Applied Intelligence 7, 205–213 (1997)
© 1997 Kluwer Academic Publishers. Manufactured in The Netherlands.

A Neural Network Parallel Algorithm for Meeting Schedule Problems

KAZUHIRO TSUCHIYA
Fuji Electric Co. Research and Development, Ltd., 1, Fuji-machi, Hino, Tokyo 191, Japan; and Faculty of Environmental Information, Keio University, 5322 Endoh, Fujisawa 252, Japan
tsuchiya@sfc.keio.ac.jp

YOSHIYASU TAKEFUJI
Faculty of Environmental Information, Keio University, 5322 Endoh, Fujisawa 252, Japan; and Department of Electrical Engineering and Applied Physics, Case Western Reserve University, Cleveland, Ohio 44106, USA
takefuji@sfc.keio.ac.jp

Abstract. A parallel algorithm for solving meeting schedule problems is presented in this paper where the problem is NP-complete. The proposed system is composed of two maximum neural networks which interact with each other. One is an $M \times S$ neural network to assign meetings to available time slots on a timetable where M and S are the number of meetings and the number of time slots, respectively. The other is an $M \times P$ neural network to assign persons to the meetings where P is the number of persons. The simulation results show that the state of the system always converges to one of the solutions. Our empirical study shows that the solution quality of the proposed algorithm does not degrade with the problem size.

Keywords: neural network, parallel algorithm, scheduling, meeting schedule

1. Introduction

It was reported by Teger that an executive spends nearly 50 percent of the time on the job for informal and formal meetings, and that even a manager spends 47 percent of his/her working time on such face-to-face type meetings [11]. Scheduling of meetings is one of the fundamental functions of office automation but it is complex and time-consuming. To find a meeting schedule solution which satisfies all the given constraints is one of the NP-complete problems [1]. Although there exist a large number of scheduling problems, there are few practical meeting schedule problems presented in literature. Sugihara et al. proposed a meeting scheduler in 1989, applying a heuristic algorithm for a timetable rearrangement problem [6]. When a new meeting is added to an existing meeting schedule, the time complexity of their rearrangement algorithm becomes $O(M^2P^2N^qT^2)$ where M, P, N, q, and T are the number of meetings, the number of persons, $\max\{|o_i(\text{st}, t(m_{\text{new}}))|\} + 1$, the number of attendants of the new meeting, and the maximum number of time slots among all the meetings, respectively. Note that $o_i(\text{st})$ is an ordered set of the meetings which person #i attends from a time slot st for the duration of a new meeting $t(m_{\text{new}})$ when person #i attends the new meeting. Their algorithm does not always provide optimum solutions. In 1991, Sen et al. proposed a formal study of distributed meeting scheduling using alternative heuristic strategies and a multistage negotiation protocol [5]. Their method is only in the preliminary stage and cannot deal with larger or complex problems although it could lead to a parallel and distributed system. To our knowledge, no parallel algorithm for meeting schedule problems has been proposed. This paper introduces a neural network parallel algorithm for meeting schedule problems.

The first neural network using sigmoid neurons was proposed by Hopfield and Tank for optimization problems [2]. Szu used the McCullock-Pitts neural network for the traveling salesman problem [7]. Takefuji et al. have proposed the hysteresis McCullock-

Pitts neural network and the maximum (winner-take-all) neural network for several NP-complete problems [3, 4, 8–10, 12, 13]. Our algorithm uses two maximum neural networks which interact with each other.

2. Meeting Schedule Problem

The following four conditions based on Sugihara et al. [6], which should not be violated when setting up the meeting schedule, are considered in our scheduling problems:

1) No person attends more than one meeting at the same time.
2) Meetings have priority where "$m_i < m_j$" means that meeting m_j must start after meeting m_i ends.
3) For any meeting m_i, persons are divided into a set of groups $g(m_i)$ where one and only one person from each of the groups can attend meeting m_i.
4) Each meeting m_i starts at one of the available time slots $s(m_i)$ and takes the amount of time $t(m_i)$ as the duration.

Consider an 8-person, 5-meeting, 16-time-slot problem as an example:

a) Persons $= \{1, 2, 3, 4, 5, 6, 7, 8\}$.
b) Meetings $= \{m_1, m_2, m_3, m_4, m_5\}$.
c) Meeting priority: $m_1 < m_5$ and $m_2 < m_4$.
d) Duration of the five meetings: $t(m_1) = 2, t(m_2) = 4, t(m_3) = 3, t(m_4) = 3, t(m_5) = 2$.
e) Groups: $g(m_1) = \{\{2, 3\}, \{7, 8\}\}$, $g(m_2) = \{\{1\}, \{2\}, \{3\}, \{4\}, \{7\}, \{8\}\}$, $g(m_3) = \{\{2\}, \{5\}, \{6, 7, 8\}\}$, $g(m_4) = \{\{1, 2\}, \{3, 4\}, \{6, 7, 8\}\}$, $g(m_5) = \{\{1, 2\}, \{3, 4\}, \{6, 7\}\}$.
f) Available time slots: $s(m_1) = \{1, 2, 3, 4, 5\}$, $s(m_2) = \{3, 4\}$, $s(m_3) = \{3, 4, 9, 10\}$, $s(m_4) = \{k: 3 \leq k \leq 13\}$, $s(m_5) = \{k: 1 \leq k \leq 15\}$.

