Journal of the Operational Research Society (1997) 48, 745–750 0160-5682/97 $12.00

Tabu search for large location–allocation problems

M Ohlemüller
TH-Darmstadt, Germany

Recently it has been demonstrated that the use of simulated annealing is a good alternative for solving the minisum location–allocation problem with rectilinear distances compared with other popular methods. In this study it is shown that the same solution quality and a great saving of computational time can be achieved by using tabu search. It is also possible to transfer this method to location–allocation problems with euclidean distances.

Keywords: location–allocation problem; tabu search; location

Introduction

The location–allocation problem is a special case of the multifacility location problem. A number of m facilities (servers) are to be located and a number of n facilities (customers) are to be allocated to the new facilities such that the total transportation costs are minimized. These costs are measured by the weighted distance between the new and the existing facilities. The new facilities are supposed to be uncapacitated. (The capacitated case is called 'transportation–location problem' and was first formulated and solved by Cooper[1,2]).

For the present the location–allocation problem with rectilinear distances will be considered. Remarks on the transfer to the euclidean case will follow afterwards. The problem can be formulated mathematically as:

$$\min f(x, y, z) = \sum_{i=1}^{m} \sum_{j=1}^{n} z_{ij} b_j d(X_i, A_j) \qquad (1)$$

subject to

$$\sum_{i=1}^{m} z_{ij} = 1, \quad j = 1, \ldots, n, \qquad (2)$$

$$z_{ij} \in \{0, 1\}, \quad i = 1, \ldots, m; j = 1, \ldots, n, \qquad (3)$$

where $f(x, y, z)$ = total transportation costs;
m = number of new facilities;
n = number of existing facilities;
b_j = demand (weight) of existing facility j;
$z_{ij} = 1$, if an existing facility j is allocated to new facility i; 0, otherwise;
$X_i = (x_i, y_i)$, the coordinates (location) of new facility i;
$A_j = (u_j, v_j)$, the coordinates (location) of existing facility j;
$d(X_i, A_j)$ = distance between new facility i and existing facility j.

In the rectilinear case: $d(X_i, A_j) = |x_i - u_j| + |y_i - v_j|$, and in the euclidean case

$$d(X_i, A_j) = \sqrt{(x_i - u_j)^2 + (y_i - v_j)^2}.$$

Because of the tremendous number of possible allocations when both n and m are large, heuristics have been developed. Early methods were provided by Cooper[3] who suggested the Alternate Location–Allocation procedure. Subsequently other approaches have been developed. Kuenne and Soland[4] described a branch and bound algorithm which achieves near-optimal solutions. An exact algorithm was developed by Love and Morris[5] and, Sherali and Shetty[6] created a convergent cutting plane algorithm. Heuristic methods which can also be applied to larger problems were developed and tested by Love and Juel.[7] Even location–allocation problems with l_p-norms can be solved by their algorithms. Recently Liu *et al*[8] described an algorithm for solving location–allocation problems with rectilinear distances by simulated annealing, and concluded that their method can be applied to much larger problems than any other existing method.

The aim of our study is to demonstrate that the use of tabu search is much more efficient for solving location–allocation problems than any existing method, because the accuracy is similar to already known algorithms but the saving of computational time is immense. The rest of the paper is organized as follows.

First the tabu heuristic for solving the rectilinear location–allocation problem is introduced. Then the performance of the program is demonstrated by comparing the results of different popular methods based on known test problems. Finally it is explained how the tabu heuristic can be transferred to location–allocation problems with euclidean distances.

Correspondence: M. Ohlemüller, Siebenburgenstrasse 17, 64295 Darmstadt, Germany.

The tabu-heuristic – program description

As mentioned in the introduction it is possible to divide the program into two main parts. After the Start-routine in which the input of the relevant data is made and an arbitrary allocation between the existing and the new facilities (with unknown locations) is determined, the real process starts with a Location-part. The requirement of this step is a given allocation. Based upon this allocation the original problem 'decomposes' into m single facility location problems (Weber Problems). The coordinates of every new facility can be computed easily by well known methods (see for instance Wesolowsky[9]). In the Allocation-part the coordinates of the new facilities are required. In this step a better allocation of the existing to the new facilities is determined by tabu search. We achieved the best results by using a *tabu–navigation–method* with variable length of the tabu list and diversification. For a comprehensive description of tabu search we refer to Voß[10] or Glover.[11,12] The essential elements that are needed here will be evident from the following.

