Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

A Coupled Transiently Chaotic Neural Network Approach for Scheduling Identical Parallel Machines with Sequence Dependent Setup Times

Aiqing Yu*. Xingsheng Gu**
Bin Jiao***

** Research Institute of Automation, East China University of Science and Technology Shanghai, China (e-mail: yuaiqing@ mail.ecust.edu.cn)*
*** Research Institute of Automation, East China University of Science and Technology Shanghai, China (e-mail: xsgu@ ecust.edu.cn)*
**** Shanghai Dianji University, Shanghai, China (e-mail: binjiaocn@ 163.com)*

Abstract: Identical parallel machine scheduling problems with sequence dependent setup times, to minimize the total completion time are studied in this paper. A mixed-integer programming formulation of this problem is presented. And a neural computation architecture based on a Coupled Transiently Chaotic Neural Network is introduced to construct the model. The transiently chaotic dynamics are defined after the energy function is constructed by a penalty function approach. Tradeoff problem existing among the penalty terms included in the energy function is overcome by using time-varying penalty parameters. Simulation results tested on different problems with 100 random initial conditions show that this approach converges to near-optimal or optimal solutions and outperforms the Hopfield neural networks.

1. INTRODUCTION

The study of parallel machine problems is relevant from both the theoretical and the practical points of view. From the practical point of view, it is important because we can find many examples of the use of parallel machines in the real world. From the theoretical point of view, it is a generalization of the single machine problem and a particular case of problems arising in flexible manufacturing systems. At present, the vast majority of papers are concentrated on identical parallel machine scheduling problem (IPMSP) (Allahverdi, A. et al., 2006). IPMSP is one of the typical NP-hard combinatorial optimization problems, which is described as follows: scheduling N jobs on M identical machines in parallel to optimize the objectives such as makespan, the total completion time, the sum of weighted tardiness and so on (Mokotoff, E., 2001). The problem considered here is to schedule jobs on identical parallel machines with sequence dependent setup times to minimize the total completion time ($P \mid ST_{sd} \mid \sum C_i$).

Over the last decade, Neural Networks (NNs) have been proposed in the existing literature as an approach to solve a wide variety of combinatorial optimization problems. Successful applications of NNs to various classification problems have caused growing research interest in neural networks. In particular, Hopfield neural networks have provided acceptable solutions to optimization problems such as linear programming and TSP (Hopfield, J.J. and Tank, D.W., 1985). However, not much progress has been made for the exploration of the use of NNs in solving multi-machines scheduling problems, especially the parallel machine scheduling. One of the difficulties is that the objective function can not be expressed in neural network energy function. Another important one is that there is no theoretically established method for choosing the values of the penalty coefficients. Attempts to resolve these difficulties have involved new networks. To the best of our knowledge, Akyol and Bayhan should be the first to propose a Hopfield type dynamical gradient neural network for solving the identical parallel machine scheduling problem(Akyol, D.E. and Bayhan, G., 2006). The majority of the existing studies are based on Hopfield network or its extensions. The difficulty encountered with optimizing networks based on the Hopfield-Tank model is their tendency to settle into local minima. Chen and Aihara proposed a transiently chaotic neural network (TCNN) as an approximation method for TSP, by introducing transiently chaotic dynamics into neural networks (Chen, L. and Aihara, K., 1995). Unlike conventional neural networks only with point attractors, TCNN has richer and more flexible dynamics, so that it can be expected to have higher ability of searching for globally optimal or near-optimal solutions.

Motivated by the recent developments, a Coupled Transiently Chaotic Neural Network (CTCNN) is presented as an innovative, alternative approach for solving IPMSP in this paper. This study is to apply a new neural network energy function in solving this problem. In order to design an analog neural network for a specified optimization problem, we construct a suitable computational energy function whose minimization leads to a system of differential equations sometimes called equations of motion. The formulation of the equations of motions is the central issue in the design of an optimizing neural network. In this paper we concentrate on the architecture of CTCNN model for IPMSP to minimize the total completion time. The major distinction of our approach is that it utilizes CTCNN to tackle with this problem including sequence dependent setup times and the objective function is denoted by penalty function.