Here, for example, "$g(m_3) = \{\{2\}, \{5\}, \{6, 7, 8\}\}$" means that the first and the second group must send person #2 and person #5 to meeting m_3, respectively, and that the third group must send one among persons #6, #7, and #8, and "$s(m_2) = \{3, 4\}$" means that meeting m_2 can start at either time slot 3 or time slot 4.

Scheduling of meetings requires two tasks under the above conditions. One is meeting-assignment to assign meetings to available time slots on a timetable and the other is person-assignment to assign persons to the meetings. Our neural network algorithm yields both the meeting-assignment and the person-assignment scheduling.

3. Neural Representation

The mathematical model of an artificial neural network consists of two components: neurons and synaptic links. The output signal transmitted from a neuron propagates through the synaptic links to other neurons as one of their inputs. Therefore, every artificial neuron has the input U and the output V. 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 which is determined by the neuron model. The input of the (i, j)th neuron $U_{i,j}$ is updated by the motion equation which represents the synaptic links. The motion equation of the (i, j)th neuron is generally given by:

$$\frac{dU_{i,j}}{dt} = -\frac{\partial E(V_{1,1}, \ldots, V_{i,j}, \ldots, V_{M,P})}{\partial V_{i,j}}$$

where E is the computational energy function following an $M \times P$-variable function: $E(V_{1,1}, \ldots, V_{i,j}, \ldots, V_{M,P})$. The artificial neural network provides a gradient descent method so as to minimize the fabricated energy function E. Usually, the left term in the above equation is directly constructed by considering the necessary and sufficient constraints and/or the cost function from the given problem to update $U_{i,j}$. In our meeting schedule problem, the cost function is not necessary for the motion equation.

Our system is composed of two neural networks which interact with each other to solve meeting schedule problems. An $M \times S$ neural network array is used for the meeting-assignment and an $M \times P$ neural network array for the person-assignment where S is the number of time slots. One of the solutions is provided in Fig. 1 for the above 8-person, 5-meeting, 16-time-slot problem. Figure 1 depicts the states of the two neural network arrays. The left side of Fig. 1 shows the state of the 5×16 meeting-assignment neural network and the right shows the 5×8 person-assignment where each square represents a state of the (m, s)th neuron and the (m, p)th neuron, respectively. The black squares show that the outputs of the neurons generate 1's. For example, in the meeting-assignment neural network, the $(1, 1)$th neuron's output is 1. It means that meeting m_1 starts on time slot 1. The $(1, 2)$th and the $(1, 7)$th neurons in the person-assignment neural network of Fig. 1 generate 1's, which means that persons #2 and #7 attend meeting m_1. The state of the

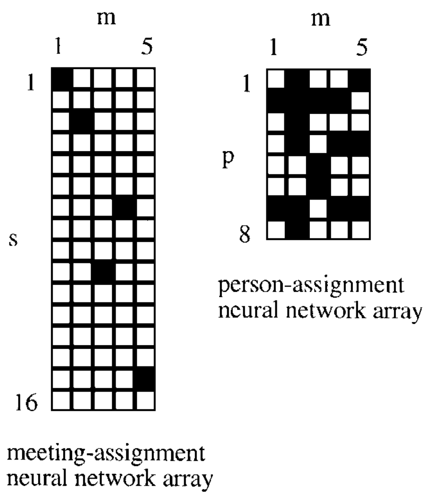

Figure 1. A solution for problem #1 (8-person, 5-meeting, 16-time-slot problem.

(i, j)th neuron is determined by the following neuron models and motion equations.

The meeting-assignment must satisfy the condition that meeting m_i is assigned to one of the time slots which is constrained by $s(m_i)$. To cope with this constraint, the meeting-assignment maximum neuron is used. The input/output function of the (i, j)th neuron is given by:

$$Vm_{i,j} = \begin{cases} 1 & \text{if } Um_{i,j} = \max\{Um_{i,s} \mid s \in s(m_i)\}; \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $Vm_{i,j}$ and $Um_{i,j}$ are the output and the input of the (i, j)th neuron for the meeting-assignment neural network, respectively, and i and j are the meeting number and the time slot, respectively. The equation "$Vm_{i,j} = 1$" means that meeting m_i starts at time slot j.

