Pergamon

0952-1976(94)E0003-5

Engng Applic. Artif. Intell. Vol. 7, No. 3, pp. 321–327, 1994

0952-1976/94 $7.00 + 0.00

Contributed Paper

Job-Shop Scheduling Based on Modified Tank–Hopfield Linear Programming Networks

SIMON Y. FOO
Florida State University

YOSHIYASU TAKEFUJI
Case Western Reserve University

HAROLD SZU
Naval Surface Warfare Center

(Received August 1993)

The Tank–Hopfield linear programming network is modified to solve job-shop scheduling, a classical optimization problem. Using a linear energy function, the approach described in this paper avoids the traditional problems associated with most Hopfield networks using quadratic energy functions. Although this approach requires more hardware (in terms of processing elements and resistive interconnects) than a recent approach by Zhou et al. (IEEE Trans. Neural Networks 2, 175–179, 1991) the neurons in the modified Tank–Hopfield network do not perform extensive calculations, unlike those described by Zhou et al.

Keywords: Job-shop scheduling, Hopfield neural networks, mixed integer-linear programming.

1. INTRODUCTION

Hopfield neural networks have been successfully applied to solving a variety of interesting problems such as the travelling-salesman problem, analog-to-digital conversion,[2] resource allocation, tiling problem, k-colorability problem, and many others as described by Takefuji.[3] Recent criticisms of the Hopfield networks such as nonconvergence of the network to valid solutions, inability to locate the global minimum, and poor scaling properties, are due to the use of quadratic energy functions, as pointed out by Zhou *et al.*[1] This work, as an extension of earlier results reported in Ref 4, uses a modified Tank–Hopfield linear programming network with a linear energy function which overcomes the shortcomings of the traditional quadratic energy functions.

2. JOB-SHOP SCHEDULING

Job-shop scheduling belongs to the large class of NP-complete (nondeterministic polynomial time complete) problems. An NP-complete problem exhibits an exponential growth in the computation time as the size of the problem increases linearly, and is commonly referred to as a "hard" problem. In general, an NP-complete problem is one which, for some input of size N, takes a time proportional to at least 2^N. Conversely, a non-NP (deterministic polynomial time) problem shows a linear growth in the computation time as the size of the problem increases linearly. An example of non-NP problems is the application of an insertion sort routine on a table of N entries, where the computation time is proportional to N^2. NP-complete problems occur naturally in many situations. For example, there are more than $N!$ ways to schedule N courses to various classrooms and instructors, which means that determining a complete schedule of a large number of classes pleasing to both students and professors can become a formidable task!

In general, the job-shop scheduling problem can be stated as follows: given n jobs that have to be processed on m machines in a prescribed order under certain restrictive assumptions, what is then the optimal order in which each machine handles the jobs? In short, job-shop scheduling is a resource-allocation problem. The resources are called *machines* and basic tasks are called *jobs*. Each job may consist of several subtasks, referred

Correspondence should be sent to: Dr S. Y. Foo, Department of Electrical Engineering, FAMU/FSU College of Engineering, Florida State University, Tallahassee, FL 32316, U.S.A.

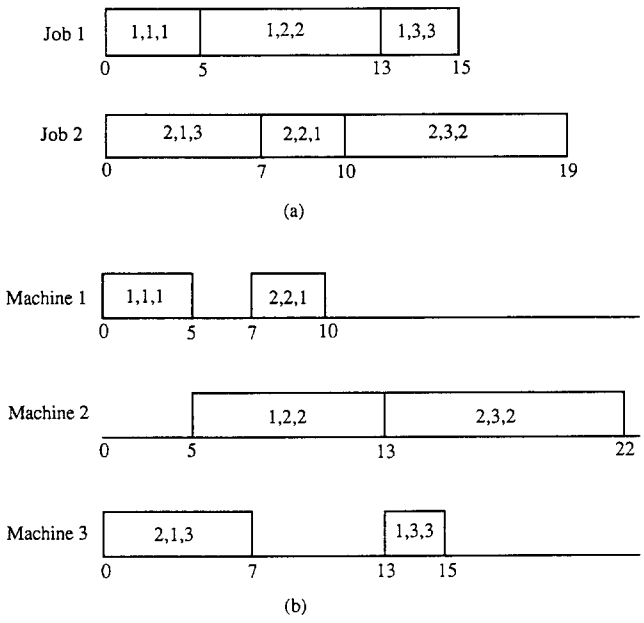

Fig. 1. (a) An example 2-job 3-machine job-shop problem. (b) The optimum schedule as produced by the network.

to as *operations*, that are interrelated by precedence restrictions. Closely resembling the job-shop problem is processor scheduling in a multiprocessor computer system where the problem definition is: given a deadline and a set of tasks of varying lengths to be performed on two identical processors, can the tasks be arranged so that the deadline is met?

