內的移步，移步之屬性可分為三種：插入法、路線間節點交換法與路線內節點交換法，此三種移步之屬性可一般化成 $s=(d_i,R_i,d_j,R_j)$，d_i, d_j 代表需求點，R_i, R_j 代表路線解 R 中車輛 i, j 所行駛的路線，為路線編號，其中 d_i 為路線 R_i 內之一需求點或 0，d_j 為路線 R_j 內之一需求點或 0，而 0 代表一個虛擬之需求點，以下介紹這三種移步：

(1) 插入法移步(屬性 s_1)

此移步屬性發生時就是將一個需求點 d_i 由第 i 部車所行駛的路線 R_i 中，重新插入第 j 部車所行駛的路線 R_j，以 $s_1=(d_i,R_i,0,R_j)$ 表示之，其中 d_i 為路線 R_i 內之一需求點。使用此法在路線間之改善方面除了降低路線距離之外，同時可以達到改變車輛的使用數。

(2) 路線間節點交換移步(屬性S_2)

此移步為將第i部車所行駛的路線R_i中的需求點d_i與第j部車所行駛的路線R_j中的需求點d_j相互對調所形成之路線之改變，以$S_2=(d_i,R_i,d_j,R_j)$表示之，d_i與d_j均非虛擬之需求點。

(3) 路線內節點交換移步(屬性S_3)

此移步為將第i部車所行駛的路線R_i中的需求點d_i與需求點d_j相互對調所形成之路線之改變，以$S_3=(d_i,R_i,d_j,R_i)$表示之，d_i與d_j均非虛擬之需求點。

二、移步之評估

移步評估值為移步選擇的依據，而移步評估值是由下列兩個元素組成：節省值與處罰值，我們將移步之評估值表示成$m(s)=x(s)+q(s)$，其中$x(s)$表示移步之節省部份，$q(s)$表示處罰值部份，以下將從這兩方面來探討移步評估值。

(1) 節省值

節省值是指移步所造成之路線總成本的差值，以$x(s)$表示，且$x(s)=\text{Max}\{x(S_1),x(S_2),x(S_3)\}$，$x(S_1)$表示插入法之節省值，$x(S_2)$表示路線間節點交換法之節省值，$x(S_3)$表示路線內節點交換法之節省值。

(a)插入法之節省值，以$x(S_1)$表示，如圖 4-2 所示，此圖說明了一個$S_1=(d_i,R_i,0,R_j)$的移步。其意義為選擇第i部車所行駛的路線R_i上的一個需求點d_i，並選擇第j部車所行駛的路線R_j上的兩個相鄰需求點d_a, d_b間進行插入，(4.1)式為插入法之節省值的計算。

$$x(s_1) = c(d_a, d_b) + c(d_{ip}, d_i) + c(d_i, d_{is}) - c(d_a, d_i) - c(d_i, d_b) - c(d_{ip}, d_{is}) \quad (4.1)$$

$c(d_a, d_i)$ 表示 d_a, d_i 兩點間之直線距離，d_{ip} 表示 d_i 在路線 R_i 上的前一個被服務的需求點，d_{is} 表示 d_i 在路線 R_i 上的後一個被服務的需求點。

(b)路線間節點交換法之節省值，以 $x(s_2)$ 表示，如圖 4-3 所示，此圖說明了一個 $s_2 = (d_i, R_i, d_j, R_j)$ 的移步。其意義為選擇路線 R_i 上的一個需求點 d_i，並選擇路線 R_j 上的一個需求點 d_j，然後進行交換，移步節省值之計算，如(4.2)式。

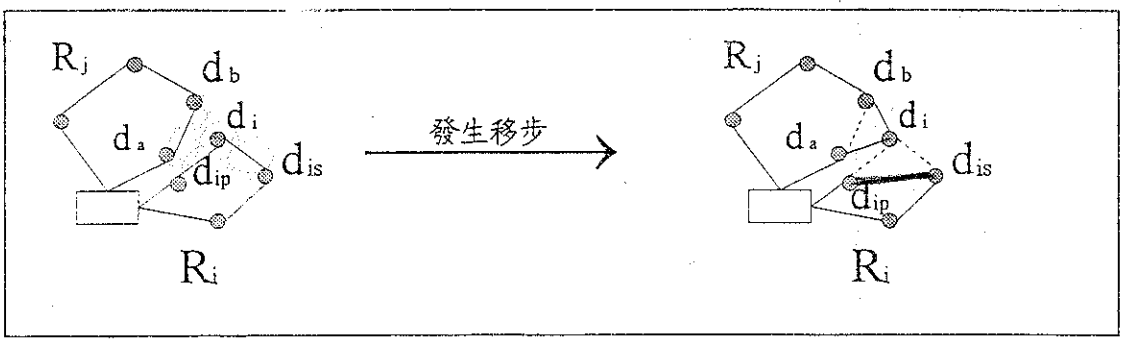
$$x(s_2) = c(g, d_i) + c(d_i, h) + c(e, d_j) + c(d_j, f) - c(g, d_j) - c(d_j, h) - c(e, d_i) - c(d_i, f) \quad (4.2)$$