The remainder of this paper is organized as follows. In the next section, a mixed-integer programming model for identical parallel machine scheduling problem is given. In Section 3, a neural computation architecture based on CTCNN is introduced to describe this problem and a penalty function approach is employed to construct a new energy function of the network. An improved approach for adjusting the penalty parameters is proposed to overcome the tradeoff problem. In Section 4, the computational results tested on different scale problems of 100 different initial conditions are analyzed to investigate efficiency of the model. Finally, we give some conclusions and the future research directions.

2. PROBLEM FORMULATION

A mixed-integer programming formulation for the $P \mid ST_{sd} \mid \sum C_i$ problem is presented in this section.

2.1 Assumptions

Let N be the number of jobs to be scheduled at time zero and M be the number of identical machines in parallel where $N \geq M$. We assume that machines are initially setup for nominal job 0 and must finish setup for a tear down job $N+1$. These jobs are the place-holder jobs, which indicate the initial machine setup and required ending state. Other assumptions are described as follows.

(i) Each job has only one operation and can be processed on any machine but it must be processed without preemption.

(ii) Each machine can process at most one job at a time.

(iii) All jobs are available for machine processing simultaneously at time zero.

(iv) Each job requires processing time units and setup time before processed.

2.2 Notations

For convenience, the notations and decision variables that are used in the remainder of this paper are defined as follows.

$p_j =$ the processing time of job j on identical machines, $j = 1, \cdots, N$

$s_{ij} =$ the sequence-dependent setup time between job i and job j , $i, j \in \{1, \cdots, N\}$

$X_{ij} = 1$ if job i is scheduled immediately before job j , else $X_{ij} = 0$, $i = 0, 1, \cdots, N,\ \ j = 1, 2, \cdots, N, N+1$

$C_j =$ the completion time of job j , $j = 1, \cdots, N$

2.3 Mixed-Inter Programming Model

The IPMSP with sequence dependent setup times (denoted by Π) can be formulated as a mixed-integer program as follows.

$$(\Pi)\ \text{Minimize } \sum_{j=1}^{N} C_j$$

$$\text{s.t.} \quad \sum_{j=1}^{N} X_{0j} = M \qquad (1)$$

$$\sum_{j=1}^{N+1} X_{ij} = 1, \qquad i = 1, \cdots, N \qquad (2)$$

$$\sum_{i=0}^{N} X_{ij} = 1, \qquad j = 1, \cdots, N \qquad (3)$$

$$C_0 = 0 \qquad (4)$$

$$C_j \geq p_j + \sum_{i=0}^{N} X_{ij}(s_{ij} + C_i), \qquad j = 1, \cdots, N \qquad (5)$$

$$C_j \geq 0 \qquad j = 0, 1, \cdots, N, \qquad (6)$$

$$X_{ij} \in \{0,1\} \quad i = 0, 1, \cdots, N,\ \ j = 1, 2, \cdots, N, N+1 \qquad (7)$$

$$X_{ij} = 0 \qquad i = j \qquad (8)$$

The objective function seeks to minimize the total completion time over all jobs. Constraint (1) ensures that exactly M machines can be scheduled. Constraint sets (2) and (3) ensure that each job is scheduled on one and only one machine. Constraint (4) defines the completion time of the non-existent job 0 so that the recursive constraint sets (5) can be used to find completion times of the jobs. Constraint sets (5) also ensure that jobs only begin processing after both the previous job and the required setup have been completed. Constraint sets (6), (7) and (8) provide limits on the decision variables. This model has $(N+1)^2$ zero-one variables and N real-valued variables.

3. A CTCNN MODEL FOR IPMSP

In this section, a neural computation architecture based on CTCNN model is introduced to define its corresponding quadratic energy function. An improved approach for adjusting the penalty parameters is proposed to overcome the tradeoff problems.