In a general job-shop problem, each operation is described by a triplet (i, j, k), i.e. operation j of job i is to be executed on machine k. Assuming there are m machines, then each job has exactly m operations, with exactly one operation on each machine. If there are n jobs, then each machine must perform n operations, which means that the number of possible sequences is therefore $n!$ for each machine. If the sequences on each machine are independent, there are $(n!)^m$ schedules. However, since each job consists of several operations with a linear precedence structure, many of these schedules will have conflicts, and therefore are invalid solutions.

An example 2-job 3-machine job-shop problem is shown by a Gantt chart in Fig. 1(a) Each job-operation-machine triplet (i, j, k) is represented by a block; the length of block is equal to the processing time t_{ijk} required to perform the operation, while the numbers on the horizontal axis represent completion times. Note that the processing order of each job by all machines and processing time of each operation are assumed known and fixed (static).

A feasible schedule is one where all operations of each job can be placed on one time axis in precedence order and without overlap. In principle, there are infinitely many feasible schedules for a given job-shop problem, since an arbitrary amount of idle time can be inserted at any machine between pairs of operations. Superfluous idle time exists in a schedule if an operation can be processed earlier in time without altering the operation sequences on any machine. In a nondelay schedule, no machine is kept idle at a time when it could begin processing some operation. Although nondelay schedules can be expected to provide very good solutions, there is no guarantee that they provide the optimum solutions. A job-shop problem is completely solved if the starting times of all operations are determined, and the precedence relationships between the operations are not violated. In general, the optimality criteria for machine scheduling can be classified into various groups. These measures of performance include criteria based on completion-dates (i.e. the time at which the last job-operation is completed), flow-times (i.e. the amount of time the job spends in the shop), and makespan (i.e. the total elapsed time required to process all of a given set of jobs). In many cases, previous researchers have used minimization of makespan as the objective function. The optimum solution to the 2-job 3-machine problem is shown in Fig. 1(b).

3. PROBLEM FORMULATION

Several integer linear programming formulations of the machine sequencing problem have been proposed in the past, but no attempt has been made to identify and exploit any special structure of this integer linear programming technique. Recently, the authors[4] proposed a mixed integer linear approach to solving job-shop scheduling. In this approach, the determination of an optimal job-shop schedule can be formulated as a linear programming with integer adjustments to minimize the starting times of all jobs, subject to a set of precedence constraints and restrictive assumptions.

Let S_{ik} denote the starting time of job i on machine k, and t_{ijk} the processing time for operation (i, j, k). Assuming operation $(i, j-1, h)$ precedes (i, j, k), then the inequalities representing precedence constraints in order for a set of S_{ik} to be feasible are:

$$S_{ik} - S_{ih} \geq t_{i,j-1,h} \quad 1 \leq j \leq m, \quad 1 \leq i \leq n, \qquad (1)$$

where m and n are the number of machines and the number of jobs, respectively. The condition that all starting times must be positive results in the constraint:

$$S_{ik} \geq 0 \quad 1 \leq i \leq n. \qquad (2)$$

It is also necessary to ensure that no two operations are processed simultaneously by the same machine at the same time. For example, if job i precedes job p on machine k [i.e. operation (p, q, k) starts after the completion of (i, j, k)], then

$$S_{pk} - S_{ik} \geq t_{ijk}.$$

On the other hand, if job p precedes job i on machine k, it is also necessary that

$$S_{ik} - S_{pk} \geq t_{pqk}.$$

Using a "zero–one" variable y_{ipk} to specify the operation sequence, i.e. $y_{ipk}=1$ if job i precedes job p on machine k, and $y_{ipk}=0$ otherwise, the above constraints become:

$$S_{pk}-S_{ik}+H^*(1-y_{ipk})\geq t_{ijk} \tag{3}$$

$$S_{ik}-S_{pk}+H^*y_{ipk}\geq t_{pqk} \tag{4}$$

where constant H represents an arbitrary positive number greater than the maximum value of all processing times t_{ijk}s such that the constraints (3) and (4) are satisfied.

