'92 KACC 1992. 10. 19～21

Petri-Net Modeled Neural Networks to Obtain a Near Optimal Jop-Shop Schedule

Modified Hofield Type Neural Network to Solve Jop-Shop Scheduling Problem

Boo Hee Nam, * Seok Ho Chang

Dept. of Control and Instrumentation Eng., * Dept. of Electronic Eng.
Kangwon Nat'l Univ.

ABSTRACT

This paper presents a modified model of the Hopfield's neural network to solve linear programming problem. Simulation results show that our model gives more exact and faster answer for solving the linear programming problem. We applied it to flexible manufacturing system case, in which it is necessary to solve LP problem, i.e. jop-shop scheduling.

1. Introduction

There are a number of methods to solve the linear programming problem. Depending on constraints such as computation costs and problem size, an appropriate technique for solving jop-shop schduling can be selected. In general, a branch and bound search algorithm is practical for small problems, while for large problems, heuristic, stochastic, and mathematical approaches are suggested. Among those, several recent studies [1], [2], [3] have been made on the scaling property of neural network. For the first time Hopfield and Tank [1] presented the fact that linear programming neural network(LPNN) has the characteristic of scaling to solve LP problem. However, several [4] criticized Hopfield nets used for combinatorial optimization, because of inability to find global minimum and poor scaling properties.

Foo and Takefuji [2] employed the TSP-type Hopfield model and integer programming neural networks to solve jop-shop scheduling. But later Kennedy and Chua [3] corrected the error in Hopfield model.

In this paper we adopted the results corrected by Chua, and added a new property to neural networks for the purpose of better scaling performance.

2.Linear Programming Formulation

Generally the process of flexible manufacturing system is described by petri-nets. And jop-shop scheduling in FMS has been also planned by petri-nets control method [5]. Here we consider how the information represented by Petri nets can be converted to the LP problem formulation. Fig. 1 shows an example modeled by petri nets. The directed arcs between ordinary place, p_{ij}, and transition, t_{ij}, inform the relations within each distinct jop(i.e. inter-operation conditions), while the directed arcs between machine place, p_{mi}, and transition tell the relationships of inter-jop condition.

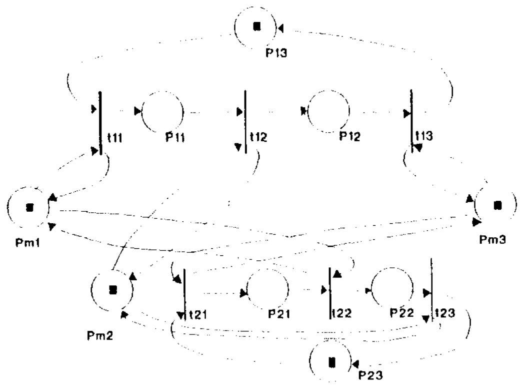


Fig. 1. Modelling of a jop shop system by Petri nets in the case of 2 jobs / 3 machines problem.

The mathematical formulation for solving the jop-shop scheduling problem is described as follows. Let S_{ik} denote the starting time, t_{ik} the processing time for operation k of job i and ki the last opeatation of each job.

$$\text{Min} \sum_{i=1}^{n} S_{ik_i} \qquad (2.1)$$

subject to the constranints

i)first operation condition:

$$S_{i1} \geq 0$$

ii)inter-operation condition:

$S_{i1} - S_{i2} \geq t_{i2}$, if operation (i,2) precedes (i,1) iii)inter-jop condition:

$$S_{ik} - S_{jp} + H(1-y_m) \geq t_{jp}$$
$$S_{jp} - S_{ik} + Hy_m \geq t_{ik}, \quad 1 \leq m \geq M$$
$$y_m = 0 \text{ or } 1, \qquad (2.2)$$

where M is the number of resouces(i.e machines) and H represents an arbitrary positive number twice the maximum processing time. In [2], Foo and Takefuji used H value that is greater than the maximum value among all processing times. But we recognized Foo and Takefuji chosen H value doesn't satisfy all the constraints.