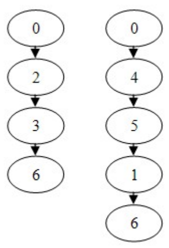
3.1 The Energy Function for IPMSP

Solving an optimization problem with constraints satisfaction requires selecting an appropriate representation of the problem, and choosing the appropriate weights for the connections and input biases. In our approach, the familiar matrix representation of neurons for solving the TSP is used. A solution of IPMSP is represented by a matrix of neurons with $N+1$ rows and $N+1$ columns, where N is the number of jobs. Assume VX_{ij} to be the neuron output which represents whether job i precedes job j on the same machine or not. Figure 1 shows a feasible solution to an example of identical parallel machine scheduling with 2 machines and 5 jobs and its transformation process from the matrix output (solution) to the Gantt chart. Job 0 and Job 6 represent the start and the end of all processing jobs respectively. The solution in Figure 1 is represented by a set of cost function trees (Fig. 1(b)) encoded in the matrix (Fig. 1 (a)). Each node in the set of trees represents a job and each link represents the interdependency between jobs. The total completion time of all jobs can be computed by traversing the paths from Job 0 to Job 6.

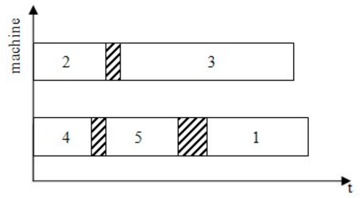
In the presented CTCNN, we adopt two types of neurons: one type of neurons to represent real valued variables C_j and the other type to represent binary valued variables X_{ij}. The inputs and outputs to these two types neurons are denoted by UC_i, VC_i and UX_{ij}, VX_{ij}, respectively.

VX_{ij}	1	2	3	4	5	6
0	0	1	0	1	0	0
1	0	0	0	0	0	1
2	0	0	1	0	0	0
3	0	0	0	0	0	1
4	0	0	0	0	1	0
5	1	0	0	0	0	0

(a) matrix representation



(b) cost function trees



(c) Gantt chart

Fig. 1. An example 5-job 2-machine problem

A neural network energy function for combinatorial optimization is usually of the form:

$$E = \mathrm{A} \cdot (\text{cost}) + \sum \mathrm{B}_i \cdot (\text{violation of constraint } i) \tag{9}$$

in which penalty weights A and $\mathrm{B}_i > 0$, and cost is the objective function value that is to be optimized and independent of constraint violations. By minimizing the energy function E, we attempt to minimize the original objective function, while simultaneously maximizing the satisfaction of the constraints. The successful use of such an energy function requires an appropriate selection of values for parameters A and B_i.

According to the assumptions and notations presented in Section 2, the energy function in this case will include five constraint functions. Here, the first term E_1 adds a positive penalty if the solution does not satisfy the initial setup constraints (1). In accordance with this constraint, E_1 will take the following form.

$$E_1 = (\sum_{j=1}^{N} VX_{0j} - M)^2 \tag{10}$$

The second term E_2 and the third one E_3 add a positive penalty if the solution does not satisfy the row constraints (2) and column constraints (3). They can be defined as

$$E_2 = \sum_{i=1}^{N} (\sum_{j=1}^{N+1} VX_{ij} - 1)^2 \tag{11}$$

$$E_3 = \sum_{j=1}^{N} (\sum_{i=0}^{N} VX_{ij} - 1)^2 \tag{12}$$

Considering the inequality constraints (5) of the completion time, penalty function is adopted to formulate the fourth term E_4. It may be written as

$$E_4 = \sum_{j=1}^{N} \nu(p_j + \sum_{i=0}^{N} VX_{ij}(s_{ij} + VC_i) - VC_j) \tag{13}$$

where ν represents the penalty function (Watta, P.B. and Hassoun, M.H., 1996).

$$\nu(\xi) = \begin{cases} \xi^2 & \xi > 0 \\ 0 & \xi \le 0 \end{cases} \tag{14}$$

Noticed that $VX_{ij} \in \{0,1\}$. These binary constraints will be captured by the fifth term and it may be written as follows.

$$E_5 = \sum_{i=0}^{N} \sum_{j=1}^{N+1} VX_{ij}(1 - VX_{ij}) \tag{15}$$

Correspondingly, the global energy function for this network consisting of the objective function $\sum C_j$ and these constraints of IPMSP can be defined as:

$$E_{total} = \mathrm{A} \cdot \sum_{j=1}^{N} VC_j + \sum_{i=1}^{5} \mathrm{B}_i \cdot E_i \tag{16}$$

3.2 The Dynamics

By setting each connection weight the same as the Hopfield neural network, the equations describing the network dynamics of CTCNN for the IPMSP are obtained as follows.