Therefore, the entire job-shop problem formulation can be summarized as minimizing the cost function

$$\sum_{i=1}^{n} S_{ik_i}$$

subject to

$$\begin{aligned}
&S_{ik}-S_{ih}\geq t_{i,j-1,h} && 1\leq j\leq m,\quad 1\leq i\leq n\\
&S_{pk}-S_{ik}+H^*(1-y_{ipk})\geq t_{ijk} && i\geq 1, p\leq n, 1\leq k\leq m\\
&S_{ik}-S_{pk}+H^*y_{ipk}\geq t_{pqk} && i\geq 1, p\leq n, 1\leq k\leq m\\
&S_{ik}\geq 0 && 1\leq i\leq n
\end{aligned}$$

where k_i is the machine which the last operation of job i is assigned. There are mn constraints of type (1) or type (2), and $mn(n-1)$ constraints of type (3) or type (4), giving a total number of mn^2 constraints. There are also mn number of S_{ik}s and $mn(n-1)/2$ number of y_{ipk}s, resulting a total number of $mn(n+1)/2$ variables. For a 2-job 3-machine problem, there are a total of 12 inequalities with 9 variables in the formulation, i.e.

$$\begin{aligned}
&S_{11}\geq 0\\
&S_{12}-S_{11}\geq t_{111}\\
&S_{13}-S_{12}\geq t_{122}\\
&S_{21}-S_{23}\geq t_{213}\\
&S_{22}-S_{21}\geq t_{221}\\
&S_{23}\geq 0\\
&S_{21}-S_{11}+H^*(1-y_{121})\geq t_{111}\\
&S_{11}-S_{21}+H^*y_{121}\geq t_{221}\\
&S_{22}-S_{12}+H^*(1-y_{122})\geq t_{122}\\
&S_{12}-S_{22}+H^*y_{122}t_{232}\\
&S_{23}-S_{13}+H^*(1-y_{123})\geq t_{133}\\
&S_{13}-S_{23}+H^*y_{123}\geq t_{213}.
\end{aligned}$$

4. NETWORK ARCHITECTURE

The analog Tank and Hopfield linear programming network[2] seeks to minimize a cost function

$$F(\mathbf{A},\mathbf{V})=\mathbf{A}\cdot\mathbf{V}$$

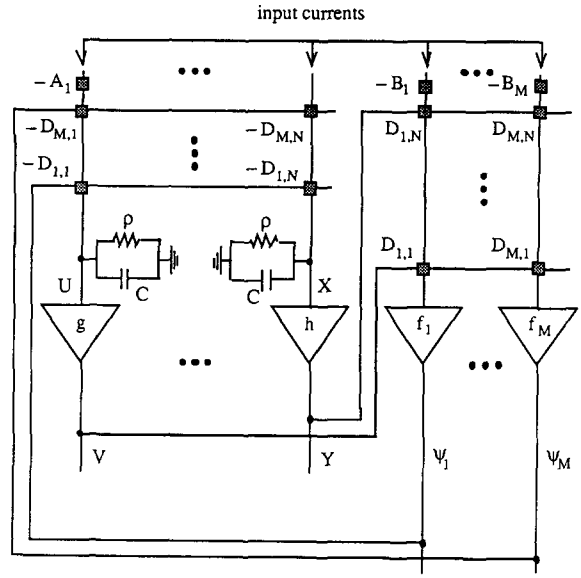

Fig. 2. The modified Tank and Hopfield network consisting of linear and nonlinear processors for solving mixed-integer linear programming.

where $\mathbf{A}$ is row vector array of N coefficients for N variables which are components of column vector $\mathbf{V}$. This linear cost function is subject to a set of M linear constraints among the N variables:

$$\mathbf{D}_j\cdot\mathbf{V}\geq\mathbf{B}_j \qquad j=1,\ldots,M$$

where matrix $\mathbf{D}_j$ contains the N variable coefficients of constraint equation j and column vector $\mathbf{B}_j$ are the bounds.

