ELSEVIER European Journal of Operational Research 89 (1996) 556–563

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Theory and Methodology

A neural network approach to facility layout problems

Kazuhiro Tsuchiya [a,b,c,*], Sunil Bharitkar [c], Yoshiyasu Takefuji [b,c]

[a] *Software and System Lab., Fuji Electric Co. Research and Development, Ltd., 1, Fuji-machi, Hino, Tokyo 191, Japan*
[b] *Faculty of Environmental Information, Keio University, 5322 Endoh, Fujisawa 252, Japan*
[c] *Department of Electrical Engineering and Applied Physics, Case Western Reserve University, Cleveland, OH 44106, USA*

Received 17 January 1994; revised 1 February 1995

Abstract

A near-optimum parallel algorithm for solving facility layout problems is presented in this paper where the problem is NP-complete. The facility layout problem is one of the most fundamental quadratic assignment problems in Operations Research. The goal of the problem is to locate N facilities on an N-square (location) array so as to minimize the total cost. The proposed system is composed of $N \times N$ neurons based on an artificial two-dimensional maximum neural network for an N-facility layout problem. Our algorithm has given improved solutions for several benchmark problems over the best existing algorithms.

Keywords: Facility layout; Quadratic assignment problem; Neural networks; Two-dimensional maximum neuron model; Parallel algorithms

1. Introduction

The facility layout problem is one of the traditional quadratic assignment problems originally presented by Koopmans and Beckmann in 1957 [13]. The problem has been widely studied by many researchers in Operations Research and management science, and known to be NP-complete (NP, nondeterministic polynomial) [18]. Good reviews have been summarized in [14,7] and several benchmark problems have been given for comparing the algorithms [16,20]. Gilmore [6] and Lawler [15] independently developed branch and bound algorithms to find an optimum solution. Several other branch and bound algorithms have been proposed by Pierce and Crowston [17], Burkard [4], and Bazaraa [2]. Another optimum algorithm called cutting plane algorithm was presented by Bazaraa and Sherali [3] and Burkard and Bonninger [5]. The optimum solutions have been confirmed only up to the 15-facility benchmark problem. Because of the prohibitively long computational time required by the optimum algorithms, the near-optimum algorithms have been studied in order to generate a "good" solution in a short time. Some of the earlier near-optim algorithms are H63 [9], HC63-66 [10], CRAFT [11], FLAC [19], FRAT [12], and Biased Sampling (BS) [16]. Relatively new near-optimum algorithms are based on simulated annealing [24,8] or tabu search (TABU) [20]. One of the best simulated annealing methods for the facility layout problem is HSA proposed by Heragu et al. [8],

[*] Corresponding author. Fax: +81-425-86-8016.

which outperformed all the existing schemes in terms of solution quality. HSA must prepare a "good" initial solution and use the simulated annealing to obtain a better solution where several parameters are to be tuned.

The goal of the problem is to locate N facilities on an N-square (location) array so as to minimize the total cost. The mathematical expression of the N-facility layout problem is given by:

Minimize the total cost:

$$\sum_{i=1}^{N} \sum_{j=i+1}^{N} \sum_{p=1}^{N} \sum_{q=1}^{N} c_{i,j} d_{p,q} V_{p,q} V_{i,p} V_{j,q} \tag{1}$$

subject to

$$\sum_{i=1}^{N} V_{i,p} = 1, \quad \text{for } p = 1,\ldots,N, \tag{2}$$

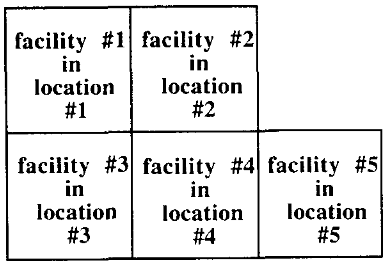
$$\sum_{p=1}^{N} V_{i,p} = 1, \quad \text{for } i = 1,\ldots,N, \tag{3}$$