The motion equation represents interconnections between the (i, j)th neuron and other neurons. The motion equation for the meeting-assignment neural network includes two violation functions corresponding to two constraints. One is the meeting-priority violation function: $\mathrm{mp}(x, y)$ which generates 1 if meeting m_y must start after meeting m_x ends and 0 otherwise. The other is the meeting-overlap violation function: $\mathrm{ovlp}(\mathrm{st}(m_x), t(m_x), \mathrm{st}(m_y), t(m_y))$ which becomes 1 if two meetings m_x and m_y overlap and 0 otherwise. Note that $\mathrm{st}(m_x)$ is the starting time slot of meeting m_x. The overlap condition for meetings m_x and m_y is given by: $\mathrm{st}(m_x) < \mathrm{st}(m_y) < \mathrm{st}(m_x) + t(m_x)$, $\mathrm{st}(m_x) < \mathrm{st}(m_y) + t(m_y) < \mathrm{st}(m_x) + t(m_x)$, $\mathrm{st}(m_y) < \mathrm{st}(m_x) < \mathrm{st}(m_y) + t(m_y)$, or $\mathrm{st}(m_y) < \mathrm{st}(m_x) + t(m_x) < \mathrm{st}(m_y) + t(m_y)$. The motion equation of the (m, s)th neuron for meeting m_m and time slot s is given by:

$$\begin{aligned}\frac{dUm_{m,s}}{dt} = &-A \sum_{\substack{i=1\\ i\neq m}}^{M} \left\{ \sum_{j=1}^{s+t(m_m)-1} \mathrm{mp}(m, i) Vm_{i,j} + \sum_{j=s-t(m_i)+1}^{S} \mathrm{mp}(i, m) Vm_{i,j} \right\} \\ &- B \left\{ \sum_{p=1}^{P} \sum_{\substack{d=1\\ d\neq m}}^{M} Vg_{m,p} Vg_{d,p} \mathrm{ovlp}(\mathrm{st}(m_m), t(m_m), \mathrm{st}(m_d), t(m_d)) \right\} \end{aligned} \quad (2)$$

where A and B are constant coefficients. Note that $Vg_{x,y}$ is the output of the (x, y)th neuron in the person-assignment neural network. $Vg_{x,y}$ becomes 1 if person #y attends meeting m_x and 0 otherwise. The first term in Eq. (2) describes the meeting-priority violation forces: the meeting-priority violation in a range of 1 to time slot $s + t(m_m) - 1$, and that in a range of $s - t(m_i) + 1$ to the end of the time slots. The second term in Eq. (2) is the inhibitory force to eliminate overlaps of meetings where meetings m_m and m_d should not be overlapped if the same person attends the meetings m_m and m_d. These two terms discourage the (m, s)th neuron from generating a nonzero output if it violates the above conditions.

In order to accelerate the simulation speed, the following Eq. (2′) is used instead of Eq. (2):

If $(t \bmod 10) < 8$ then

$$\begin{aligned}\frac{dUm_{m,s}}{dt} = &-A \sum_{\substack{i=1\\ i\neq m}}^{M} \left\{ \sum_{j=1}^{s+t(m_m)-1} \mathrm{mp}(m, i) Vm_{i,j} + \sum_{j=s-t(m_i)+1}^{S} \mathrm{mp}(i, m) Vm_{i,j} \right\} Vm_{m,s} \\ &- B \left\{ \sum_{p=1}^{P} \sum_{\substack{d=1\\ d\neq m}}^{M} Vg_{m,p} Vg_{d,p} \mathrm{ovlp}(\mathrm{st}(m_m), t(m_m), \mathrm{st}(m_d), t(m_d)) \right\} Vm_{m,s} \end{aligned}$$

else

$$\frac{dUm_{m,s}}{dt} = -A\sum_{\substack{i=1\\i\neq m}}^{M}\left\{\sum_{j=1}^{s+t(m_m)-1}\mathrm{mp}(m,i)Vm_{i,j} + \sum_{j=s-t(m_i)+1}^{S}\mathrm{mp}(i,m)Vm_{i,j}\right\} - B\left\{\sum_{p=1}^{P}\sum_{\substack{d=1\\d\neq m}}^{M}Vg_{m,p}Vg_{d,p}\mathrm{ovlp}(\mathrm{st}(m_m), t(m_m),\mathrm{st}(m_d),t(m_d))\right\} \qquad (2')$$

where t is the number of iteration steps. The first equation is activated only for neurons generating nonzero outputs while the second one is for all neurons. This method helps the state of the system to escape from the local minimum [9]. The convergence of the maximum neural network is attached in Appendix 1.