The proposed network described in this paper for solving mixed integer-linear programming problems is based on a slightly modified Tank and Hopfield network, as shown in Fig. 2. The difference is the addition of nonlinear h-amplifiers. The output voltage V_i of linear g-amplifier represents the starting time of each operation after minimization, while output Y_i of nonlinear h-amplifier represents the zero–one variable. The components of $\mathbf{A}$ are proportional to the input currents fed into the g- and h-amplifiers. Each g- and h-amplifier has an input time constant $\rho_i C_i$ which acts as a memory element. The g-amplifier circuit can be implemented as a noninverting amplifier which obeys a linear relationship, as shown in Fig. 3. The gain of the g-

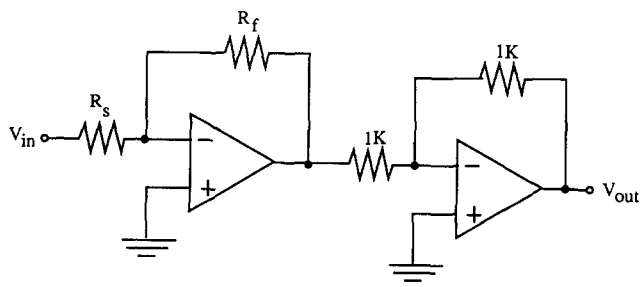

Fig. 3. A noninverting amplifier circuit for implementing the linear g-amplifier.

amplifier is adjusted by varying R_s or R_f. The h-amplifier is simply a high-gain analog comparator, as shown in Fig. 4. The neutral position of the step function is adjusted by V_{ref}. Description of other analog components such as variable resistors, and sigmoid amplifier circuits for implementing neural networks can be found in Ref. 5.

The M outputs (Ψ_j) of the f-amplifers represent constraint satisfaction. This set of amplifiers have a nonlinear transfer function

$$\psi_j = f(U_j) = \begin{cases} 0 & U_j \geq 0 \\ -U_j & U_j < 0 \end{cases}$$

where

$$\mathbf{U}_j = \mathbf{D}_j \cdot \mathbf{V} - \mathbf{B}_j,$$

i.e. if $U_j < 0$ the output Ψ_j of the f-amplifier will be a large positive value, indicating a violation of the jth constraint. When this happens, a current of value ($\Psi_j D_{ji}$) will be fed back into the inputs of the g- and h-amplifiers, causing the outputs V_i to oscillate between +Vcc and −Vcc, where Vcc is the power supply voltage of the operational amplifiers. If $U_j \geq 0$, the output Ψ_j of the f-amplifier is zero, corresponding to a valid solution and no current will be fed back to the g- and h-amplifiers. As a result, the output voltages of the g- and h- amplifiers will be stable and directly proportional to the input bias currents A_is.

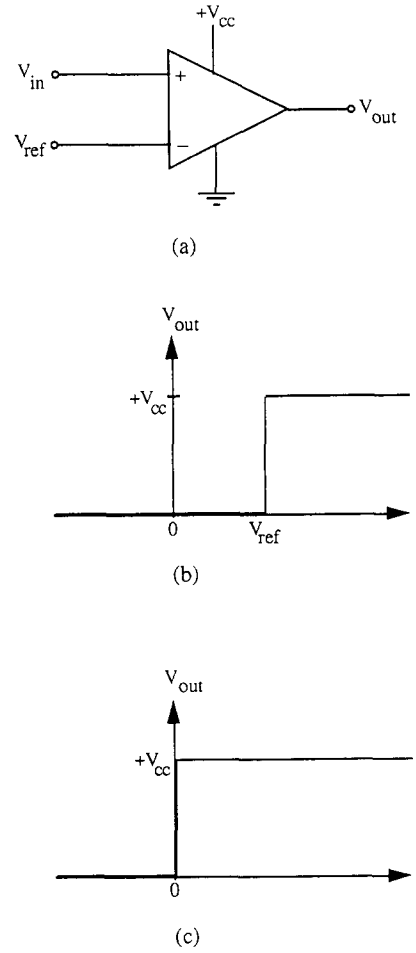

Fig. 4. (a) An analog comparator for implementing an h-amplifier. (b) The adjusting step function. (c) The neutral position with $V_{ref} = 0$.