Description of the parameters

n = number of existing facilities;
m = number of new facilities;
z = allocation matrix; binary matrix with n columns and m rows. In each line there are $m-1$ zero entries and exactly one 1 entry that indicates which new facility the existing facility is allocated to.
l = actual length of the tabu list; raised by 1 if the best neighbour achieves no improvement (see Allocation (2)). If $l > l_{max}$, then $l := l_{min}$.
l_{min} = minimal length of the tabu list;
l_{max} = maximal length of the tabu list;
contr = control- or tabu matrix; relating to the computational time it was profitable to store the tabu 'list' in a matrix. Each position is related to a positive value iff this allocation is tabu (otherwise zero). One element is determined to be tabu if in the Allocation-step an existing allocation is rejected because of a better solution. (This means that the same positions are tabu which receive value 0 instead of 1 in the allocation matrix z.) In each iteration all positive values of the control–matrix are raised by one; if an entry is equivalent to l_{max} the related position receives the value zero and thus is not tabu any longer.
num = 'number of internal iterations': Determines the number of iterations in the Allocation-step. In each of the *num* iterations in allocation a best neighbour is determined (l possible neighbours are tabu).
div = degree of diversification: If a local optimum is reached (the performance of *num* Allocation-steps leads to no improvement on the start-solution of the Allocation-part), in *div* per cent of the columns the actual allocation is replaced by an arbitrary new one.

Besides, at the beginning of the program a variation-factor (number between 1 and 2000) has to be chosen to generate random numbers. Related to this variation-factor a different first allocation—this means a different initial solution is created.

Tabu-algorithm for a location–allocation problem

Start: **Given:** n; coordinates and demand of the existing facilities.
Input: $m, l_{min}, l_{max}, div, num$
contr $(i,j) = 0$ for all i,j.
Obtain an arbitrary first allocation z.
While: stopping criterion is not fulfilled, **do begin**
Location: (1) Compute the optimal coordinates of the new facilities based on the given allocation.
(2) Compute the corresponding objective value.
(3) **Goto** Allocation.
Allocation: **While** number of iterations is less than *num* **do begin:**
(1) Obtain the neighbour with the best objective value (which is not tabu).
(2) **If** the best neighbour achieves no improvement
then→Raise length of tabu list.
(3) Update contr.
(4) Compute the current objective value.
(5) **If** the objective value is less than the current optimum:
then→store value and related allocation.
(6) **IF** a known objective value was computed:
then→diversification
(7) **Goto** (1)
Endwhile
GOTO Location.
Endwhile.

The neighbourhood in the program above is defined in the following way: Two allocations are said to be neighbours if one can be obtained from the other by changing exactly two entries in z (that is one assignment of a facility). In one column the entry with the value 1 receives the value 0, and at another position of the same line one entry changes from 0 to 1. Therefore the number of possible neighbours is given by $n(m-1)$ minus the number of entries which are tabu. A possible stopping criterion can be, for example a prespecified time limit, or a fixed number of Location-steps.

A reallocation of the neighbour with the best objective value (step 1) leads to the largest saving in the objective; this might be negative. It should be mentioned that in Allocation (1) only the deviation from the current objective value is computed; otherwise the complexity of the algorithm would raise from $O(n \cdot m)$ to $O(n^2 \cdot m^2)$.

Performance of tabu search compared to other methods for solving location–allocation problems

First of all, a comparison of the run times and the best-found solutions between the exact algorithm from Love and Morris[5] (LM), Love and Juel's heuristic H5[7] (LJ5), the simulated annealing heuristic from Liu *et al*[8] (SA) and our tabu heuristic (TABU) is demonstrated. The test problems are taken from Love and Juel's paper.[7] The number of existing facilities differs from 20–100 while the number of the new facilities is fixed ($m = 5$). Afterwards a detailed comparison of SA and TABU is given for an extended problem with up to 150 existing facilities. The data for the first comparison is taken from appendix 2 and for the second from appendix 3 of Liu *et al*'s paper.[8]

It is always a problem to compare the run times of the different algorithms because they were measured on different computers: LM and LJ5 obtained their results on a UNIVAC 1110 mainframe computer, whereas SA was running on an IBM PC–AT and TABU was also programmed on a PC (486 DX/2 66) in Fortran 77. Referring to the paper from Liu *et al*[8] it can be maintained, that the speed of a UNIVAC 1110 is faster than that of an IBM PC–AT.

Table 1 shows a comparison of run times and the solutions between TABU and the three other algorithms (test problems 1–6).

The solutions and the run times of SA and TABU are averaged over five randomly generated initial solutions. The results of LM, LJ5 and SA are reported in references 5, 7 and 8. For TABU the following parameters have been chosen: $l_{min} = 5$; $l_{max} = 7$; div = 30%; num = 15.