The other $M \times P$ neural network array is used to assign persons to the meetings where one and only one person from each group must attend the meeting. The person-assignment maximum neuron is employed to satisfy the condition. The input/output function of the (i, j)th neuron is given by:

$$Vg_{i,j} = \begin{cases} 1 & \text{if } Ug_{i,j} = \max\{Ug_{i,p} \mid p \in g_k(m_i)\}; \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

where $Vg_{i,j}$ and $Ug_{i,j}$ are the output and the input of the (i, j)th neuron for the person-assignment neural network, respectively, and i, j, and $g_k(m_i)$ are the meeting number, the person number, and the kth group in meeting m_i, respectively. The equation "$Vg_{i,j} = 1$" means that person $\#j$ attends meeting m_i.

Since the neuron model includes one of the constraints, the motion equation of the (m, p)th neuron for meeting m_m and person $\#p$ is simply given by:

$$\frac{dUg_{m,p}}{dt} = -C\sum_{\substack{d=1\\d\neq m}}^{M}Vg_{d,p} \times \mathrm{ovlp}(\mathrm{st}(m_m), t(m_m), \mathrm{st}(m_d), t(m_d)) \qquad (4)$$

where C is a constant coefficient. This equation prohibits one person $\#p$ from attending different meetings m_m and m_d simultaneously. In other words, if the (m, p)th neuron violates the condition, then the input decreases so that its output generates zero.

In order to accelerate the simulation speed and help the system to escape from the local minimum, the following Eq. (4′) is used instead of Eq. (4):

If $(t \bmod 10) < 8$ then

$$\frac{dUg_{m,p}}{dt} = -CVg_{m,p}\sum_{\substack{d=1\\d\neq m}}^{M}Vg_{d,p} \times \mathrm{ovlp}(\mathrm{st}(m_m), t(m_m), \mathrm{st}(m_d), t(m_d))$$

else

$$\frac{dUg_{m,p}}{dt} = -C\sum_{\substack{d=1\\d\neq m}}^{M}Vg_{d,p} \times \mathrm{ovlp}(\mathrm{st}(m_m), t(m_m), \mathrm{st}(m_d), t(m_d)) \qquad (4')$$

Notice that all the necessary and sufficient constraints for the meeting schedule problems are taken into account in the neuron models and motion equations shown above.

4. Parallel Algorithm

The outputs of the neurons, determined by Eq. (3), for the person-assignment neural network are used in Eq. (2′) for the meeting-assignment neural network, and those determined by Eq. (1) for the meeting-assignment neural network affect the meeting-overlap violation function in Eq. (4′) for the person-assignment neural network. In this paper, a synchronous parallel system is used to simulate the interaction between the two neural networks where the input states of all the neurons for the person-assignment neural network are updated by Eq. (4′) simultaneously, and all the outputs are evaluated at the same time by Eq. (3), then all the inputs for the meeting-assignment neural network are updated by Eq. (2′) simultaneously, and all the outputs are evaluated by Eq. (1) at the same time, as shown in the following procedure.

Step 0. Set $t = 0$ and $A = 5$, $B = C = 1$.

Step 1. Assign uniformly generated random numbers to the initial values of $Um_{m,s}(0)$ and $Ug_{m,p}(0)$ for $m = 1, \ldots, M$, $s = 1, \ldots, S$, and $p = 1, \ldots, P$.

Step 2. Evaluate $Vm_{m,s}(0)$ and $Vg_{m,p}(0)$ for $m = 1, \ldots, M$, $s = 1, \ldots, S$, and $p = 1, \ldots, P$, using Eqs. (1) and (3), respectively.

Table 1. Summary of the results.

Problem #	The # of persons	The # of meetings	The # of time-slots	Av. # of iteration steps	Standard deviation	Av. comp. time (sec)	The # of conv./ the # of trials
1	8	5	16	47.0	50.5	0.03	1000/1000
2	15	10	17	45.9	37.0	0.06	1000/1000
3	29	15	18	38.6	20.7	0.14	1000/1000
4	29	20	18	55.9	50.5	0.27	1000/1000
5	35	12	23	88.1	107.5	0.59	1000/1000
6	35	18	23	132.1	125.5	1.4	1000/1000
7	35	24	23	96.6	118.3	0.86	1000/1000
8	35	24	31	77.9	90.5	0.85	1000/1000
9	39	24	31	86.9	111.0	1.1	1000/1000
10	45	24	31	92.2	100.4	1.5	1000/1000

#: number; Av.: Average; comp.: computational; conv.: convergence; Av. comp. times were measured on an HP 9000/710.