$$V_{i,p} = \begin{cases} 1 & \text{if facility } \#i \text{ is assigned to location } \#p, \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where $c_{i,j}$ is the cost between facility #i and facility #j, $d_{p,q}$ is the manhattan grid distance between location #p and location #q, $c_{i,j} = c_{j,i}$, and $d_{p,q} = d_{q,p}$. Note that the cost means the flow or the weight between two facilities. Fig. 1(a) and Fig. 1(b) show a solution of the 5-facility layout problem and the cost/distance table respectively where 5 facilities are allocated on a 5-square array. The total cost is 33 $(= c_{1,2}d_{1,2} + c_{1,3}d_{1,3} + c < {}_{1,4}d_{1,4} + c_{1,5}d_{1,5} + c_{2,3}d_{2,3} + c_{2,4}d_{2,4} + c_{2,5} < d_{2,5} + c_{3,4}d_{3,4} + c < {}_{3,5}d_{3,5} + c_{4,5}d_{4,5} = 5 + 2 + 8 + 3 + 6 + 0 + 4 + 0 + 0 + 5)$. Usually each problem has different optimum solutions with the same minimum total cost. Fig. 2 shows one of the optimum solutions for the same problem in Fig. 1 where the total cost is 25.

Hopfield and Tank proposed the first neural network for optimization problems [11]. They used a sigmoid neural network for the traveling salesman problem. Takefuji et al. have proposed a hysteresis McCulloch–Pitts neural network and a one-dimensional maximum (winner-take-all) neural network for NP-complete problems [21–23].

a


facility #1 in location #1
facility #2 in location #2
facility #3 in location #3
facility #4 in location #4
facility #5 in location #5

b

p →

dpq (=dqp)

i ↓	1	2	3	4	5	q ↓
1	--	1	1	2	3	1
2	5	--	2	1	2	2
3	2	3	--	1	2	3
4	4	0	0	--	1	4
5	1	2	0	5	--	5
cij (=cji)	1	2	3	4	5	

j →

Fig. 1. An example of the 5-facility layout problem. (a) A solution of the 5-facility layout problem. (b) The cost (lower left triangle)/distance (upper right triangle) table.

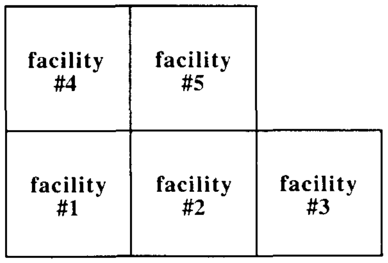


Fig. 2. An optimum solution for the 5-facility layout problem.

In this paper, a two-dimensional maximum neural network is newly introduced for solving the facility layout problem, and the quadratic assignment problem in general.

In the next section, we review the basic mechanism of the artificial neural network for optimization and combinatorics, and explain our neural network representation for the facility layout problem. In Section 3, we describe the neural network parallel algorithm, and then discuss the experimental results in Section 4 where several benchmark problems are used to show the effectiveness of our algorithm. We summarize this paper in Section 5.

2. The neural network representation

The mathematical model of the artificial neural network consists of two components; neurons and weighted synaptic links. An output signal transmitted from a neuron propagates to other neurons as an input signal through the synaptic links. Every artificial neuron has the input U and the output V. The input of the i,j-th neuron $U_{i,j}$ is updated by a motion equation which represents the weighted synaptic links between the i,j-th neuron and other neurons. The output of the i,j-th neuron is given by $V_{i,j} = f(U_{i,j})$ where f is called the neuron's input/output function. Note that $V_{i,j}$ is also used in Eq. 1. The output of the i,j-th neuron $V_{i,j} = 1$ means that facility #i is assigned to location #j.

The input/output function of the two-dimensional maximum neuron is given by:

$$
\begin{array}{lll}
1. & V_{\mathrm{a,b}} = 1 & \text{if } U_{\mathrm{a,b}} = \max\{U_{i,j}\}; \\
2. & V_{\mathrm{c,d}} = 1 & \text{if } U_{\mathrm{c,d}} = \max\{U_{i,j} \mid i \neq \mathrm{a}, j \neq \mathrm{b}\}; \\
3. & V_{\mathrm{e,f}} = 1 & \text{if } U_{\mathrm{e,f}} = \max\{U_{i,j} \mid i \neq \mathrm{a,c}, j \neq \mathrm{b,d}\}; \\
\vdots & \vdots & \vdots \\
N. & V_{\mathrm{g,h}} = 1 & \text{if } U_{\mathrm{g,h}} = \max\{U_{i,j} \mid i \neq \mathrm{a,c,e},\dots, \\
 & & \qquad j \neq \mathrm{b,d,f},\dots\}; \\
 & V_{i,j} = 0 & \text{otherwise, where } i \neq \mathrm{a,c,e},\dots,\mathrm{g}, \\
 & & j \neq \mathrm{b,d,f},\dots,\mathrm{h}.
\end{array}
\tag{5}
$$

