A Coupled Transiently Chaotic Neural Network Approach for Identical Parallel Machine Scheduling

YU Ai-Qing[1] GU Xing-Sheng[1]

Abstract Scheduling jobs on identical machines is a situation frequently encountered in various manufacturing systems. In this paper, a new coupled transiently chaotic neural network (CTCNN) is put forward to solve identical parallel machine scheduling. A mixed integer programming model of this problem is transformed into a CTCNN computation architecture by introducing a permutation matrix expression. A new computational energy function is proposed to express the objective besides all the constraints. In particular, the tradeoff problem existing among the penalty terms in the energy function is overcome by using time-varying penalty parameters. Finally, results tested on 3 different scale problems with 100 random initial conditions show that the network converges and can solve these problems in the reasonable time.

Key words Scheduling, identical parallel machines, coupled transiently chaotic neural network, time-varying penalty coefficients

Identical parallel machine scheduling problem (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 and sum of weighted tardiness. This paper considers the problem of scheduling jobs on identical parallel machines, without preemption, to minimize makespan, which was shown to be NP-hard by a reduction to a bipartitioning problem[1].

Over the last decade, neural networks (NNs) have been proposed as an approach to solve a wide variety of combinatorial optimization problems. Successful applications of NNs to various classification problems have caused a growing research interest in neural networks. In particular, Hopfield neural networks have provided acceptable solutions to optimization problems such as A/D conversion, linear programming, and the traveling salesman problem (TSP)[2−3]. However, little progress has been made for the exploration of the use of NNs in solving multimachine scheduling problems, especially the parallel machine scheduling. One of the difficulties is that the objective function cannot 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 been made by researchers. To the best of our knowledge, Akyol and Bayhan[4] were the first to propose a Hopfield type dynamical gradient neural network for solving the identical parallel machine scheduling problem. 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[5] propose a transiently chaotic neural network (TCNN) by introducing transiently chaotic dynamics into neural networks. 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. In this paper, a coupled transiently chaotic neural network is proposed as an innovative, alternative approach to solve identical parallel machine scheduling problem to minimize the makespan.

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 2, a new coupled transiently chaotic neural network is put forward and is applied to IPMSP. Coupled transiently chaotic neural network (CTCNN) consists of two coupled sub-networks, which are real valued neural network and binary valued neural network. A computation architecture based on CTCNN is introduced and a penalty function approach is used to construct a new energy function of the network. An improved approach for adjusting the penalty parameters is proposed to overcome the tradeoff problem. The computational results tested on 3 different scale problems of 100 different initial conditions are analyzed to verify efficiency of the proposed approach in Section 3. Finally, we give some conclusions and the future research directions in Section 4.

1 Problem formulation

A mixed-integer programming model for this problem ($P||C_{\max}$) is presented in this section. First, we will describe assumptions and notations for IPMSP and then present the formulation with a discussion of the constraints.

1.1 Assumptions

1) Each job is available at time zero and can be processed on any machine;

2) Identical machines in parallel are continuously available and are never kept idle while work is waiting;

3) Each machine can process at most one job at a time;

4) Once processing begins on a job, it continues to completion without interruption.

1.2 Notations

$i = 1, 2, \cdots, m$: the machines;

$j = 1, 2, \cdots, n$: the jobs to be scheduled;

p_j : processing time of job j;

$x_{ij} = 1$ if job j is scheduled on machine i, else $x_{ij} = 0$;

$C_{\max}$: makespan, the maximum completion time of all jobs.

1.3 Mixed-integer programming model for IPMSP

The basic IPMSP (denoted by Π) can be formulated as a mixed-integer program as follows:

$$\Pi: \quad \min(C_{\max})$$

$$\text{s.t.} \qquad C_{\max} - \sum_{j=1}^{n} p_j x_{ij} \geq 0 \qquad i = 1, \cdots, m \tag{1}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad j = 1, \cdots, n \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} \geq 1 \qquad i = 1, \cdots, m \tag{3}$$

Received April 27, 2007; in revised form December 28, 2007
Supported by National Natural Science Foundation of China (60674075, 60774078)
1. Research Institute of Automation, East China University of Science and Technology, Shanghai 200237, P. R. China
DOI: 10.3724/SP.J.1004.2008.00697

$$x_{ij} \in \{0,1\} \quad i = 1, \cdots, m \quad j = 1, \cdots, n \tag{4}$$

Constraint (1) defines the maximum completion time of all jobs. Constraint (2) ensures that each job is scheduled on one and only one machine. Constraint (3) ensures that exactly m machines can be scheduled. Constraint sets (4) provide limits on the decision variables x_{ij}. This model has nm zero-one variables and one real-valued variable.