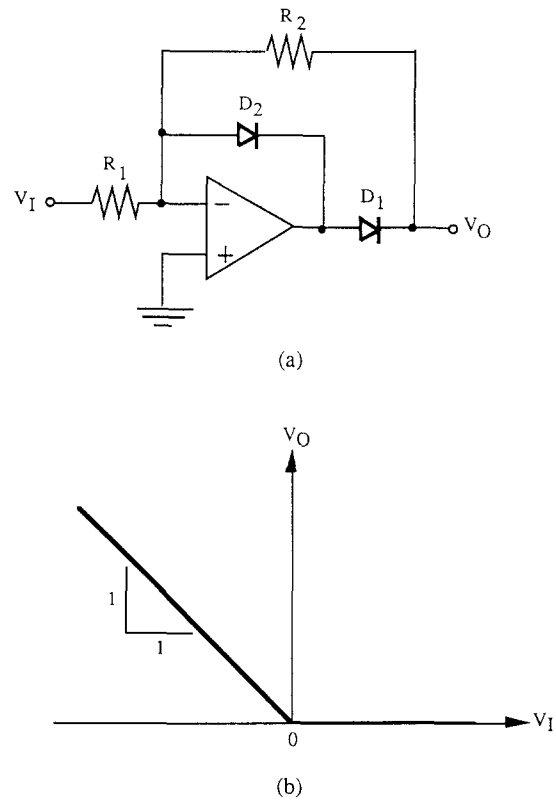

Fig. 5. (a) A precision inverting half-wave rectifier implementing the f-amplifier circuit. The diode D_2 keeps the feedback path closed around the op-amp during the times when the rectifier diode D_1 is in cut-off, thus preventing the op-amp from saturating, which causes slower slew-rate. (b) The transfer function of the circuit for $R_1 = R_2$.

The nonlinear f-amplifier can be implemented using a precision inverting half-wave rectifier, as shown in Fig. 5. The rectifier circuit operates in the following manner: For positive V_I diode D_2 conducts and closes the negative feedback loop around the op-amp. A virtual ground appears at the inverting input terminal and the op-amp output is clamped at −0.7 V. This negative voltage keeps D_1 in cut-off, which means no current flows through feedback resistance R_2. Hence the rectifier output voltage is zero. When V_I goes negative, the voltage at the op-amp output terminal becomes positive, causing D_2 to be reverse-biased. Since the inverting input terminal of the op-amp always appears as virtual ground, diode D_1 is forward-biased and thus conducts through R_2 and establishes a negative feedback loop around the op-amp. The current through the feedback resistance R_2 is equal to the current through input resistance R_1. Therefore, if $R_1 = R_2$, then the output voltage of the rectifier is:

$$V_O = V_I \qquad \text{for } V_I \leq 0.$$

The complete transfer function of the precision half-wave rectifier is shown in Fig. 5(b). Notice that the

slope of the characteristic can be set to any desired value by selecting the appropriate values for R_1 and R_2.

Since the currents generated by the g- and h-amplifiers are summed at the inputs of the f-amplifiers, it is necessary that the time constants $(\rho_i C_i)$ at the inputs of g- and h-amplifiers be matched. If the response time of the f-amplifier is assumed to be negligible compared to the g- and h-amplifiers, then the dynamics of the g-amplifier can be described by:

$$C_i \frac{dU_i}{dt} = -A_i - \frac{U_i}{R_i} - \sum_j^M D_{ji} f(D_j \cdot B_j) \tag{5}$$

where $M = mn^2$ (i.e. total number of constraints). Similarly, the differential equation describing the behavior of the h-amplifier is:

$$C_i \frac{dX_i}{dt} = -A_i - \frac{X_i}{R_i} - \sum_j^N D_{ji} f(D_j \cdot Y - B_j). \tag{6}$$

Consider an energy function for the entire circuit of the form:

$$E = \sum_i^P A_i V_i + \sum_i^P \frac{1}{R_i} \int_0^{V_i} g^{-1}(V)\, dV + \sum_i^P \sum_j^M \int f(z)\, dz + \sum_i^Q \frac{1}{R_i} \int_0^{Y_i} h^{-1}(Y)\, dY + \sum_i^Q \sum_j^M \int f(w)\, dw$$

where

$U = g^{-1}(V)$

$X = h^{-1}(Y)$

$z = D_j \cdot V - B_j$

$w = D_j \cdot Y - B_j$

$P = mn$ (i.e. total number of S_{ik}s)

$Q = mn(n-1)/2$ (i.e. total number of y_{ipk}s).