Analysing the results of Table 1 it is evident that an exact algorithm (like LM) cannot solve a little bit larger problems in justifiable time. Besides, Table 1 shows a great difference if required expenditure of time between TABU and the 'competitors' to find a good solution. Only in test problem 6 SA were we able to find a better solution, but

Table 1 Comparison of solutions and run times (in seconds) between TABU and three other algorithms

Number of test problem	*Problem size*	*SA'94 IBM PC–AT*		*LM'75 UNIVAC 1110*	
		Best objective	*CPU time*	*Objective*	*CPU time*
1	$m=3$				
	$n=16$	191354085	34.85	191354085	362.04
2	$m=2$				
	$n=30$	516254946	45.24	516254946	1995.18
3	$m=3$				
	$n=20$	140236644	65.38	140236644	
4	$m=2$				
	$n=35$	598084656	95.21	598084656	5483.34
5	$m=5$				
	$n=60$	528731912	497.39		
6	$m=2$				
	$n=100$	1575490910	1016.42		
Number of test problem	*Problem size*	*TABU'95 486 DX/2 66*		*LJ5'82 UNIVAC 1110*	
		Best objective	*CPU time*	*Objective*	*CPU time*
1	$m=3$				
	$n=16$	191354085	<0.5	191354085	small
2	$m=2$				
	$n=30$	516254946	<0.5	516254946	small
3	$m=3$				
	$n=20$	140236644	<0.5	140236644	small
4	$m=2$				
	$n=35$	598084656	<0.5	598084656	small
5	$m=5$				
	$n=60$	528731912	21.6	528731912	295
6	$m=2$				
	$n=100$	1583327772	1.3	1583327696	533

this was only less than 0.5% below the value found by LJ5 and TABU. But on the other hand the required run time was much larger than the one of TABU.

In addition to this, Liu *et al*[8] tested further problems with $m = 2, \ldots, 5$ and $n = 20$, 40, 60, 80, 100 and even 150 facilities. (For the data see appendix 3 of Liu *et al*'s paper.[8] A comparison between SA and TABU for these problems are reported in Table 2a (best solution found) and Table 2b (run times).

Note: In TABU the following parameters were chosen:

Problems with $n \leqslant 80 : l_{\min} = 5$, $l_{\max} = 7$, num = 10, div = 30%

Problems with $n = 100 : l_{\min} = 5, l_{\max} = 7$, num = 15, div = 50%

Problems with $n = 150 : l_{\min} = 5, l_{\max} = 7$, num = 20, div = 50%

Table 2a shows that both algorithms achieve good results with their respective run times. No solutions differ more than 1% from the optimal value. Considering the fact that Love and Juel[7] characterized a solution with a difference of less than 0.5% as optimal (deviations of less than 0.5% were quoted as 0.0%) underlines the superior quality of the methods. According to this measure TABU achieved with only one exception always the optimum. A measure of this kind seems to be justified because both methods are heuristics, and if they are applied in practice the exact solution is unknown. Therefore the fact that SA was able to obtain the exact solution value in many test problems is certainly positive, but for a 'real' problem with a unknown solution there will be always a certain uncertainty because even for small problems with $n = 20$ it was in some cases impossible to reach the optimum. Whereas TABU found for every problem with $n \leqslant 80$ facilities the exact solution, and even for the large problems ($n = 100$, $n = 150$) the maximal deviation from the optimum was less than 0.7%.

The evident superiority of TABU does not become obvious until analysing the run times (Table 2b). All test problems up to $n = 80$ facilities were solved (exactly!) with an average run time of less than 20 s. SA required for these problems between 29 and more than 800 s (respective to the size of the problem). For the test problems with $n = 100$ or $n = 150$ facilities SA needed even between 1000 and 3000 s; whereas TABU obtained in at most 60 s comparable results. The saving of required run time can be characterized by a factor of at least 50.

In addition it should be noticed that also TABU achieved with a bit longer run times (and the same parameters as

Table 2a Best solution found

		n					
m		*20*	*40*	*60*	*80*	*100*	*150*
2	Best solution	297190271	670581985	984411533	1286172384	1575490910	2507065582
	% above Best solution found (SA)	0.00	0.00	0.00	0.00	0.00	0.00
	% above Best solution found (TABU)	0.00	0.00	0.00	0.00	0.497	0.3368
3	Best solution	198756665	511181127	732169081	1010412584	1230162263	1989604280
	% above Best solution found (SA)	0.00	0.00	0.00	0.00	0.00	0.00
	% above Best solution found (TABU)	0.00	0.00	0.00	0.00	0.462	0.447
4	Best solution	163355407	411927358	616836446	856084093	1053678953	1709841768
	% above Best solution found (SA)	0.76	0.00	0.00	0.37	0.00	0.55
	% above Best solution found (TABU)	0.00	0.00	0.00	0.00	0.666	0.322
5	Best solution	131993316	359920789	528731912	724230428	919515658	1480630032
	% above Best solution found (SA)	0.00	0.75	0.02	0.00	0.11	0.00
	% above Best solution found (TABU)	0.00	0.00	0.00	0.00	0.04	0.490