Step 3. Compute Eq. (4′) of the $M \times P$ person-assignment neural network for $m = 1, \ldots, M$ and $p = 1, \ldots, P$ to obtain $\Delta Ug_{m,p}(t)$:

$$\Delta Ug_{m,p}(t) = \frac{dUg_{m,p}}{dt} \tag{5}$$

Step 4. Update $Ug_{m,p}(t+1)$ for $m = 1, \ldots, M$ and $p = 1, \ldots, P$, based on the first-order Euler method:

$$Ug_{m,p}(t+1) = Ug_{m,p}(t) + \Delta Ug_{m,p}(t) \tag{6}$$

Step 5. Evaluate $Vg_{m,p}(t+1)$ for $m = 1, \ldots, M$ and $p = 1, \ldots, P$ using Eq. (3) and increment t by 1.

Step 6. Compute Eq. (2′) of the $M \times S$ meeting-assignment neural network for $m = 1, \ldots, M$ and $s = 1, \ldots, S$ to obtain $\Delta Um_{m,s}(t)$:

$$\Delta Um_{m,s}(t) = \frac{dUm_{m,s}}{dt} \tag{7}$$

Step 7. Update $Um_{m,s}(t+1)$ for $m = 1, \ldots, M$ and $s = 1, \ldots, S$, based on the first-order Euler method:

$$Um_{m,s}(t+1) = Um_{m,s}(t) + \Delta Um_{m,s}(t) \tag{8}$$

Step 8. Evaluate $Vm_{m,s}(t+1)$ for $m = 1, \ldots, M$ and $s = 1, \ldots, S$ using Eq. (1) and increment t by 1.

Step 9. If $Vm_{m,s} = 1$ and $Um_{m,s}(t) = 0$ for $m = 1, \ldots, M$ and $s = 1, \ldots, S$, and $Vg_{m,p} = 1$ and $\Delta Ug_{m,p}(t) = 0$ for $m = 1, \ldots, M$ and $p = 1, \ldots, P$, then terminate this procedure otherwise go to Step 3.

The first-order Euler method is used in Eqs. (6) and (8) to update the inputs. Step 9 shows the termination condition in which all the necessary and sufficient constraints for a meeting schedule problem are satisfied. Note that the values of the coefficients set in Step 1 does not affect the results very much although A is set bigger than B so that meeting priority is satisfied first.

5. Experimental Results

The proposed algorithm was implemented on a Macintosh PowerBook 170 computer and an HP 9000/710 computer, although the algorithm is executable both on a sequential machine and a parallel one. We have examined ten problems to test our algorithm. Table 1 summarizes the results. Note that these results were obtained when one thousand simulation runs were performed in every problem by using different initial states with the uniformly generated random numbers. The number of persons and the number of meetings were almost doubled from problems #1 to #2 and #2 to #4 while the number of time slots was increased 1 slot per problem. Namely, the meeting schedule became denser and the scheduling became more complex. The simulation results showed, however, that the average number of iteration steps for each problem is almost the same. The average number of iteration steps increased gradually but only slightly with the problem size for other problems. Some of the typical distributions of the number of iteration steps to converge

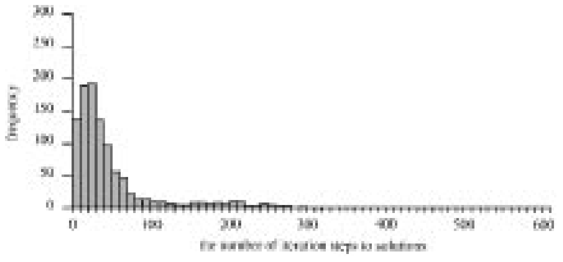

Figure 2. The distribution of the convergence iteration steps for problem #1.

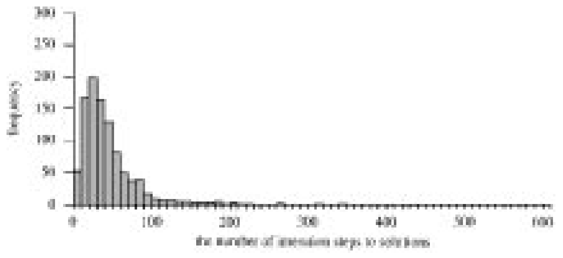

Figure 3. The distribution of the convergence iteration steps for problem #3.