After a number of simulations, we discovered the convergence of neural network was disturbed by zero-one variables y_m. Coexistence of linear and zero-one variables forces the neural network for solving the LP problem not to get a good scaling property. Hence inter-jop condition that avoids the conflict when more than two operations try to use the same machine may require some change. Inter-jop conditions are dependent upon inter-operation conditions. Therefore, only one of two inter-jop conditions works and the other operates like as a null constraint. New representation which eliminates the zero-one variables is as follows:

Inter-jop condition:

$$S_{ik} - S_{jp} \geq t_{jp} \qquad S_{ik} \geq S_{jp}$$
$$S_{jp} - S_{ik} \geq t_{ik} \qquad S_{jp} > S_{ik} \qquad (2.3)$$

3. LP Neural Network for Solving Jop-Shop Scheduling

A. Anaysis of the LPNN

A pure linear programming problem can be defined as the attempt to minimize a cost function

$$\pi = \vec{A} \cdot \vec{V}$$

where $\vec{A}$ is an N-dimentional vector of coefficients for N variables. This minimization is subject to a set of K linear constraints among variables:

$$\vec{D_j} \cdot \vec{V} \geq B_j, \ j = 1, \ldots, K$$

where the $\vec{D_j}$ contains N variable coefficients in a constraint equation and the B_j's are the bounds.

Typical motion equation describing the Hopfield's LPNN is [1]:

$$C_i \frac{du_i}{dt} = -A_i - \frac{u_i}{R} - \sum_j D_{ji} \, f(\vec{D_j}\vec{V} - B_j) \qquad (3.1)$$

where ui, R and C_i are input voltage, constant input resistance, and input capacitance, respectively.

The third term on the right side forces the solution vector to go into the feasible solution space.

If $\vec{D_j} \cdot \vec{V} - B_j < 0$, f amplifier operates as a penalty function which moves the solution vector to the direction of decreasing the distance from the feasible solution area.

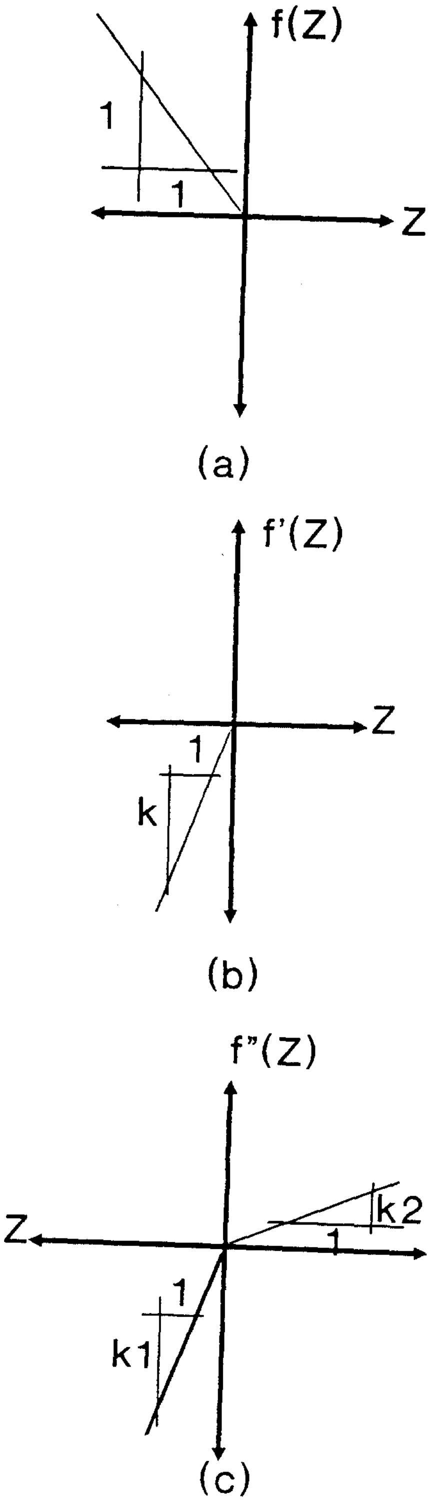


Fig. 2 Penalty Function
(a) Tank and Hopfield's nonlinearity, $f(Z)$
(b) The nonlinearity of Chua and Lin, $f'(Z)$
(c) Our proposed nonlinearity, $f''(Z)$