$$\frac{\partial E_{total}}{\partial VC_j} = \mathrm{A} + \mathrm{B}_4 \cdot \left[\begin{array}{c} (-1) \cdot \nu'(p_j + \sum_{i=0}^{N} VX_{ij}(s_{ij} + VC_i) - VC_j) \\ + \sum_{i=1}^{N} VX_{ji} \cdot \nu'(p_i + \sum_{l=0}^{N} VX_{li}(s_{li} + VC_l) - VC_i) \end{array} \right] \tag{17}$$

$$\begin{aligned} \frac{\partial E_{total}}{\partial VX_{ij}} &= \mathrm{B}_1 \cdot 2 \cdot (\sum_{l=1}^{N} VX_{0l} - M) \cdot \delta_{i0} \cdot (1\text{-}\delta_{j(N+1)}) \\ &+ \mathrm{B}_2 \cdot 2 \cdot (\sum_{l=1}^{N+1} VX_{il} - 1) \cdot (1\text{-}\delta_{i0}) + \mathrm{B}_3 \cdot 2 \cdot (\sum_{l=0}^{N} VX_{lj} - 1) \cdot (1\text{-}\delta_{j(N+1)}) \\ &+ \mathrm{B}_4 \cdot (s_{ij} + VC_i) \cdot \nu'(p_j + \sum_{l=0}^{N} VX_{lj}(s_{lj} + VC_l) - VC_j) \cdot (1\text{-}\delta_{j(N+1)}) \\ &+ \mathrm{B}_5 \cdot (1 - 2 \cdot VX_{ij}) \end{aligned} \tag{18}$$

$$\text{where } \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \ne j \end{cases} \tag{19}$$

and ν' is the derivative of the penalty function ν.

$$\nu'(\xi) = \begin{cases} 2\xi & \xi > 0 \\ 0 & \xi \le 0 \end{cases} \tag{20}$$

Since the computation is performed in all neurons at the same time, the CTCNN operates in a fully parallel mode, which consists of two sub networks, that is real valued neural networks and binary valued neural networks.

The states of real valued neurons and binary valued neurons are updated as follows.

$$UC_j(t+1) = kUC_j(t) + \alpha(-\frac{\partial E_{total}}{\partial VC_j}) - z(t)(VC_j(t) - I_0) \tag{21}$$

$$UX_{ij}(t+1) = kUX_{ij}(t) + \alpha(-\frac{\partial E_{total}}{\partial VX_{ij}}) - z(t)(VX_{ij}(t) - I_0) \tag{22}$$

$$z(t+1) = (1-\beta)z(t) \tag{23}$$

where the variable $z(t)$ is assumed to be common to all neurons.

Neuron outputs are calculated by $V = f(U)$, where $f(\cdot)$ is the activation function, U is the input and V is the output of the neuron. Each type of neurons has its own activation function. Real valued neurons' activation functions is a piecewise linear function (see Equation (24)). Binary valued neurons outputs are calculated by a log-sigmoid function.

$$VC_j = \begin{cases} UC_j & UC_j \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

$$VX_{ij}(t) = \frac{1}{1+e^{-UX_{ij}(t)/\varepsilon}} \tag{25}$$

3.3 *Time-Varying Penalty Coefficients*

The energy function for this network is constructed by using the well known "penalty methods," which convert a constrained problem to an unconstrained one by having penalty terms on constraint violations. The unconstrained problem is then solved by the neural networks as mentioned above. To obtain a solution without having coefficients tend to infinity, a tradeoff between solution optimality and constraint satisfaction has to be made through the fine turning of algorithm parameter.