需求點 g, h 與 d_i 在路線 R_i 之前後順序，與需求點 e, f 與 d_j 之前後順序，如圖 4-3 所示。

(c)路線內節點交換法之節省值，以 $x(s_3)$ 表示，如圖 4-4 所示，此圖說明了一個 $s_3 = (d_i, R_i, d_j, R_i)$ 的移步。其意義為選擇路線 R_i 上的一個需求點 d_i 與另一個需求點 d_j，然後進行交換，移步節省值之計算，如(4.3)式。

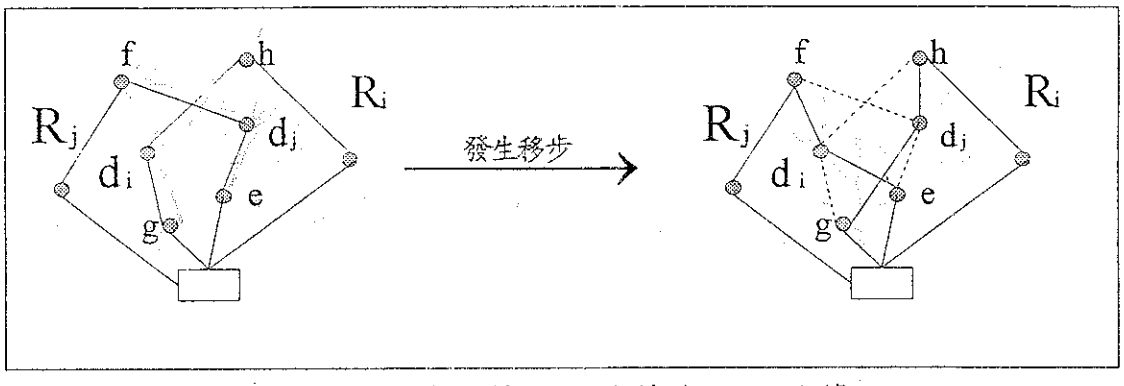
$$x(s_3) = c(g, d_i) + c(d_j, 0) - c(g, d_j) - c(d_i, 0) \quad (4.3)$$

需求點 g, h, d_i 與 d_j 在路線 R_i 之前後順序，如圖 4-4 所示。



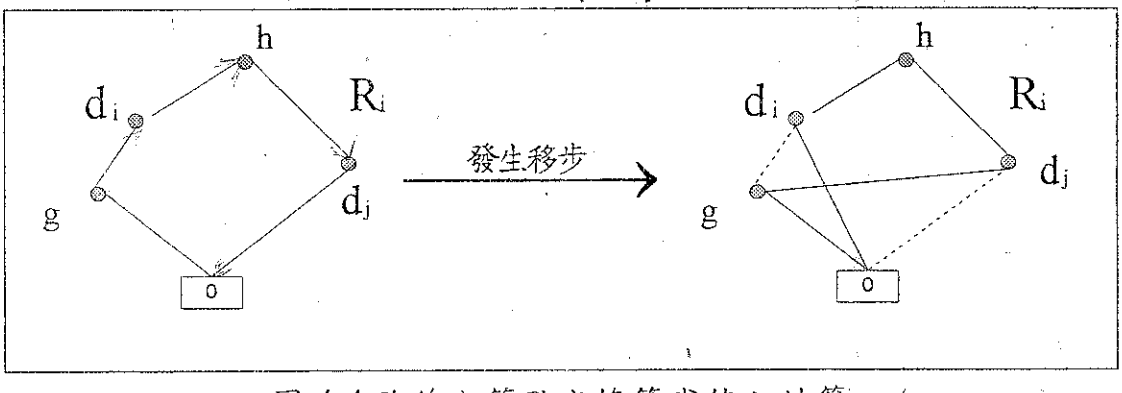
(1-0)