Table 2b Run times in seconds

		n					
m		*20*	*40*	*60*	*80*	*100*	*150*
2	(SA)	29.69	146.63	349.53	610.11	1016.24	2664.11
	(TABU)	<0.5	<0.5	2.32	0.96	1.5	3.7
3	(SA)	56.76	190.57	386.65	698.56	1153.83	2868.72
	(TABU)	0.58	2.36	0.98	3.42	2.06	2.6
4	(SA)	55.53	229.39	423.61	799.45	1249.45	2952.89
	(TABU)	2.84	1.04	18.84	13.3	32.44	12.06
5	(SA)	70.84	274.74	497.39	842.43	1320.02	2961.19
	(TABU)	6.06	3.00	15.98	10.44	25.84	61.4

before) the exact solution for some test problems where SA failed to do so. For instance with $n = 100$ and $m = 5$ facilities TABU obtained with an average run time of 73.36 s the optimal solution, whereas SA had a mean deviation of 0.11% with an average run time of 1302.02 s.

Interesting examples of the interdependence of the run times from changing parameters are shown in Figures 1–3 ($n = 80$ and $m = 5$ in all examples).

Figure 1 demonstrates the effect of a changing diversification-factor on the run times. Two conclusions can be drawn: A diversification-factor of 10% is too low to overcome local optima efficiently. On the over hand a factor of more than 60% leads to growing run times on an average, because once a local optimum is reached too much 'good information' about this point is lost.

Consequences of changes in the length of the tabu list are shown in Figure 2a and 2b. Especially Figure 2a is interesting because it emphasizes the positive effects of a variable length of the tabu list.

Concluding Figure 3 demonstrates the effect of the number of internal iterations (that is the number of obtained best neighbours in each Allocation-part) on the run times. The tendency is obvious: Growing numbers of internal iterations are leading to longer run times to find the best objective value. But if the chosen number is too low the run times are raising again because too many Location-steps are required in the algorithm.

Remarks on the euclidean case

For solving location–allocation problems with euclidean distances only a few modifications have to be made. The Allocation-part can be left unchanged, in which for existing coordinates of the new facilities with the use of tabu search a better allocation with a lower solution value is computed. Only the Location-step is a little bit more complicated. Now the well known hyperboloid approximation can be used (see Eyster *et al*,[13] Wesolowsky[9]). Because of this there are two iterative processes in the algorithm and the cost of (run-) time related to the improvement of the current solution for both iterations must be judged very carefully.

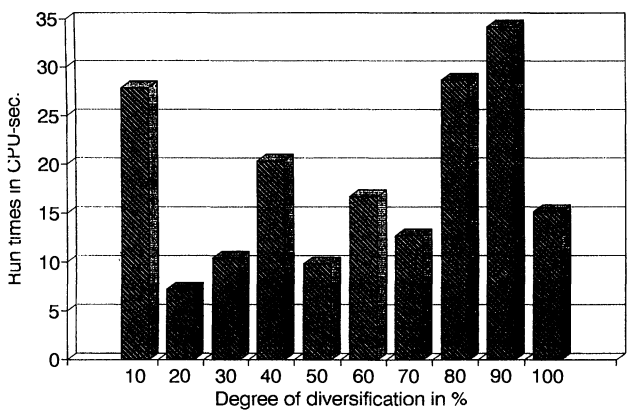

Figure 1 Degree of diversification. Length of tabu list = 5–7; num = 10.

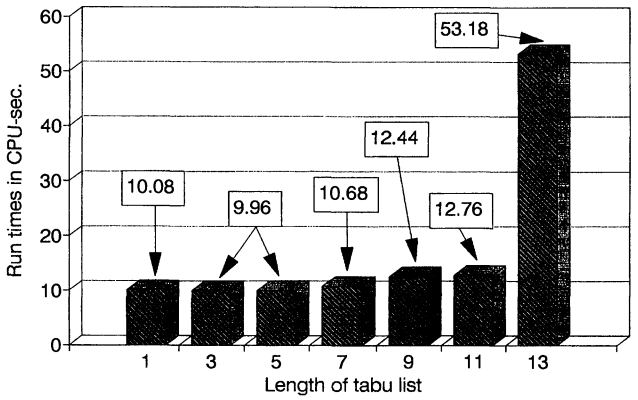

Figure 2a Length of tabu list. div = 30%; num = 10.