The general form of energy function includes different penalty parameters for each constraint. The choice of initial values of penalty parameters is a problem, since they usually have a major effect on the overall optimization accuracy and efficiency. If these parameters are chosen to be too small, the obtained final result may be inexact or even absolutely incorrect, because of the major constraint violations. On the other hand, large values of the penalty parameters ensure a near satisfaction of all constraints, but may create a computationally and very poorly-conditioned (strongly nonlinear) energy function, and, consequently, a set of stiff differential equations, if the initial point is far away from the desired solution. Because there is no theoretically established method for choosing the values of the penalty coefficients for an arbitrary optimization problem, the appropriate values for these coefficients can be determined by empirically running simulations and observing the optimality and/or feasibility of the resulting equilibrium points of the system (Watta, P.B. and Hassoun, M.H., 1996). Recently, time based penalty parameters are proposed to overcome the tradeoff. In order to determining the appropriate values of the penalty parameters, Wang used monotonically time-varying penalty parameters for solving convex programming problems (Wang, J., 1991). Akyol and Bayhan proposed to use time varying penalty parameters increased in a linear fashion in a stepwise manner to reduce the feasible region (Akyol, D.E. and Bayhan, G., 2006). In this paper, we make penalty parameters time variables, starting with small values and continuously increasing them when their corresponding constraints are not satisfied during the optimization process. Firstly, we try to satisfy the inequality constraints by penalizing them and run the simulations without considering any other constrains. Secondly, row and column constraints besides binary constraints are taken into consideration and corresponding penalty parameter will be adjusted until all of them are satisfied. Finally, the penalty parameter of the objective function is set to 1 and corresponding penalty parameter will be adjusted until all the constraints are satisfied.

3.4 *Pseudo-Code of CTCNN Algorithm*

In the following, the pseudo-code of CTCNN algorithm is described in detail.

Step 1 Generate the initial neuron states randomly.

Step 2 Determinate network parameters α, β, k, ε, I_0, $z(0)$.

Step 3 Set B1=B2=B3=B5=0 and B4=1 (the coefficient of the inequality constraint). If the constraint associated with parameter B4 is satisfied, go to Step 4, otherwise increase the value of B4 and then go to Step 6.

Step 4 Select parameters B2, B3 and B5 (higher values than B4) and use the predetermined value of B4 to check whether both of the constraints associated with these parameters are satisfied. If yes, go to Step 5, otherwise increase the value of parameter whose associated constraint is not satisfied and then go to Step 6.

Step 5 Set A=1, select parameter B1 (a higher value than B2 to increase the effect of the initial setup constraint) and use the predetermined values of B2, B3 and B4 to check whether all the constraints associated with these parameters are satisfied. If yes, go to Step 6, otherwise increase the value of parameter whose associated constraint is not satisfied and then go to Step 6.

Step 6 Update all the neurons using equations (21) to (25) and repeat a number of times. If A=B1=B2=B3=B5=0, go to Step 3. If A=B1=0, go to Step 4. If A=1, increase the value of parameter whose associated constraint is not satisfied.

Step 7 If the end condition is not satisfied, go to Step 6, otherwise stop the evolution and check the feasibility and optimality of the final solution.

4. SIMULATION ANALYSIS

In this section, the CTCNN approach has been applied to identical parallel machine scheduling problems. The program for the IPMSP was coded in Matlab software and implemented on a personal computer equipped with Intel Pentium IV 3.07GHz microprocessor and 512M RAM. To evaluate the performance of CTCNN approach, computational experiments were performed on randomly generated test problems. For the experiments, different size problems were generated. The processing time p_j of job j was generated from the uniform distribution $[1,10]$. After the processing time were generated, the setup time s_{ij} were set to $a \times \min(p_i, p_j)$, where a is a coefficient randomly generated in $[A,B]$, where $[A,B] \in \{[0.01,0.1],[0.1,0.2],[0.2,0.5]\}$ (Nessah, R. et al., 2007). Considering the solution quality depending highly on initial conditions, all the solutions were obtained by simulations with 100 random initial conditions. Problems with 10, 20, 30 jobs were generated with 2 and 3 machines.

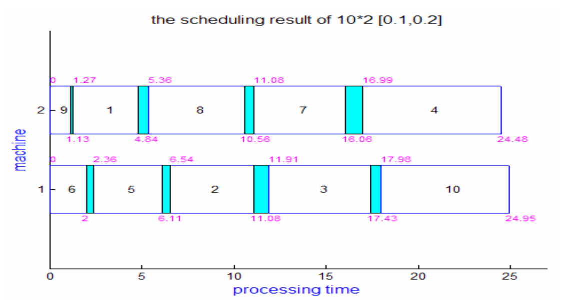
CTCNN parameters for different size problems were determined by trial and error as shown in Table 1. Parameter α affects the energy function to generate transient chaos. Parameter β governs the bifurcation speed of the transient chaos.