2 Coupled transiently chaotic neural network and its application in IPMSP

A coupled transiently chaotic neural network is put forward and is applied to IPMSP in this section. A new energy function is proposed for IPMSP, which includes the objective and all the constraints. An improved approach for adjusting the penalty parameters is presented to overcome the tradeoff problems encountered in using penalty function approach to construct the energy function.

2.1 CTCNN model 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 m rows and n columns, where n and m are the number of jobs and the number of identical machines in parallel, respectively. Table 1 shows a feasible solution to an example of identical parallel machine scheduling with 2 machines and 5 jobs. The permutation matrix in Table 1 represents that jobs 1 and 4 are assigned to machine 1 and jobs 2, 3, and 5 are assigned to machine 2.

Table 1 The permutation matrix

x_{ij}	Job 1	Job 2	Job 3	Job 4	Job 5
Machine 1	1	0	0	1	0
Machine 2	0	1	1	0	1

In the presented CTCNN, we use two types of neurons: a continuous type neuron to represent real valued variables $C_{\max}$ and discrete type of neurons to represent binary valued variables x_{ij}. The input and output to these two type neurons are denoted by $UC_{\max}$, Ux_{ij}, and $VC_{\max}$, Vx_{ij}, respectively. According to constraint (1), we can figure out that the objective $C_{\max}$ is relevant to x_{ij}, which means neurons $VC_{\max}$ are coupled with neuron Vx_{ij}. Therefore, a coupled transiently chaotic neural network is proposed by updating two types of neurons in different ways.

All the neurons are updated by the following equations (5) $\sim$ (9), which are partly based on TCNN presented in [5]. Neuron outputs are calculated by $V = f(U)$, where $f(\cdot)$ is the activation function. In this paper, there are two types of neurons and each type has its own activation function.

$$UC_{\max}(t+1) = kUC_{\max}(t) + \alpha(-\frac{\partial E}{\partial VC_{\max}}) - z(t)(VC_{\max}(t) - I_0) \tag{5}$$

$$Ux_{ij}(t+1) = kUx_{ij}(t) + \alpha\left(-\frac{\partial E}{\partial Vx_{ij}}\right) - z(t)\left(Vx_{ij}(t) - I_0\right) \tag{6}$$

$$z(t+1) = (1-\beta)\, z(t) \tag{7}$$

$$VC_{\max} = \begin{cases} UC_{\max}, & UC_{\max} \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

$$Vx_{ij}(t) = \frac{1}{1 + \mathrm{e}^{-Ux_{ij}(t)/\varepsilon}} \tag{9}$$

where

E = energy function defined in Section 2.2;

α = positive scaling parameter for inputs;

k = damping factor of nerve membrane ($0 \leq k \leq 1$);

ε = steepness parameter of the output function ($\varepsilon > 0$);

$z(t)$ = self-feedback connection weight or refractory strength $z(t) \geq 0$;

β = damping factor of the time-dependent $z_i(t)$ ($0 \leq \beta \leq 1$);

I_0 = positive parameter.

2.2 Energy function and the dynamics of CTCNN for IPMSP

To design an analog neural network for IPMSP, 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 motion is the central issue in the design of an optimizing neural network.

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

$$E = A \cdot (\cos t) + \sum B_i \cdot (\text{violation of constraint } i) \tag{10}$$

where penalty weights A and $B_i > 0$, cost is the objective function value that is to be optimized and is independent of constraint violations. By minimizing the energy function E, we attempt to minimize the cost, whereas 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 mixed-integer programming model presented in Section 1, there are three constraints to be encoded in CTCNN architecture.

Here, the first term E_1 adds a positive penalty if the solution does not satisfy constraint (1).

$$E_1 = \sum_{i=1}^{m} v(\sum_{j=1}^{n} p_j Vx_{ij} - VC_{\max}) \tag{11}$$

In accordance with constraints (2) and (3), E_2 and E_3 will take the following forms.

$$E_2 = \sum_{j=1}^{n} \sum_{i=1}^{m} \sum_{k=1, k\neq i}^{m} Vx_{ij} Vx_{kj} \tag{12}$$

$$E_3 = (\sum_{i=1}^{m} \sum_{j=1}^{n} Vx_{ij} - n)^2 \tag{13}$$

where v represents the penalty function[6].

$$v(\xi) = \begin{cases} \xi^2, & \xi > 0 \\ 0, & \xi \leq 0 \end{cases} \tag{14}$$

Equation (11) is zero if and only if constraint (1) is satisfied, which ensures that the objective $C_{\max}$ is at least the

completion time of each machine. Equation (12) is zero if and only if each column in the matrix of neurons does not contain more than one neuron turned on ($Vx_{ij} = 1$), the rest of the neurons in that particular column being turned off ($Vx_{ij} = 0$). (13) is zero if and only if there are n neurons being turned on in the whole matrix of neurons.