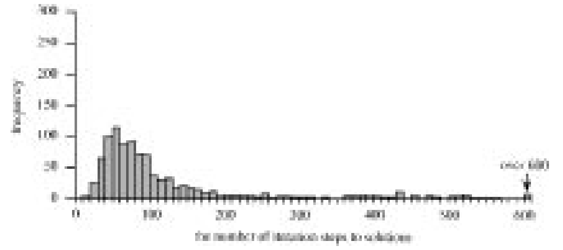

Figure 4. The distribution of the convergence iteration steps for problem #6.

to the solutions are shown in Figs. 2, 3, 4, and 5 for problems #1, #3, #6, and #9, respectively. They show that the average number of iteration steps increased because the distribution became larger. Average computational times were measured on an HP 9000/710 computer. The average computational time increased with the problem size. Notice that, however, the average computational time on a parallel computer is supposed to be proportional to the average number of iteration steps. Through the simulation runs, the state of the system always converged to one of the solutions within 900 iterations for every problem. The algorithm obtained several solutions for every problem from the different initial values of $Ug_{m,p}$ and $Um_{m,s}$. Figures 1, 6, 7, and 8 depict one of the solutions for problems #1, #3, #6, and #9, respectively.

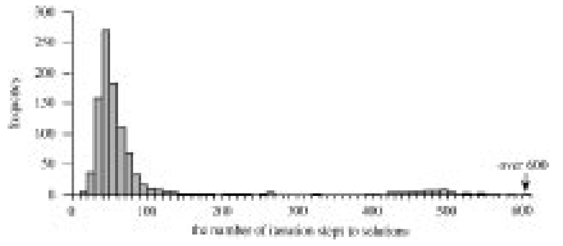

Figure 5. The distribution of the convergence iteration steps for problem #9.

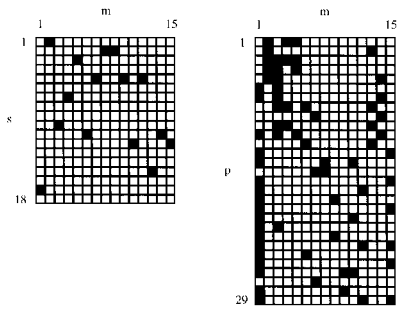

Figure 6. A solution for problem #3 (29-person, 15-meeting, 18-time-slot problem).

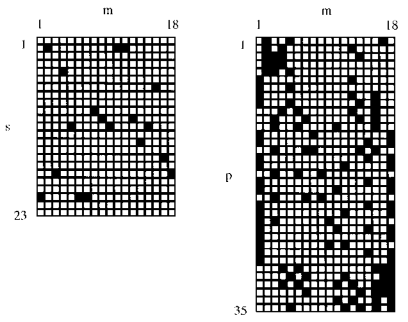

Figure 7. A solution for problem #6 (35-person, 18-meeting, 23-time-slot problem).

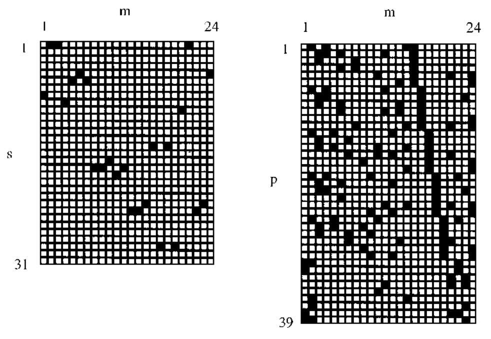

Figure 8. A solution for problem #9 (39-person, 24-meeting, 31-time-slot problem).

6. Conclusion

We proposed a parallel algorithm using two maximum neural networks for meeting schedule problems in office automation. The proposed algorithm requires $M \times S$ and $M \times P$ neural networks for the meeting-assignment and the person-assignment, respectively. Our neuron models and motion equations satisfy all the constraints for meeting schedule problems by a gradient descent method where the inputs and the outputs are updated in parallel. The two neural networks interact with each other to generate meeting schedule solutions yielding both the meeting-assignment and the person-assignment scheduling. The state of the system always converged to a solution. The algorithm also obtained several different solutions for each problem. Simulation results showed that the solution quality of the proposed algorithm does not degrade with the problem size.

We are planning to apply the proposed algorithm to other kinds of scheduling problems in which "meeting" could be renamed as "process" and "person" could be "instrument". We will also try to install the system on a parallel machine in the future.

Appendix 1

Convergence Property of the Maximum Neural Network

Convergence property of the maximum neural network is determined by the time derivatives of energy of the system, $\frac{dE}{dt}$ where E is the energy function. In Lemma 1, the convergence of the maximum neural network for the meeting schedule problem is given, where for convenience, only the meeting-assignment neural network is considered because the convergence of the person-assignment neural network can also be proved in the same way as for the meeting-assignment neural network.