Table 1 CTCNN parameters for different size problems

n	k	α	β	I_0	ε	$z(0)$
10	0.997	0.008	0.001	0.65	0.008	0.08
20	0.997	0.010	0.001	0.65	0.008	0.08
30	0.998	0.015	0.0008	0.65	0.008	0.1

In Appendix A, the results were compared with those of Hopfield-like dynamic neural network (HDNN) proposed in (Akyol, D.E. and Bayhan, G., 2006) in terms of the best result, average of results, the worst one and Percent Feasibility of the Solutions (PFS). The results show that along with the increase of the test problem size, the percent feasibility of solutions decreased and the network were trapped into local minima more easily. Compared to HDNN, CTCNN model can find better solutions; sometimes even the average of results outperformed the best one obtained by HDNN, which has been denoted in bold.

We had to rely upon time varying penalty parameters, while there is no systematic guidance available as to what values of the parameters ought to be. However, in the experiments, an appropriate parameter set of that could lead to a good solution was obtained within a few runs, each of which used random initial state. It was also noticed that when values were too large, the network could not achieve the steady state. Figure 2 shows the Gantt charts of the best solutions for 6 test problems whose $[A,B]=[0.1,0.2]$.



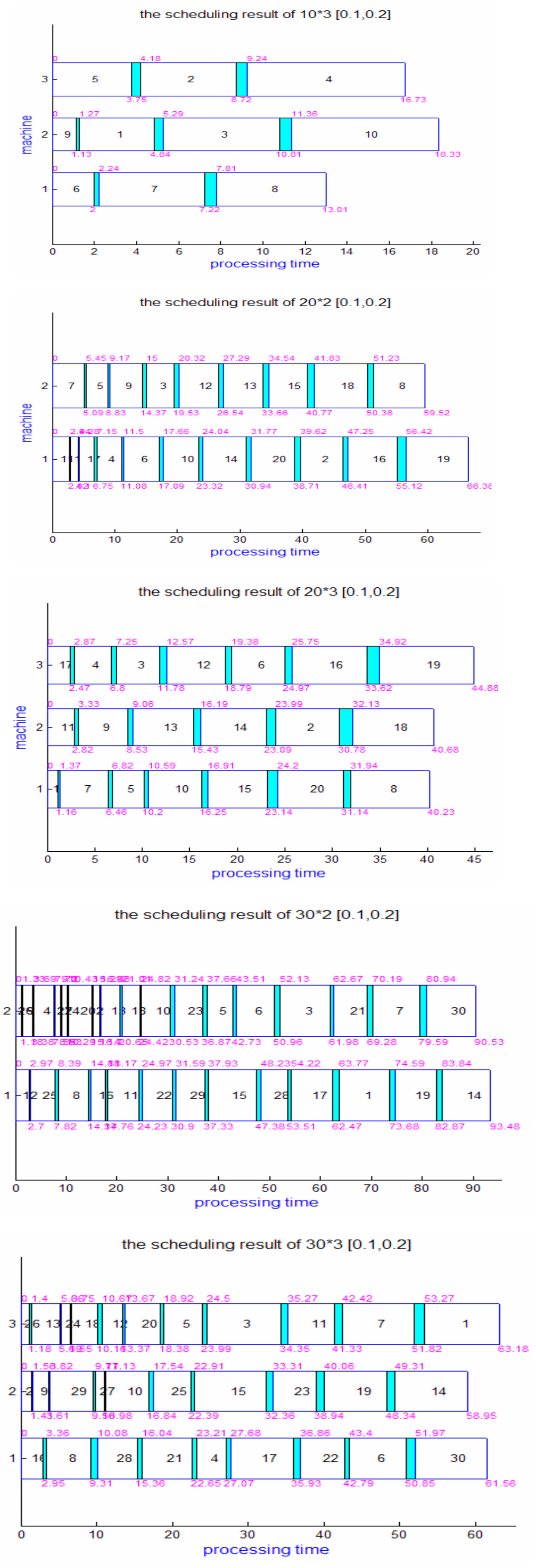


Fig. 2. Gantt charts of the best solutions for 6 problems

5. CONCLUSIONS

A Coupled Transiently Chaotic Neural Network computation scheme that employs time-varying penalty coefficients have been introduced to solve identical parallel machines scheduling problems with sequence dependent setup times. Simulation experiments show that tradeoff problem existing among the penalty terms included in the energy function is overcome by using time-varying penalty parameters and the proposed network generates feasible solutions. After running simulations with time evolving penalty coefficients, near optimal and optimal solutions can be found in reasonable and finite time.