Fig. 2 shows three kinds of penalty function. Fig. 2(a) was corrected to Fig. 2(b) by Chua[2]. *f* function scales down the solution vector so as to enter feasible area, but it can not serve for searching optimum point. In (3.1) it is the first term on the right side to function as seeking for optimal solution. We will call the phenomenon A-effect. However, a fixed quantity, bias, affects always the solution vector regardless of the precision of currunt state. In other words there is no adaptation to accuracy in the feasible area, blinded LPNN merely goes down gradually. This method requires long time to reach a optimal point, and the solution obtained will not be expected to have good precise value because of A-effct. Thus we eliminated A-effect and proposed the function in fig. 3(c) instead of fig. 3(b).

$$C_i \frac{du_i}{dt} = - \frac{u_i}{R} - \sum_j D_{ji} \, f(\bar{D}_j \bar{V} - B_j) \qquad (3.2)$$

The penalty function of Fig. 2(c) offers new adapatation performance in the feasible area. For example, consider the following simple problem.

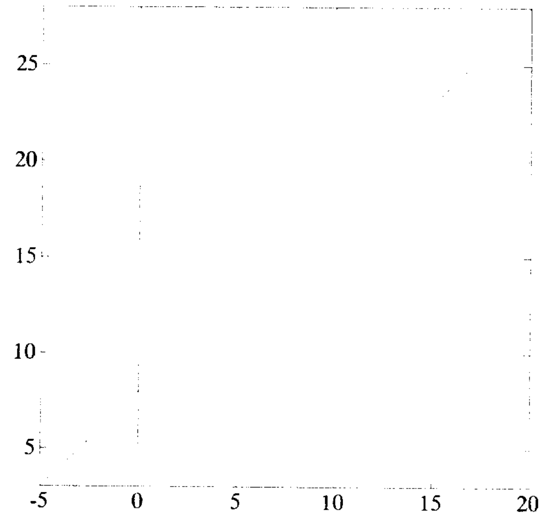
$$\mathrm{Min} \sum_i v_i$$

subject to

$$v1 \geq 0$$

$$v2 - v1 \geq 8$$

Through the simulation we obtained good result. The comparision with Chua model in the same circumstances is shown in Fig. 3(a) and 3(b). Faster and more precise result than conventional LPNN was acquired.



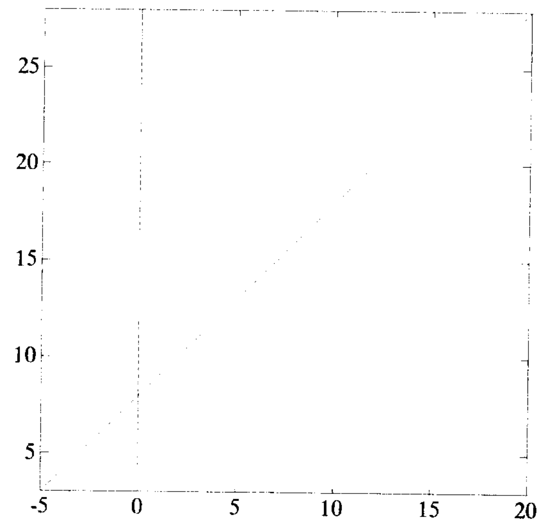


Fig. 3. (a) Chua model, (b) Adaptable LPNN

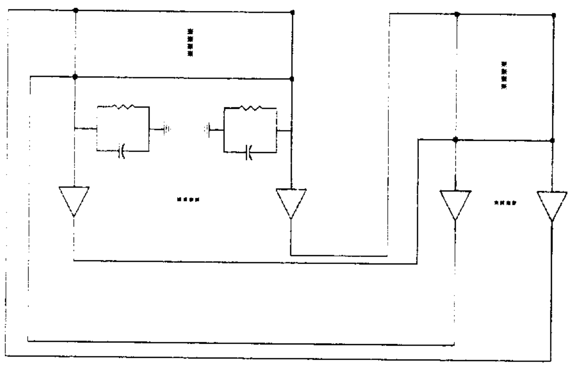


Fig. 4. LP Neural Network Diagram

B. Application to Jop-Shop Scheduling.

We applied the proposed method to jop-shop scheduling, 2 jobs and 3 machines allocation problem. Table 1 shows the routing of the processes.

Table 1. Jop-Shop scheduling routing.
The value in perenthesis is processing time.