For example, facility #a should be assigned to location #b if $U_{\mathrm{a,b}}$ is the largest among all $U_{i,j}$'s, and then facility #c should be assigned to location #d if $U_{\mathrm{c,d}}$ is the largest among all $U_{i,j}$'s except for $i = \mathrm{a}$ or $j = \mathrm{b}$. Note that if two or more than two neurons have the same largest input, one neuron should be selected arbitrarely from among them. The system is composed of $N \times N$ neurons. N neurons always generate nonzero outputs and the other $(N^2 - N)$ neurons generate zero. This two-dimensional maximum neuron model satisfies the constraints of Eqs. (2)–(4).

The motion equation of the i,j-th neuron is generally given by:

$$\frac{\mathrm{d}U_{i,j}}{\mathrm{d}t} = \frac{\partial E(V_{1,1},\dots,V_{i,j},\dots,V_{N,N})}{\partial V_{i,j}}, \tag{6}$$

where E is the computational energy function following an $N \times N$-variable function: $E(V_{1,1},\dots, V_{i,j},\dots, V_{N,N})$. The artificial neural network provides a gradient descent method so as to minimize the fabricated energy function E. Usually the right term in Eq. (6) can be constructed by considering the necessary and sufficient constraints and/or the cost function from the given problem because they give information on interconnections of the artificial neural network. The combination of those constraints and the cost function, however, complicates the motion equation and makes it especially difficult to compute the cost function. The proposed two-dimensional maximum neural network needs only the cost function since all the necessary and sufficient constraints, namely Eqs. (2)–(4), are included in the neuron model. It provides a faster convergence speed and higher convergence rate than those of conventional McCulloch–Pitts neuron or sigmoidal neuron models which require both those constraints and the cost function in the motion equation.

The motion equation of the i,j-th neuron for the facility layout problem is given by:

$$\frac{\mathrm{d}U_{i,j}}{\mathrm{d}t} = Q - R, \tag{7}$$

where Q and R are the objective cost and the real total cost respectively. Q can be set by a user as an expected total cost or even set to be zero.

Our simulation is terminated when R reaches Q. It means that $dU_{i,j}/dt$ is always negative or zero. R is given by Eq. (1) where it is assumed that facility #i is allocated to location #j. For example, to calculate $dU_{2,1}/dt$ at time t in Fig. 1(a), facility #2 in location #2 is allocated to location #1 temporarily as shown in Fig. 3 while facility #1 in location #1 is moved to location #2 temporarily. The real total cost R is 28. If $Q = 20$, then $dU_{2,1}/dt = 20 - 28 = -8$. To calculate $dU_{2,2}/dt$, the assignment of facility #2 to location #2 must be maintained because facility #2 has been actually assigned to location #2 at time t. Since R is 33, $dU_{2,2}/dt = -13$. Eq. (7) describes the degree of penalty which discourages the highly penalized neurons from generating nonzero outputs.

In order to improve the global minimum convergence and to accelerate the simulation speed [23], Eq. (7) was replaced by:

$$\frac{dU_{i,j}}{dt} = \begin{cases} (Q-R)V_{i,j} & \text{if } (t \bmod 10) < \omega, \\ Q-R & \text{otherwise,} \end{cases} \tag{8}$$

where t and ω are the number of iteration steps and a constant parameter respectively. In the first equation, $dU_{i,j}/dt = 0$ if $V_{i,j} = 0$. In other words, the first equation is activated and imposes the penalty on the i,j-th neuron only when it generates a nonzero output. It gives zero-generating neurons chances to generate nonzero outputs and allows the system to escape from local minima. The ω is only one parameter and helps the state of the system to avoid stacking at the local minima. The convergence theorem with proof for the two-dimensional maximum neural network is given in Appendix A.