圖 4-2 插入法節省值之計算



(1-1)

圖 4-3 路線間節點交換節省值之計算



(SWAP)

圖 4-4 路線內節點交換節省值之計算

(2) 處罰值

在建構起始解的階段，為了速度的要求，可能違反車輛容量的限制。因此當路線出現違反容量限制時，加入處罰值可適當地表達在此情況下一個解真正之目標函數值。當違反車輛容量的現象發生變化時，移步中處罰值的差值就可用來驅動非可行解，使之逐漸符合車輛容量限制。

我們令 $s=(d_i,R_i,d_j,R_j)$，目前的解為 R，$R' = M(R, s)$，R' 為發

生移步後的解，$R' \in \tilde{R}$，R與R′的差異為R_i變為R'_i、R_j變為R'_j因此，處罰值的變化只會發生在R'_i與R'_j上，(4.4)式表示處罰值的設定。

$$\begin{aligned} q(s) = \alpha\left[\max\left(0, q(R_i) - w(R_i)\right) + \max\left(0, q(R_j) - w(R_j)\right)\right] \\ - \alpha\left[\max\left(0, q(R'_i) - w(R'_i)\right) + \max\left(0, q(R'_j) - w(R'_j)\right)\right] \end{aligned} \quad (4.4)$$

因此當一個移步造成一條路線的總需求違反車輛裝載容量限制發生變化時，$q(s)$的加入才可以代表移步真正的評估值。其中$w(R_i)$表示服務路線R_i之車輛容量，而$q(R_i)$表示路線R_i內之總需求，α為一處罰值係數，以避免所獲得之解出現非可行解現象。

三、移步之記錄

令移步屬性$s = (d_i, R_i, d_j, R_j)$，當此移步發生時，需求點d_i會從R_i重新分派至R_j，需求點d_j會從R_j重新分派至R_i。因此，若希望在未來的一段時間，避免d_i回到R_i、d_j回到R_j，而這段時間也就是禁忌名單大小(Tabu List Size)或是保留期限。

在禁忌名單之資料結構設計上，可以一$n \times n$的二維陣列來儲存相關資訊，n表示書店總數，故共有$n \times n$個儲存格(cell)。每個儲存格的記憶體位置可以$Tabu(d_i, d_j)$表示，意指加入節線(d_i, d_j)為一禁忌的移步，而每個記憶體位置存有一個元素t_size，t_size記錄著節線(d_i, d_j)仍為禁忌的期間長度。

例如當插入法移步發生時，即刪除節線(d_a, d_b),(d_{ip}, d_i),(d_i, d_{is})，加入節線$(d_a, d_i)$$(d_i, d_b)$$(d_{ip}, d_{is})$，記錄為

$Tabu(d_a, d_b) \cdot t\_size = tls$

$Tabu(d_{ip}, d_i) \cdot t\_size = tls$

$Tabu(d_i, d_{is}) \cdot t\_size = tls$

其中 *tls* 為禁忌名單大小，因此在未來移步期間內，加入節線 $(d_a, d_b), (d_{ip}, d_i), (d_i, d_{is})$ 是一個禁忌的移步。

四、免禁準則

(4.5)式將符合免禁準則之禁忌移步以集合的方式來表示，

$$A(R) = \left\{ s \mid s \in T(R),\ z(M(R,s)) < Z_{best},\ R\&M(R,s) \in \tilde{R} \right\} \quad (4.5)$$

$A(R)$ 表示本迭次路線解R符合免禁準則的禁忌移步集合，R為本迭次的路線解，s為移步屬性，$T(R)$ 代表由R轉移至鄰近解之禁忌移步的集合，Z_{best} 為到目前迭次為止所搜尋到的最好的目標函數值，$M(R,s)$ 表示路線解R發生移步 s 後的解，$z(M(R,s))$ 表示路線解R發生移步後解的目標函數值，$\tilde{R}$ 表示所有可行解的集合。

五、候選名單之設計

所有候選名單的集合可以(4.6)式來表示：

$$Cset(R) = S(R) - T(R) + A(R) \quad (4.6)$$

其中R表示本迭次的路線解，$S(R)$ 表示本迭次可由 R 轉移至鄰近解 $N(R)$ 之所有集合，$A(R)$ 為符合禁忌準則的集合。

六、搜尋停止準則

由於希望本研究所提出之演算法在不同的系統所求得的解，不會在解的品質上有所差異，因此採用預設可允許之最大迭代次數為搜尋停止準則並以 *Maxi* 來表示。