It can be easily shown that E is always a decreasing energy function which seeks out a minimum in E and stops. The proof is as follows:

$$\begin{aligned}
\frac{dE}{dt} &= \sum_i^P A_i \frac{dV_i}{dt} + \sum_i^P \frac{1}{R_i} U_i \frac{dV_i}{dt} + \sum_i^P \sum_j^M f(z) \frac{dz}{dt} \\
&\quad + \sum_i^Q \frac{1}{R_i} X_i \frac{dY_i}{dt} + \sum_i^Q \sum_j^M f(w) \frac{dw}{dt} \\
&= \sum_i^P A_i \frac{dV_i}{dt} + \sum_i^P \frac{U_i}{R_i} \frac{dV_i}{dt} + \sum_i^P \sum_j^M f(D_j \cdot V - B_j) D_{ji} \\
&\quad \times \frac{dV_i}{dt} + \sum_i^Q \frac{X_i}{R_i} \frac{dY_i}{dt} + \sum_i^Q \sum_j^M f(D_j \cdot Y - B_j) D_{ji} \frac{dY_i}{dt} \\
&= \sum_i^P \frac{dV_i}{dt} \left\{ -C_i \frac{dU_i}{dt} \right\} + \sum_i^Q \frac{dY_i}{dt} \left\{ -C_i \frac{dX_i}{dt} \right\} \\
&= -\sum_i^P C_i g^{-1'}(V_i) \left(\frac{dV_i}{dt} \right)^2 - \sum_i^Q C_i h^{-1'}(Y_i) \left(\frac{dY_i}{dt} \right)^2.
\end{aligned}$$

Since C_i is positive, and $g^{-1'}(V_i)$ and $h^{1'}(Y_i)$ are monotone-increasing and step functions, respectively, it follows that $dE/dt \leq 0$ with $dE/dt = 0$ if $dV_i/dt = dY_i/dt = 0$ for all i. In other words, the time evolution of the network is a motion in state space which seeks out a minimum in E and stops.

5. NETWORK SIMULATIONS

To illustrate how the modified Tank and Hopfield network works, consider the formulation of an example 2-job 3-machine problem shown in Fig. 1(a). Let x_1–x_6 uniquely identify the starting times of the operations, while x_y–x_9 represent the zero–one variables, and V_i represents the output voltage of the g- and h-amplifiers corresponding to variable x_i. Choosing constant $H = 10$ (i.e. any number greater than the maximum processing time value of 9), the complete set of inequalities is:

$$\begin{aligned}
&x_1 \geq 0 \\
&-x_1 + x_2 \geq 5 \\
&-x_2 + x_3 \geq 8 \\
&x_4 - x_6 \geq 7 \\
&-x_4 + x_5 \geq 3 \\
&x_6 \geq 0 \\
&-x_1 + x_4 - 10x_7 \geq -5 \\
&x_1 - x_4 + 10x_7 \geq 3 \\
&-x_2 + x_5 - 10x_8 \geq -2 \\
&x_2 - x_5 + 10x_8 \geq 9 \\
&-x_3 + x_6 - 10x_9 \geq -8 \\
&x_3 - x_6 + 10x_9 \geq 7.
\end{aligned}$$

Then the corresponding $\mathbf{D}_j$ matrix is:

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
-1 & 0 & 0 & 1 & 0 & 0 & -10 & 0 & 0 \\
1 & 0 & 0 & -1 & 0 & 0 & 10 & 0 & 0 \\
0 & -1 & 0 & 0 & 1 & 0 & 0 & -10 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & 0 & 10 & 0 \\
0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & -10 \\
0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 10
\end{bmatrix}$$

Using 1 kΩ as base unit, the resistor network for implementing $\mathbf{D}_j$ becomes:

$$\begin{bmatrix}
1K & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
1K & 1K & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & 1K & 1K & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & 1K & \infty & 1K & \infty & \infty & \infty \\
\infty & \infty & \infty & 1K & 1K & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & 1K & \infty & \infty & \infty \\
1K & \infty & \infty & 1K & \infty & \infty & 0.1K & \infty & \infty \\
1K & \infty & \infty & 1K & \infty & \infty & 0.1K & \infty & \infty \\
\infty & 1K & \infty & \infty & 1K & \infty & \infty & 0.1K & \infty \\
\infty & 1K & \infty & \infty & 1K & \infty & \infty & 0.1K & \infty \\
\infty & \infty & 1K & \infty & \infty & 1K & \infty & \infty & 0.1K \\
\infty & \infty & 1K & \infty & \infty & 1K & \infty & \infty & 0.1K
\end{bmatrix}$$

where " ∞ " denotes "no connection". If equal weight is placed on all production costs such as operation costs, machine idle costs, etc., then the input bias currents $-A_i$s can be arbitrarily set to -1 mA. If an external potential of -1 V is applied to provide the $-A_i$s, then the set of input resistances is [1 kΩ, 1 kΩ, . . . , 1 kΩ].