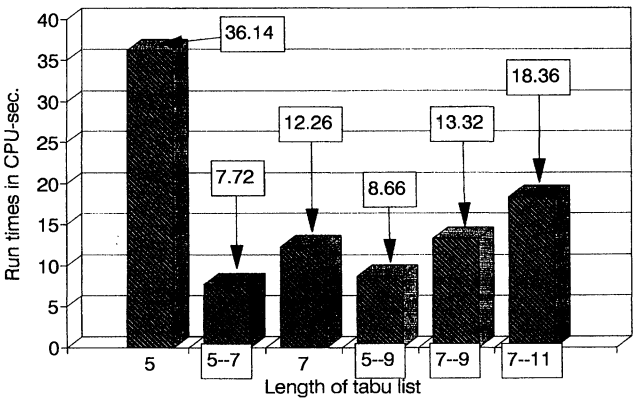

Figure 2b Length of tabu list. div = 20%; num = 10.

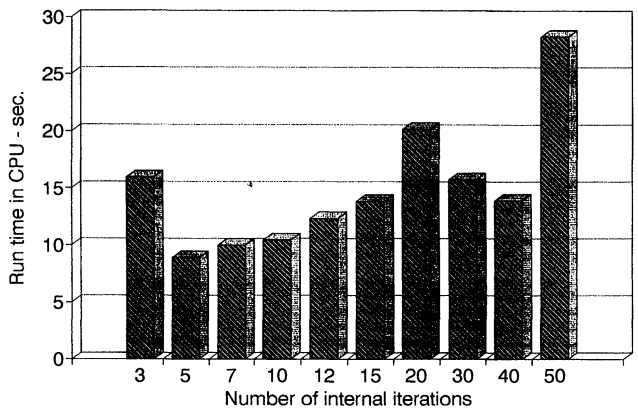

Figure 3 Number of internal iterations. Length of tabu list = 5–7; div = 30%.

Conclusions

Analysing the results shown above, it is evident that the incorporation of the tabu search heuristic for solving location–allocation problems with rectilinear distances is very efficient. The expected deviation from the optimum is at

least as low as of other known methods, whereas the saving of required run times is immense. Therefore this seems to be the best method to find good solutions for large location–allocation problems (especially if $n > 100$) in an acceptable time. The main reason for this saving is obvious: instead of using a very time consuming method for determining a server to each existing facility (in the here analysed uncapacitated case it is always the nearest facility) as done in known algorithms, it is possible to determine with the use of tabu search in a much shorter time an excellent approximation. In addition, it is demonstrated how this draft can easily be transferred to the solution of large location–allocation problems with euclidean distances.

References

1 Cooper L (1963). Location–allocation problems. *Opns Res* **11**: 331–343.
2 Cooper L (1972). The transportation–location problem. *Opns Res* **20**: 94–108.
3 Cooper L (1964). Heuristic methods for location–allocation problems. *SIAM Rev* **6**: 37–53.
4 Kuenne RE and Soland RM (1972). Exact and approximate solutions to the multisource Weber problem. *Math Prog* **3**: 193–209.
5 Love RF and Morris JG (1975). A computation procedure for the exact solution of location–allocation problems with rectangular distances. *Naval Res Logist Q* **22**: 441–453.
6 Sherali AD and Shetty CM (1977). The rectilinear distance location–allocation problem. *AIIE Trans* **9**: 136–143.
7 Love RF and Juel H (1982). Properties and solution methods for large location–allocation problems. *J Opl Res Soc* **33**: 443–452.
8 Liu CM, Kao RL and Wang AH (1994). Solving location–allocation problems with rectilinear distances by simulated annealing. *J Opl Res Soc* **45**: 1304–1315.
9 Wesolowsky GO (1993). The Weber problem. *Location Sci* **1**: 5–23.
10 Voß S (1993). *Intelligent Search*, Informal report, TH Darmstadt, Germany.
11 Glover F (1989). Tabu search—part I. *ORSA J Computing* **1**: 190–206.
12 Glover F (1990). Tabu search—part II. *ORSA J Computing* **2**: 4–32.
13 Eyster JW, White JA and Wierwille WW (1973). On solving multifacility location problems using a hyperboloid approximation procedure. *AIIE Trans* **5**: 1–6.

Received January 1996;
accepted February 1997 after two revisions