Lemma 1. $\frac{dE}{dt} \leq 0$ *is satisfied under two conditions such as*

(1) $\frac{dUm_{i,j}}{dt} = -\frac{\partial E}{\partial Vm_{i,j}}$ *and*
(2) $Vm_{i,j} = 1$ *if* $Um_{i,j} = \max\{Um_{i,s} \mid s \in s(m_i)\}$ *and* 0 *otherwise*

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

$$\frac{dE}{dt} = \sum_i \sum_j \frac{dUm_{i,j}}{dt} \frac{dVm_{i,j}}{dUm_{i,j}} \frac{\partial E}{\partial Vm_{i,j}}$$
$$= -\sum_i \sum_j \left(\frac{dUm_{i,j}}{dt}\right)^2 \frac{dVm_{i,j}}{dUm_{i,j}}$$

where $\frac{\partial E}{\partial Vm_{i,j}}$ is replaced by $-\frac{dUm_{i,j}}{dt}$ (condition 1). Let $\frac{dUm_{i,j}}{dt}$ be $\frac{Um_{i,j}(t+dt)-Um_{i,j}(t)}{dt}$. Let $\frac{dVm_{i,j}}{dUm_{i,j}}$ be $\frac{Vm_{i,j}(t+dt)-Vm_{i,j}(t)}{Um_{i,j}(t+dt)-Um_{i,j}(t)}$. Let us consider the term $\sum_i(\frac{dUm_{i,j}}{dt})^2 \frac{dVm_{i,j}}{dUm_{i,j}}$ for each meeting separately. Let $Um_{i,a}(t+dt)$ be the maximum at time $t+dt$ and $Um_{i,b}(t)$ be the maximum at time t for meeting m_i:

$$Um_{i,a}(t+dt) = \max\{Um_{i,s}(t+dt) \mid s \in s(m_i)\}$$
$$Um_{i,b}(t) = \max\{Um_{i,s}(t) \mid s \in s(m_i)\}$$

It is necessary and sufficient to consider the following two cases:

1) $a = b$
2) $a \neq b$

If Case 1) is satisfied, then there is no state change for meeting m_i. Consequently, $\sum_i(\frac{dUm_{i,j}}{dt})^2 \frac{dVm_{i,j}}{dUm_{i,j}}$ must be zero.
If Case 2) is satisfied, then

$$\begin{aligned}
&\sum_i \left(\frac{dUm_{i,j}}{dt}\right)^2 \frac{dVm_{i,j}}{dUm_{i,j}} \\
&= \left(\frac{Um_{i,a}(t+dt)-Um_{i,a}(t)}{dt}\right)^2 \frac{Vm_{i,a}(t+dt)-Vm_{i,a}(t)}{Um_{i,a}(t+dt)-Um_{i,a}(t)} \\
&\quad + \left(\frac{Um_{i,b}(t+dt)-Um_{i,b}(t)}{dt}\right)^2 \frac{Vm_{i,b}(t+dt)-Vm_{i,b}(t)}{Um_{i,b}(t+dt)-Um_{i,b}(t)} \\
&= \left(\frac{Um_{i,a}(t+dt)-Um_{i,a}(t)}{dt}\right)^2 \frac{1}{Um_{i,a}(t+dt)-Um_{i,a}(t)} \\
&\quad + \left(\frac{Um_{i,b}(t+dt)-Um_{i,b}(t)}{dt}\right)^2 \frac{-1}{Um_{i,b}(t+dt)-Um_{i,b}(t)} \\
&= \frac{Um_{i,a}(t+dt)-Um_{i,a}(t)}{(dt)^2} - \frac{Um_{i,b}(t+dt)-Um_{i,b}(t)}{(dt)^2} \\
&= \frac{1}{(dt)^2}\{Um_{i,a}(t+dt)-Um_{i,a}(t)-Um_{i,b}(t+dt) \\
&\quad + Um_{i,b}(t)\} \\
&= \frac{1}{(dt)^2}\{Um_{i,a}(t+dt)-Um_{i,b}(t+dt)+Um_{i,b}(t) \\
&\quad - Um_{i,a}(t)\} \\
&> 0
\end{aligned}$$

because $Um_{i,a}(t+dt)$ is the maximum at time $t+dt$ and $Um_{i,b}(t)$ is the maximum at time t for meeting m_i.

The contribution from each term is either 0 or positive, therefore

$$\sum_i \left(\frac{dUm_{i,j}}{dt}\right)^2 \frac{dVm_{i,j}}{dUm_{i,j}} \geq 0$$

and

$$-\sum_i \sum_j \left(\frac{dUm_{i,j}}{dt}\right)^2 \frac{dVm_{i,j}}{dUm_{i,j}} \leq 0 \Rightarrow \frac{dE}{dt} \leq 0$$

□

Lemma 1 states that the solution quality improves as time elapses until no further improvement can be achieved and that the state of the system finally reaches an equilibrium state or the optimal (near-optimum) solution.