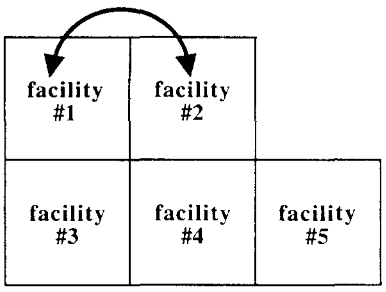


Fig. 3. How to calculate $dU_{2,1}/dt$.

3. Parallel algorithm

A simulator based on the proposed neural network was developed and a synchronous parallel system was simulated. The first-order Euler method was used to solve N^2 equations in Eq. (8) numerically. The following steps describe the proposed algorithm for an N-facility layout problem. Note that t_limit is the maximum number of iteration steps for the system termination condition.

Step 0. Set $t = 0$, and set t_limit, Q, and ω for the N-facility problem.

Step 1. Initialize values of $U_{i,j}(t)$ for $i, j = 1, \dots, N$ using uniform random numbers.

Table 1
Summary of the total costs

N-facility problem	H63 [9]	HC63-66 [10]	CRAFT [1]	BS [16]	FLAC [19]	FRAT [12]	TABU [20]	HSA [8]	This work
5 [16]	25	29	25	25	25	28	–	25	**25**
6 [16]	43	43	43	43	43	45	–	43	**43**
7 [16]	77	74	74	74	74	80	–	74	**74**
8 [16]	109	107	107	107	107	111	–	107	**107**
12 [16]	300	304	289	289	289	302	–	289	**289**
15 [16]	617	578	583	575	585	602	575	575	**575**
20 [16]	1384	1319	1324	1304	1303	1335	1285	1285	**1285**
30 [16]	3244	3161	3148	3093	3079	3160	3062	3062	**3062**
42 [20]	–	–	–	–	–	–	7932	7927	**7926**
49 [20]	–	–	–	–	–	–	11768	11739	**11732**

–: not available.

Step 2. Evaluate $V_{i,j}(t)$ for $i, j = 1,\ldots,N$, using Eq. 5.

Step 3. Compute Eq. (8) for the $N \times N$ neural network for $i, j = 1,\ldots,N$ to obtain $\Delta U_{i,j}(t)$:

$$\Delta U_{i,j}(t) = \frac{\mathrm{d}U_{i,j}}{\mathrm{d}t}. \tag{9}$$

Step 4. $\Delta U_{i,j}(t) \geq 0$, then generate the solution and terminate this procedure.

Step 5. Update $U_{i,j}(t+1)$ for $i, j = 1,\ldots,N$, based on the first-order Euler method:

$$U_{i,j}(t+1) = U_{i,j}(t) + \Delta U_{i,j}(t). \tag{10}$$

Step 6. Evaluate $V_{i,j}(t+1)$ for $i, j = 1,\ldots,N$, using Eq. (5).

Step 7. If $t = \text{t\_limit}$, then terminate this procedure; else increment t by 1 and go to Step 3.

Steps 3 through 5 can run in parallel. The proposed algorithm was tested on a Sun Sparc10 and an HP 9000/710 computer although the algorithm is executable either on a sequential machine or a parallel one.

4. Experimental results

We have examined ten benchmark problems to test our algorithm. Table 1 summarizes the solution quality of the proposed algorithm and those of the best existing algorithms. The proposed algorithm generated better solutions in every benchmark problem. Our simulator discovered improved solutions in the 42-facility and the 49-facility benchmark problems. Fig. 4 shows the configuration of the 42-facility problem with the total cost of 7926 while the best known total cost was 7927 [8]. Fig. 5(a) and Fig. 5(b) show different configurations of the 49-facility problem with the total cost of 11732 while the best known total cost was 11739 [8]. The system converged to these solutions within 9000 iteration steps. Table 2 depicts a summary of the computation time for the proposed algorithm. The value of ω used for each benchmark problem is also shown in Table 2. When the problem size is increased, more computation time is required. Fig. 6 shows the relationship between the total costs obtained by the proposed algorithm and the convergence frequency for the 30-facility layout problem. Note that the result was obtained by 1000 simulation runs using different initial uniform random input sets. Better total costs were obtained as t_limit became larger. Fig. 6 shows that our algorithm

8	6	29	34	11	42	25
3	38	40	15	7	17	41
1	37	21	9	12	32	5
16	24	4	20	26	19	18
36	30	39	14	35	10	2
23	22	33	28	31	13	27

The total cost = 7926

Fig. 4. An improved solution for the 42-facility benchmark problem.