The global energy function for this network consisting of the objective function $C_{\max}$ and these constraints of IPMSP can be defined as

$$E = A \cdot VC_{\max} + B_1 \cdot E_1 + B_2 \cdot E_2 + B_3 \cdot E_3 \tag{15}$$

where A, B_1, B_2, and B_3 are positive penalty parameters.

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

$$\frac{\partial E}{\partial VC_{\max}} = A + B_1 \cdot (-1) \cdot \sum_{i=1}^{m} v'(\sum_{j=1}^{n} p_j Vx_{ij} - VC_{\max}) \tag{16}$$

$$\begin{aligned} \frac{\partial E}{\partial Vx_{ij}} = {} & B_1 \cdot p_j \cdot v'(\sum_{l=1}^{n} p_l Vx_{il} - VC_{\max}) + \\ & B_2 \cdot 2 \cdot \sum_{k=1, k\neq i}^{m} Vx_{kj} + \\ & B_3 \cdot 2 \cdot (\sum_{k=1}^{m} \sum_{l=1}^{n} Vx_{kl} - n) \end{aligned} \tag{17}$$

where v' is the derivative of the penalty function v.

$$v'(\xi) = \begin{cases} 2\xi, & \xi > 0 \\ 0, & \xi \leq 0 \end{cases} \tag{18}$$

2.3 Time-varying penalty coefficients

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[6]. Recently, time based penalty parameters are proposed to overcome the tradeoff. In order to determine the appropriate values of the penalty parameters, Wang[7] used monotonically time-varying penalty parameters for solving convex programming problems. Akyol and Bayhan[4] used time varying penalty parameters increased in a linear fashion in a stepwise manner to reduce the feasible region.

In this paper, penalty parameters A, B_1, B_2, and B_3 are time variables, starting with small values and continuously increasing when their corresponding constraints are not satisfied during the optimization process. First, we try to satisfy the inequality constraints by penalizing them and run the simulations without considering any other constrains. Second, the column constraint is taken into consideration and the corresponding penalty parameter will be adjusted when both of the constraints are satisfied. Third, the penalty parameter of the objective function is set to 1 and the corresponding penalty parameter will be adjusted when all the constraints are satisfied.

2.4 Pseudocode of CTCNN based scheduling algorithm

In this subsection, the pseudocode of CTCNN based scheduling algorithm will be described in detail.

Step 1. Set parameters p_j, m, and n for IPMSP.

Step 2. Determine CTCNN parameters α, β, k, ε, I_0, and $z(0)$.

Step 3. Generate the initial neuron states randomly.

Step 4. Set penalty parameters $A = B_2 = B_3 = 0$ and $B_1 = 1$ (B_1 is the coefficient of the inequality constraint). If the constraint associated with parameter B_1 is satisfied, go to Step 5, otherwise increase the value of B_1 and then go to Step 7.

Step 5. Select parameter B_2 (a higher value than B_1 to increase the effect of the column constraint) and use the predetermined value of B_1 to check whether both of 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 7.

Step 6. Set $A = 1$, select parameter B_3 (a higher value than B_2 to increase the effect of the global constraint), and use the predetermined values of B_1 and B_2 to check whether all the constraints associated with these parameters are satisfied. If yes, go to Step 7, otherwise increase the value of parameter whose associated constraint is not satisfied and then go to Step 7.

Step 7. Update all the neurons using (5) $\sim$ (9), (16) $\sim$ (18) and repeat a number of times. If $A = B_2 = B_3 = 0$, go to Step 4. If $A = B_3 = 0$, go to Step 5. If $A = 1$, increase the value of parameter whose associated constraint is not satisfied.

Step 8. If the end condition is not satisfied, go to Step 7, otherwise stop the evolution and check the feasibility and optimality of the final solution.

3 Simulation analysis

To evaluate performance of the proposed CTCNN, computational experiments were performed on randomly generated test problems. For the experiments, 3 different size problems were generated. The processing time p_j of job j was generated from the uniform distribution over the interval [1.00, 5.00]. Considering the solution quality depending highly on initial conditions, all the solutions were obtained by simulations with 100 random initial conditions.