Let each unit of processing time be represented by 1 mA of current. Then the current vector representing the bounds of constraint equations is $-\mathbf{B}_j = [0, -5\text{ mA}, -8\text{ mA}, -7\text{ mA}, -3\text{ mA}, 0, 5\text{ mA}, -3\text{ mA}, 2\text{ mA}, -9\text{ mA}, 8\text{ mA}, -7\text{ mA}]$. Assuming the set of resistor values for implementing $-\mathbf{B}_j$ is [100 Ω, 100 Ω, . . . , 100 Ω], then the input potentials to the set of resistors must be [0, −500 mV, −800 mV, −700 mV, −300 mV, 0, +500 mV, −300 mV, +200 mV, −900 mV, +800 mV, −700 mV]. The complete circuit for solving the 2-job 3-machine problem is shown in Fig. 6.

When the network reaches stable states, the output voltages of the g- and h-amplifiers are measured. The results are (in mV):

$$V_1 = 0.0,\quad V_2 = 0.5,\quad V_3 = 1.3,\quad V_4 = 0.7,\quad V_5 = 1.3,$$
$$V_6 = 0.0,\quad Y_1 = 1.0,\quad Y_2 = 1.0,\quad Y_3 = 0.0.$$

where V_is are outputs of the linear g-amplifiers representing the starting times, and Y_is are outputs of the h-amplifiers representing the zero–one variables. By normalization, the starting times for the operations are:

$$S_{11} = 0,\quad S_{12} = 5,\quad S_{13} = 13,\quad S_{21} = 7,\quad S_{22} = 13,\quad S_{23} = 0.$$

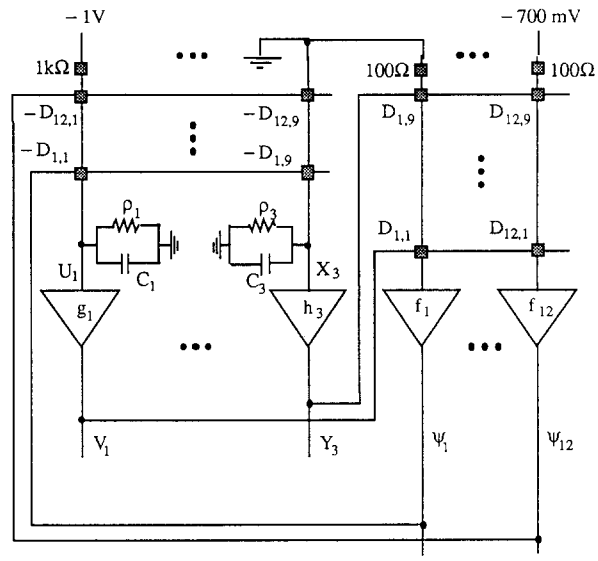

Fig. 6. Circuit diagram of the network for solving a 2-job 3-machine problem.

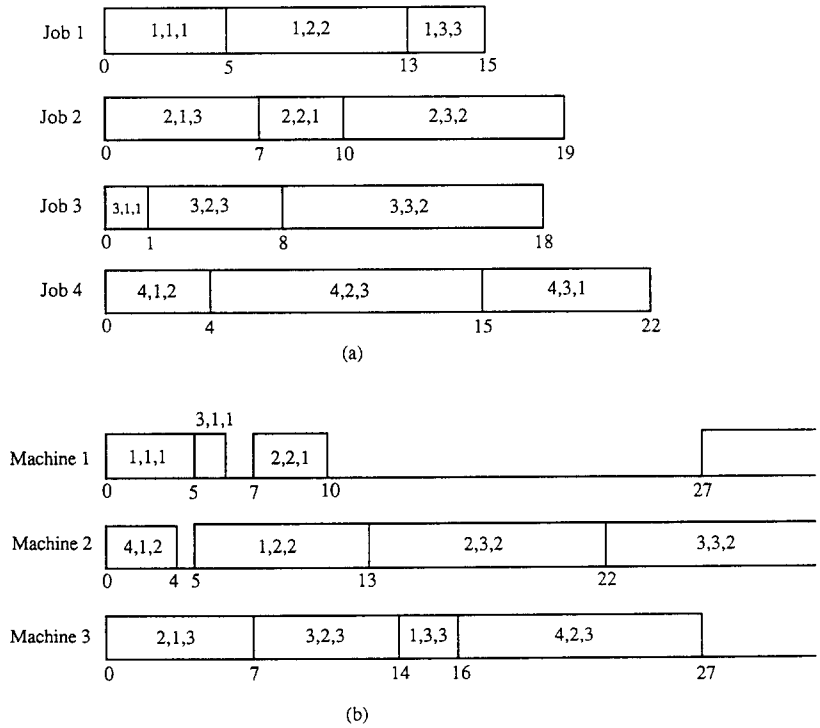

Fig. 7. (a) An example 4-job 3-machine job-shop problem. (b) A near-optimum schedule as produced by the simulator.