22	23	3	48	12	40	49
44	2	31	27	43	11	39
37	16	6	38	45	29	46
30	33	5	34	20	18	28
21	8	10	35	41	14	25
19	32	15	9	42	7	26
17	4	24	1	13	36	47

The total cost = 11732

22	23	3	48	12	40	49
44	2	31	27	43	11	39
37	33	6	38	45	20	46
30	16	5	34	29	18	28
21	8	10	35	41	42	25
19	32	15	9	14	7	26
17	4	24	1	13	36	47

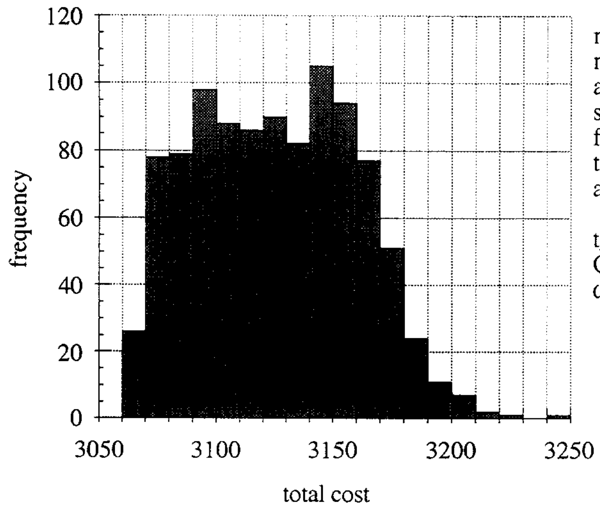
The total cost = 11732

Fig. 5. Two different improved solutions for the 49-facility benchmark problem.

Table 2
Summary of the computation time and value of ω

N-facility problem	CPU time (s)	ω
5 [16]	0.04	9
6 [16]	0.04	9
7 [16]	0.04	9
8 [16]	0.04	7
12 [16]	0.06	5
15 [16]	0.34	5
20 [16]	2.64	3
30 [16]	56.01	3
42 [20]	1746	2
49 [20]	3334	2

CPU times were measured on a HP 9000/710.



min. total cost = 3062
max. total cost = 3248
av. total cost = 3124
std. dev. = 33.83
frequency of convergence to min. total cost = 8
av. iterations = 1806

t_limit = 3000
Q = 3062
ω = 3

Fig. 6. The distribution of the total costs for the 30-facility problem.

found "good" solutions even from random initial configurations.

5. Conclusion

In this paper we have proposed a near-optimum parallel algorithm based on the two-dimensional maximum neural network for facility layout problems. The proposed algorithm uses $N \times N$ neurons for an N-facility layout problem, where only one simple parameter ω is needed to tune and "good" initial configurations are not required. The simulation results demonstrated that our algorithm is capable of generating better solutions over the existing algorithms for some of the most widely used benchmark problems. Newly discovered solutions are able to substantiate the effectiveness of the proposed algorithm.

We are going to not only solve larger size problems but also apply the proposed algorithm to other quadratic assignment problems in the future.

Acknowledgements

This work is partly supported by Fuji Electric Co., Ltd. and TEPCO Science Foundation. The authors are also thankful to Dr. J. Skorin-Kapov for providing data for the larger size problems.

Appendix A. Convergence property of the two-dimensional maximum neural network

The convergence property of the two-dimensional maximum neural network is determined by the time derivatives of the energy of the system, $\mathrm{d}E/\mathrm{d}t$. Lemma 1 is introduced to prove that the proposed system is always allowed to converge to the equilibrium state or the optimal (near-optimum) solution.