Jop	operation		
	1	2	3
1	1(5)	2(8)	3(2)
2	3(7)	1(3)	2(2)

At this time corresponding constraint set is:

$v1 \geq 0$

$v2 - v1 \geq t1$

$v3 - v2 \geq t2$
$v4 \geq 0$
$v5 - v4 \geq t4$
$v6 - v5 \geq t5$
$v1 - v5 \geq t5$ or $v5 - v1 \geq t1$
$v2 - v6 \geq t6$ or $v6 - v2 \geq t2$
$v3 - v4 \geq t4$ or $v4 - v3 \geq t3.$

However, the solution vector of this example can't satisty all the equalities of the constraints. At optimum some constraints reach their boundray walls, on the other hand the rest settle down in surplus state, in terms of the distance from the constraint wall.

Therefore in order to use adaptaional scaling property, we inserted surplus variables that transform inequality constraints to equality. New dynamic equation is:

$$C_i \frac{du_i}{dt} = -\frac{u_i}{R} - \sum_j D_{ji}\, f(\bar{D}_j\bar{V} - B_j - S_j) \qquad (3.3)$$

At this time corresponding constraint set is converted to followings:

$v1 - s1 = 0$
$v2 - v1 - s2 = t1$
$v3 - v2 - s3 = t2$
$v4 - s4 = 0$
$v5 - v4 - s5 = t4$
$v6 - v5 - s6 = t5$
$v1 - v5 - s7 = t5$ or $v5 - v1 - s7 = t1$
$v2 - v6 - s8 = t6$ or $v6 - v2 - s8 = t2$
$v3 - v4 - s9 = t4$ or $v4 - v3 - s9 = t3$

The change rule as to $\vec{S}$ is described as (3.3). The flag $\vec{M}$, indicating how closely the solution vector reach the optimum is the margin of surplus. The overall operation algorithm is (3.3) to (3.5).

$s_j = D_j v(k-1) - B_j$ (3.3)

$M_j = D_j(v(k-1) - v(k))$ (3.4)

if $M_j > 0$, $f(M)=K_1M_j$

else $f(M_j)=0$ (3.5)

Fig. 5 shows the obtained solution vector, optimal solution.

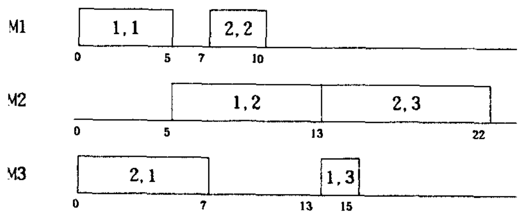


Fig. 5. Solution represented by Gantt chart.

4. Conclusions

We developed a modifield Hopfield model to solve the linear programming problem for jop-shop scheduling. The proposed algorithm seems to yield better results tnan the existing ones.

The more rigorous and mathematical analysis on this model is necessaary in further research.

References

[1] D. W. Tank and J. J. Hopfield, "Simple Neural Optimization Networks: An A/D Converter, Signal Decision Circuits, and a Linear Programming Circuit", IEEE Trans. on Circuits and Systems, vol CAS-33, May 1986.

[2] Y. P. S. Foo and Y. Takefuji, "Integer Linear Programming Neural Networks for Jop-Shop Scheduling", IJCNN 88, San Diego, CA, July 1988.

[3] M. P. Kennedy and L. O. Chua, "Unifying the Tank and Hopfield Linear Programming Circuit and Nonlinear Programming Circuit of Chua and Lin", IEEE Trans. on Circuits and Systems, vol. CAS-34, No.2, pp. 210-214, 1987.

[4] DARPA Neural Network Study, AFCEA International Press, 1988.

[5] Myung-Geun Chun and Zeung-Nam Bien, " A Hierarchical and Decentralized Modeling and Scheduling of Jop Shop Using Timed Petri Nets", The Transanctions of the Korean Institute of Electronic Engineers, vol. 28, pp. 470 - 478, July 1991.

[6] David G. Luenberger, Linear and Nonlinear Programming, Addison Wesley, 1984.

[7] James A. Freeman and David M. Skapura, Neural Networks, Addison Wesley, 1991.

[8] Andrew Kusiak, Intelligent Manufacturing Systems, Prentice-Hall Inc, 1990.