In this paper, we just construct the network model for the basic identical parallel machines scheduling problem. Other models with ready time, job splitting etc. can be further research directions. Besides, more effective architecture of the neural network, selection of penalty parameters and construction of energy function also may be research topics.

ACKNOWLEDGEMENT

This work is supported by National Natural Science Foundation of China (Grant No. 60674075 and No. 60774078), the Key Technologies Program of Shanghai Educational Committee (Grant No. 05ZZ73) and Shanghai Leading Academic Discipline Project (Project No. 13504).

REFERENCES

Akyol, D.E. and Bayhan, G. (2006). Minimizing Makespan on Identical Parallel Machines Using Neural Networks. In: Proc. ICONIP, Part III, 553-562.

Allahverdi, A. et al. (2006). A survey of scheduling problems with setup times or costs. European Journal of Operational Research. doi:10.1016/j.ejor.2006.06.060.

Chen, L. and Aihara, K. (1995). Chaotic Simulated Annealing by A Neural Network Model with Transient Chaos. Neural Networks, 8, 915-930.

Hopfield, J.J. and Tank, D.W. (1985). Neural Computation of Decisions in Optimization Problems, Biological Cybernetics, 55, 141-152.

Mokotoff, E. (2001). Parallel Machine Scheduling Problems: A Survey. Asia-Pacific Journal of Operational Research, 18, 193-242.

Nessah, R. et al. (2007). An exact method for $Pm / sds, r_i / \sum_{i=1}^{n} C_i$ problem. Computers & Operations Research, 34, 2840-2848.

Watta, P.B. and Hassoun, M.H. (1996). A Coupled Gradient Network Approach for Static and Temporal Mixed-Integer Optimization. IEEE Transactions on Neural Networks, 7, 78-593.

Wang, J. (1991). A Time-Varying Recurrent Neural System for Convex Programming. In: Proceedings of IJCNN-91-Seattle International Joint Conference on Neural Networks, 147-152.

Appendix A. Computation results for test problems

			CTCNN		HDNN	
n	m	[A,B]	Best/Avg./Worst	PFS(%)	Best /Avg. /Worst	PFS(%)
10	2	[0.01,0.1]	87.27/**88.33**/89.77	100	**88.42**/89.56/90.01	96
		[0. 1,0.2]	118.64/**118.89**/119.36	100	**119.01**/119.73/120.47	93
		[0. 2,0.5]	147.35/**148.78**/150.45	100	**149.06**/149.51/151.13	92
	3	[0.01,0.1]	63.19/65.60/67.01	100	64.55/67.02/68.45	91
		[0. 1,0.2]	86.54/87.79/88.60	100	87.24/89.92/91.99	95
		[0. 2,0.5]	107.05/108.48/109.18	100	107.62/109.34/110.47	90
20	2	[0.01,0.1]	668.3/688.87/695.5	100	678.91/694.37/706.51	88
		[0. 1,0.2]	561.41/568.59/572.69	98	564.05/570.98/576.76	85
		[0. 2,0.5]	668.01/679.85/688.55	97	676.72/682.14/692.19	86
	3	[0.01,0.1]	462.92/473.15/484.85	98	472.2/482.56/492.44	82
		[0. 1,0.2]	393.22/407.11/412.18	96	400.96/413.39/422.22	85
		[0. 2,0.5]	473.04/484.04/498.53	95	475.26/488.73/493.67	87
30	2	[0.01,0.1]	1201.7/1248.9/1279.3	92	1220.5/1236.4/1283.2	83
		[0. 1,0.2]	1119.2/**1130.5**/1137.1	100	**1130.6**/1138.2/1210.1	77
		[0. 2,0.5]	1413/**1432.3**/1445.8	93	**1434.8**/1437.2/1451.3	84
	3	[0.01,0.1]	848.24/881.23/931.22	89	856.4/883.09/934.55	76
		[0. 1,0.2]	781.35/805.8/823.07	95	790.49/813.54/828.92	75
		[0. 2,0.5]	985.99/**1001.1**/1103.8	88	**1006**/1015.4/1043.2	72