Lemma 1. $\mathrm{d}E/\mathrm{d}t \leq 0$ *is satisfied under the following two conditions:*

(1) $\mathrm{d}U_{i,j}/\mathrm{d}t = -\partial E/\partial V_{i,j} = Q - R$ *and*

(2) *The input / output function of the neuron model is given by:*

$$\begin{array}{lll}
1. & V_{\mathrm{a,b}} = 1 & \text{if } U_{\mathrm{a,b}} = \max\{U_{i,j}\}; \\
2. & V_{\mathrm{c,d}} = 1 & \text{if } U_{\mathrm{c,d}} = \max\{U_{i,j} \mid i \neq \mathrm{a}, j \neq \mathrm{b}\}; \\
3. & V_{\mathrm{e,f}} = 1 & \text{if } U_{\mathrm{e,f}} \\
 & & \quad = \max\{U_{i,j} \mid i \neq \mathrm{a,c}, j \neq \mathrm{b,d}\}; \\
\vdots & \vdots & \vdots \\
N. & V_{\mathrm{g,h}} = 1 & \text{if } U_{\mathrm{g,h}} = \max\{U_{i,j} \mid i \neq \mathrm{a,c,e},\ldots, \\
 & & \qquad j \neq \mathrm{b,d,f},\ldots\}; \\
 & V_{i,j} = 0 & \text{otherwise.}
\end{array} \tag{A.1}$$

Proof. Consider the derivatives of the computational energy E with respect to time t.

$$\frac{dE}{dt} = \sum_i \sum_j \frac{dU_{i,j}}{dt} \frac{dV_{i,j}}{dU_{i,j}} \frac{\partial E}{\partial V_{i,j}}$$

$$= -\sum_i \sum_j \left(\frac{dU_{i,j}}{dt} \right)^2 \frac{dV_{i,j}}{dU_{i,j}},$$

where $\partial E/\partial Vi,j$ is replaced by $-dU_{i,j}/dt$ (Condition 1). Let

$$\frac{dU_{i,j}}{dt} = \frac{U_{i,j}(t+dt) - U_{i,j}(t)}{dt},$$

$$\frac{dV_{i,j}}{dU_{i,j}} = \frac{V_{i,j}(t+dt) - V_{i,j}(t)}{U_{i,j}(t+dt) - U_{i,j}(t)}.$$

Assume that the input of the (a,b)-th neuron is only changed during time t and $t + dt$ in the system.

$$-\sum_i \sum_j \left(\frac{dU_{i,j}}{dt} \right)^2 \frac{dV_{i,j}}{dU_{i,j}}$$

$$= -\sum_i \sum_j \left(\frac{U_{i,j}(t+dt) - U_{i,j}(t)}{dt} \right)^2 \times \frac{V_{i,j}(t+dt) - V_{i,j}(t)}{U_{i,j}(t+dt) - U_{i,j}(t)}$$

$$= -\sum_i \sum_j \frac{U_{i,j}(t+dt) - U_{i,j}(t)}{(dt)^2} \times \left(V_{i,j}(t+dt) - V_{i,j}(t)\right)$$

$$= -\frac{U_{a,b}(t+dt) - U_{a,b}(t)}{(dt)^2} \times \left(V_{a,b}(t+dt) - V_{a,b}(t)\right).$$

It is necessary and sufficient to consider the following two cases (Conditions 1 and 2):

(1) $V_{a,b}(t+dt) = V_{a,b}(t)$;

(2) $V_{a,b}(t+dt) = 0$, $V_{a,b}(t) = 1$.

If Case 1 is satisfied, then

$$-\sum_i \sum_j \left(\frac{dU_{i,j}}{dt} \right)^2 \frac{dV_{i,j}}{dU_{i,j}} = 0$$

because $V_{a,b}(t+dt) - V_{a,b}(t) = 0$.

If Case 2 is satisfied, then

$$\sum_i \sum_j \left(\frac{dU_{i,j}}{dt} \right)^2 \frac{dV_{i,j}}{dU_{i,j}} < 0$$

because $V_{a,b}(t+dt) - V_{a,b}(t) = -1$ and

$$\frac{U_{i,j}(t+dt) - U_{i,j}(t)}{dt} = \frac{dU_{i,j}}{dt} = Q - R < 0$$

(Condition 1).