CTCNN parameters for 3 different size problems were determined by trial and error as shown in Table 2. Parameter α affects the energy function to generate transient chaos. Parameter β governs the bifurcation speed of the transient chaos. In Table 3, the results are compared with those of Hopfield-like dynamic neural network (HDNN) proposed in [4] in terms of best $C_{\max}$, average $C_{\max}$, worst $C_{\max}$ 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 networks were trapped into local minima more easily. Compared to HDNN, the proposed CTCNN might find a better solution to $C_{\max}$.

Table 2 CTCNN parameters for 3 different size problems

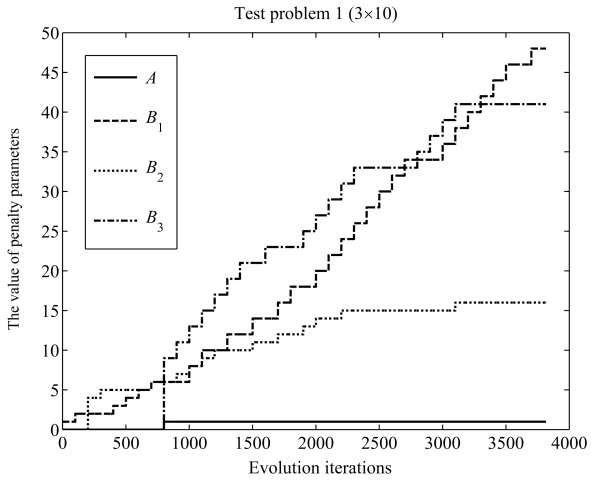
Problem size ($m \times n$)	k	α	β	I_0	ε	$z(0)$
3×10	0.997	0.008	0.001	0.65	0.008	0.08
3×20	0.997	0.010	0.002	0.65	0.008	0.08
3×50	0.998	0.015	0.0008	0.5	0.008	0.1

We had to rely upon time varying penalty parameters, as there was no systematic guidance available as to what the values of the parameters ought to be. However, in the ex-

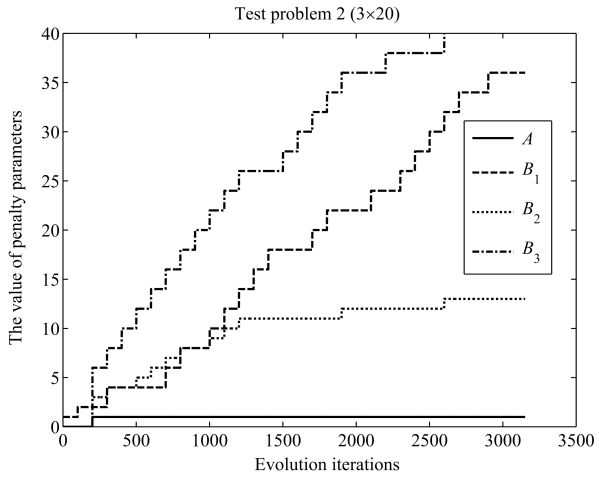
Table 3 Results of 100 different initial conditions for 3 problems

	CTCNN				HDNN			
Problem size ($m \times n$)	Best $C_{\max}$	Average $C_{\max}$	Worst $C_{\max}$	PFS	Best $C_{\max}$	Average $C_{\max}$	Worst $C_{\max}$	PFS
3×10	9.74	10.91	11.49	100%	10.35	11.71	12.42	92%
3×20	20.00	20.52	22.18	96%	21.08	22.29	23.36	88%
3×50	47.61	48.52	49.10	89%	48.84	50.18	51.41	73%

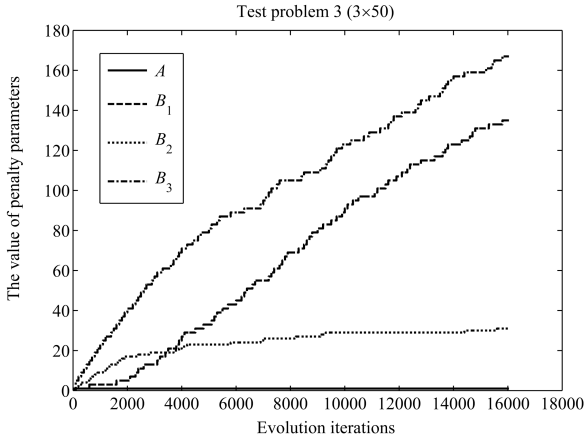
periments, an appropriate parameter set leading to a good solution could be obtained within a few runs, each of which used random initial state. It was noticed that when values were too large, the network could not achieve the steady state. Fig. 1 shows the time varying penalty parameters of the best solutions for the 3 different size problems. The coefficient B_1 of the inequality constraint and coefficient B_3 of the global constraint were higher than parameter B_2 of the column constraint during the evolution process. Along with the increase of the test problem size, evolution iterations aggrandized correspondingly. Fig. 2 is the Gantt chart of the best solution for test problem 3 (3×50). The computation results show that CTCNN proposed for IPMSP can provide efficient solutions for medium even large sized problems.