Figure 1(b) shows the Gantt chart of the job-shop schedule constructed from the above results, which turns out to be an optimum schedule. Another simulation run was performed based on a 4-job 3-machine problem [Fig. 7(a)]. as reported in an earlier work.[4] There are $N=30$ variables and $M=48$ constraints in the formulation of a 4-job 3-machine problem. The result turned out to be a near-optimum schedule as shown in Fig. 7(b). One reason for not finding the optimum solution is probably due to the fact that the highly convoluted energy surface has many local minima.

6. DISCUSSIONS

It has been shown how a modified Tank and Hopfield network can be utilized to solve difficult optimization problems such as the classical job-shop scheduling. Zhou *et al.*[1] also presented a modified Tank and Hopfield analog computational network which exhibits linear scaling property, and thus appears to be better than the original approach by Foo and Takefuji[4]. However, a closer look reveals that the former approach requires extensive computations by each neural processor. Thus, there is less network complexity, at the expense of more-complex processing by each neuron. This contradicts the popular concept of neurons with simple activation functions. The approach proposed here may require more neurons and interconnects, but each neuron has a very simple activation function. Thus, the network is self-contained and does not need extensive calculations as required in the approach by Zhou *et al.*

The software simulator written to simulate the behaviour of the network uses the 4th-order Runge–Kutta integration technique to solve the differential equations (5) and (6). So far, only small problems have been simulated. Results shows that as the problem size increases, there is no guarantee of finding an optimum solution using this deterministic analog computational network, but the network will always provide very good, valid solutions.

Acknowledgements—The authors would like to thank Lisa R. Anderson and Joe Cordaro for implementing the linear programming circuit which was later modified to solve the job-shop scheduling.

REFERENCES

1. Zhou D. N., Cherkassky V., Baldwin T. R. and Olson D. E. A neural network approach to job-shop scheduling. *IEEE Trans. Neural Networks* **2**, 175–179 (1991).
2. Tank D. W. and Hopfield J. J. Simple neural optimization networks: an A/D converter, signal decision circuit, and a linear programming circuit. *IEEE Trans. Circuits Syst.* **CAS-33**, 533–541 (1986).
3. Takefuji Y. *Neural Network Parallel Computing*. Kluwer, Amsterdam (1992).
4. Foo S. Y. and Takefuji Y. Integer-linear programming neural networks for job-shop scheduling. *Proceedings of IEEE IJCNN '88*, San Diego, pp. 341–348 (1988).
5. Foo S. Y., Anderson L. R. and Takefuji Y. Analog components for the VLSI of neural networks. *IEEE Circuits Devices* **6**, No. 4, 18–26 (1990).

AUTHORS' BIOGRAPHIES

Simon Y. Foo received his B.S., M.S., and Ph.D. in Electrical Engineering from the University of South Carolina in 1983, 1984, and 1988, respectively. He is currently an Assistant Professor at the Department of Electrical Engineering, Florida State University, Florida. His research interests include neural networks, design and test of analog and digital integrated circuits, and nanometer microelectronics. Dr Foo is a member of Eta Kappa Nu and the IEEE.

Yoshiyasu Takefuji joined the Faculty of Environmental Information at Keio University in April 1992, having previously worked at Case Western Reserve University, and the Universities of South Florida and South Carolina. He received his B.S. (1978), M.S. (1980), and Ph.D. (1983) in Electrical Engineering from Keio University. His research interests focus on neural network parallel computing for solving real-world problems, VLSI applications and silicon architecture. He authored "Neural Network Parallel Computing", and coauthored four other books. He was an Editor of the *Journal of Neural Network Computing*, and has published more than 100 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*.

Harold Szu is currently a senior scientist with Naval Surface Warfare Center, Maryland. He is also the current President of the International Neural Network Society (INNS). He has been teaching many short courses in neural networks.