References

1. M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman: San Francisco, CA, 1979.
2. J. Hopfield and D. Tank, "Neural computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.
3. K.C. Lee and Y. Takefuji, "A generalized maximum neural network for the module orientation problem," *Int. J. Electronics*, vol. 72, no. 3, pp. 331–335, 1992.
4. K.C. Lee, N. Funabiki, and Y. Takefuji, "A parallel improvement algorithm for bipartite subgraph problem," *IEEE Trans. on Neural Networks*, vol. 3, no. 1, pp. 139–145, 1992.
5. S. Sen and E.H. Durfee, "A formal study of distributed meeting scheduling: Preliminary results," *SIGOIS Bulletin*, vol. 12, nos. 2/3, pp. 55–68, 1991.
6. K. Sugihara, T. Kikuno, and N. Yoshida, "A meeting scheduler for office automation," *IEEE Trans. on Software Engineering*, vol. 15, no. 10, pp. 1141–1146, 1989.
7. H. Szu, "Fast TSP algorithm based on binary neuron output and analog input using the zero-diagonal interconnect matrix and necessary and sufficient constraints of the permutation matrix," in *Proc. of the International Conference on Neural Networks*, IEEE: Piscataway, NJ, 1988, vol. II, pp. 59–266.
8. Y. Takefuji, *Neural Network Parallel Computing*, Kluwer Academic Publishers: Norwell, MA, 1992.
9. Y. Takefuji and K.C. Lee, "A parallel algorithm for tiling problems," *IEEE Trans. on Neural Networks*, vol. 1, no. 1, pp. 143–145, 1990.
10. Y. Takefuji and K.C. Lee, "Artificial neural networks for four-coloring map problem and k-colorability problems," *IEEE Trans. on Circuits Sys.*, vol. 38, no. 3, pp. 326–333, 1991.
11. S.L. Teger, "Factor impacting the evolution of office automation," *Proc. IEEE*, vol. 71, no. 4, pp. 503–511, 1983.
12. K. Tsuchiya and Y. Takefuji, "A neural network algorithm for the no-three-in-line problem," *Neurocomputing*, vol. 8, pp. 43–49, 1995.
13. K. Tsuchiya, S. Bharitkar, and Y. Takefuji, "A neural network approach to facility layout problems," to appear in *European Journal of Operational Research*.

Kazuhiro Tsuchiya received his B.S. (1983) in chemical engineering and M.S. (1985) in material chemistry from Tohoku University (Japan). He has worked for Fuji Electric Co., Ltd. in Japan since 1985. He has also been a research associate in Keio University since 1994. He was a visiting researcher at Department of Electrical Engineering and Applied Physics in Case Western Reserve University from 1991 through 1994. His research interests focus on neural network parallel computing for solving real-world problems such as graph theory problems, operations research problems, management science problems, and VLSI design. He is also interested in the VLSI applications and silicon architecture. He is a member of the IEEE Control Society and International Neural Network Society.

Yoshiyasu Takefuji has been on the tenured faculty of Electrical Engineering at Case Western Research University since 1988 and has also been on the faculty of Keio University since 1992. His research interests focus on neural computing for solving real-world problems. He is interested in VLSI applications and silicon architecture. He received the National Science Foundation/Research Initiation Award in 1989 and received the distinct service award from IEEE Trans. on Neural Networks in 1992 and has been an NSF advisory panelist. A member of the IEEE Computer Society, International Neural Network Society, he received the Information Processing Society of Japan's best paper award in 1980. He received the TEPCO research award in 1993/1994/1995 respectively. He received the Kanagawa Academy of Science and Technology research award in 1993/1994/1995 respectively. He also received the Takayanagi research award in 1995. He authored ten books: Neural Network parallel computing from Kluwer Academic Pub. in 1992, Analog VLSI neural networks from KAP in 1993, Neural networks in design and manufacturing from World Scientific Pub. in 1993, Neural computing for optimization and combinatorics from WSP in 1996, netiquette from Kyoritsu in 1996, neural computing from Corona-sha in 1996, neural networks from Sangyo-tosho in 1996, CALS: Social revolution from JustSystem in 1995, Digital circuits from Ohm-sha in 1984, and Neurocomputing from Baifukan Pub. in 1992. He was an Editor of the Journal of Neural Network Computing, associate editor of IEEE Trans. on Neural Networks, Neural/parallel/scientific computations, and Neurocomputing, and a guest editor of Journal Analog Integrated Circuits and Signal Processing in the special issue on analog VLSI neural networks and also guest editor of Neurocomputing in the special issue on neural network optimization. He is currently an associate editor of the international journal of multimedia tools and applications and also a guest editor of the special issue on multimedia projects in Asia. He has published more than 140 papers. He is included in who's who in America, who's who in the Midwest, who's who in science and engineering, and men of achievement. He is currently an advisor to the Hong Kong government on neural computing.