(a)



(b)



(c)

Fig. 1 Time varying penalty parameters of the best solutions for the 3 test problems

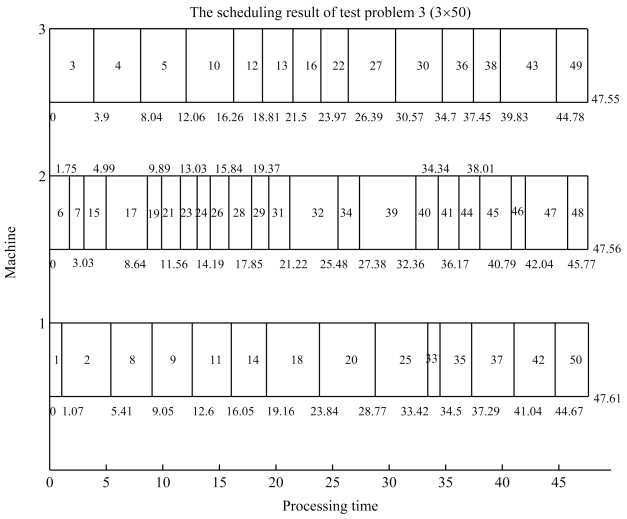


Fig. 2 Gantt chart of the best solution for test problem 3 (3×50)

4 Conclusion

In this paper, a coupled transiently chaotic neural network is presented to solve identical parallel machine scheduling problems. A new computational energy function for IPMSP is proposed to obtain the near-optimal and optimal solutions, including the objective and all the constraints. Moreover, the tradeoff problem existing among the penalty terms included in the energy function is overcome by using time-varying penalty parameters. Simulation experiments show that the proposed network can generate feasible solutions. After running simulations with

time evolving penalty coefficients, near optimal and optimal solutions can be found in a reasonable and finite time. Because parallel machine scheduling problems are NP-hard, it is apparent that an optimum solution to a large problem is difficult to find. However, in most cases, what is truly desired is a very good solution.

CTCNN model for the basic identical parallel machine scheduling problem is constructed in this paper. Further research directions can topics hotspots such as construction of models with setup times, job splitting, and ready time for IPMSP or uniform parallel machine scheduling problems. In addition, more effective architecture of NNs, permutation matrix expression, and construction of more suitable energy function might also serve as future research topics.

References

1 Thatcher J W. *Complexity of Computer Computations*. New York: Plenum Press, 1972

2 Hopfield J J, Tank D W. Neural computation of decisions in optimization problems. *Biological Cybernetics*, 1985, **52**(3): 141−152

3 Tank D W, Hopfield J J. Simple neural optimization networks: an A/D converter, signal decision circuit, and a linear programming circuit. *IEEE Transactions on Circuits and Systems*, 1986, **33**(5): 533−541

4 Akyol D E, Bayhan G M. Minimizing makespan on identical parallel machines using neural networks. In: Proceedings of International Conference on Neural Information Processing. Heidelberg, Germany: Springer Berlin, 2006. 553−562

5 Chen L, Aihara K. Chaotic simulated annealing by a neural network model with transient chaos. *Neural Networks*, 1995, **8**(6): 915−930

6 Watta P B, Hassoun M H. A coupled gradient network approach for static and temporal mixed-integer optimization. *IEEE Transactions on Neural Networks*, 1996, **7**(3): 578−593

7 Wang J. A time-varying recurrent neural system for convex programming. In: Proceedings of International Joint Conference on Neural Networks. Seattle, USA: IEEE, 1991. 147−152

YU Ai-Qing Ph. D. candidate in Research Institute of Automation at East China University of Science and Technology. Her research interest covers intelligent optimization for production scheduling. E-mail: yuaiqing@mail.ecust.edu.cn

GU Xing-Sheng Professor in Research Institute of Automation at East University of Science and Technology. His research interest covers control theory and control engineering. Corresponding author of this paper. E-mail: xsgu@ecust.edu.cn