Therefore $dE/dt \leq 0$. □

References

[1] Armour, G.C., and Buffa, E.S., "A heuristic algorithm and simulation approach to relative location of facilities", *Management Science* 9 (1963) 294–309.

[2] Bazaraa, M.S., and Elshafei, A.N., "An exact branch and bound procedure for quadratic assignment problems", *Naval Research Logistics Quarterly* 26 (1979) 109–121.

[3] Bazaraa, M.S., and Sherali, M.D., "Benders' partitioning scheme applied to a new formulation of the quadratic assignment problem", *Naval Research Logistics Quarterly* 27/1 (1980) 29–41.

[4] Burkard, R.E., "Die Störungsmethode zur Lösung Quadratisches Zuordnungsprobleme", *Operations Research Verfahren* 16 (1973) 84–108.

[5] Burkard, R.E., and Bonninger, T., "A heuristic for quadratic Boolean program with applications to quadratic assignment problems", *European Journal of Operational Research* 13 (1983) 374–386.

[6] Gilmore, P.C., "Optimal and suboptimal algorithms for the quadratic assignment problem", *Journal for the Society of Industrial and Applied Mathematics* 10 (1962) 305–313.

[7] Heragu, S.S., "Recent models and techniques for solving the layout problem", *European Journal of Operational Research* 57 (1992) 136–144.

[8] Heragu, S.S., and Alfa, A.S., "Experimental analysis of simulated annealing based algorithms for the layout problem", *European Journal of Operational Research* 57 (1992) 190–202.

[9] Hillier, F.S., "Quantitative tools for plant layout analysis", *Journal of Industrial Engineering* 14 (1963) 33–40.

[10] Hillier, F.S., and Connors, M.M., "Quadratic assignment problem algorithms and the location of indivisible facilities", *Management Science* 13 (1966) 42–57.

[11] Hopfield, J.J., and Tank, D.W., "Neural computation of decisions in optimization problems", *Biological Cybernetics* 52 (1985) 141–152.

[12] Khalil, T.M., "Facilities relative allocation technique (FRAT)", *International Journal of Productions Research* 11/2 (1973) 183–194.

[13] Koopmans, T.C., and Beckman, M., "Assignment problems and the location of economic activities", *Econometrica* 25 (1957) 53–76.
[14] Kusiak, A., and Heragu, S.S., "The facility layout problem", *European Journal of Operational Research* 29 (1987) 229–251.
[15] Lawler, E.L., "The quadratic assignment problem", *Management Science* 13 (1963) 42–57.
[16] Nugent, C.E., Vollmann, T.E., and Ruml, J., "An experimental comparison of techniques for the assignment of facilities to locations", *Operations Research* 16 (1968) 150–173.
[17] Pierce, J.F., and Crowston, W.B., "Tree-search algorithms for quadratic assignment problems", *Naval Research Logistics Quarterly* 18 (1971) 1–36.
[18] Sahni, S., and Gonzalez, T., "P-complete approximation problem", *Journal of Associated Computing Machinery* 23/3 (1976) 555–565.
[19] Scriabin, M., and Vegin, R.C., "A cluster analytic approach to facility layout", *The Journal of Industrial Engineering* 18/2 (1985) 690–695.
[20] Skorin-Kapov, J., "Tabu search applied to the quadratic assignment problem", *OSRA Journal on Computing* 2/1 (1990) 33–45.
[21] Takefuji, Y., and Lee, K.C., "An artificial hysteresis binary neuron: A model suppressing the oscillatory behaviour of neural dynamics", *Biological Cybernetics* 64 (1991) 353–356.
[22] Takefuji, Y., and Lee, K.C., "An artificial maximum neural network: A winner-take-all neuron model forcing the state of the system in a solution domain", *Biological Cybernetics* 67 (1992) 243.
[23] Takefuji, Y., *Neural Network Parallel Computing*, Kluwer Academic Publishers, Boston, 1992.
[24] Wilhelm, M.R., and Ward, T.L., "Solving quadratic assignment problems by simulated annealing", *IIE Transactions* (1987) 107–119.