17

Identical Parallel Machine Scheduling with Dynamical Networks using Time-Varying Penalty Parameters

Derya Eren Akyol
Dokuz Eylul University
Turkey

1. Introduction

The classical identical parallel machine scheduling problem can be stated as follows: Given n jobs and m machines, the problem is to assign each job on one of the identical machines during a fixed processing time so that the schedule that optimizes a certain performance measure is obtained. Having numerous potential applications in real life, in recent years, various research works have been carried out to deal with the parallel scheduling problems.
The literature of parallel machine scheduling problems has been extensively reviewed by (Cheng & Sin, 1990; Mokotoff, 2001). Among many criteria, minimizing makespan (maximum completion time) has been one of the most widely studied objectives in the literature.
Using the three-field classification introduced in (Graham et al., 1976), the problem is denoted in the scheduling literature as $P||Cmax$ where P designates the identical parallel machines, C_{max} denotes the makespan. We assume, as is usual, that the processing times are positive and that $1<m<n$. The problem is known to be NP-hard in the strong sense (Garey & Johnson, 1979; Sethi, 1977).
Although traditional techniques such as complete enumeration, dynamic programming, integer programming, and branch and bound were used to find the optimal solutions for small and medium sized problems, they do not provide efficient solutions for the problems with large size. Having found no efficient polynomial algorithm to find the optimal solution led many researchers to develop heuristics to obtain near optimal solutions. Though, efficient heuristics can not guarantee optimal solutions, they provide approximate solutions as good as the optimal solutions. These can be broadly classified into constructive heuristics and improvement heuristics. Most of the algorithms belong to the first category and have known worst case performance ratio (Coffman et al., 1978; Friesen & Langston, 1986; Friesen, 1987; Graham, 1969; Hochbaum & Shmoys, 1987; Leung, 1989; Sahni, 1976). The LPT rule of Graham, one of the most popular constructive heuristics, has been shown to perform well for the makespan criterion. This rule arranges jobs in descending order of processing times, such that $p_1 \geq p_2 \geq \ldots \geq p_n$, and then successively assigns jobs to the least loaded machine. The MULTIFIT algorithm, a classical constructive heuristic developed by (Coffman et al., 1978), determines the smallest machine capacity to find a feasible solution using the LPT scheme. This is achieved by solving heuristically a series of bin packing

Source: Multiprocessor Scheduling: Theory and Applications, Book edited by Eugene Levner,
ISBN 978-3-902613-02-8, pp.436, December 2007, Itech Education and Publishing, Vienna, Austria

problems. Although MULTIFIT is not guaranteed to perform better than LPT, it has been shown that it has a worst case bound better than LPT.

Improvement based algorithms are based upon local search in a neighbourhood in which a feasible solution is taken as a starting point and then tried to be improved by iterative changes. Application of these algorithms to the P||Cmax problem can be found in (Frangioni et al., 1999; Hübscher & Glover, 1994; Jozefowska et al. 1998).

Although a large number of approaches such as mathematical programming, dispatching rules, expert systems, and neighborhood search to the modeling and solution of scheduling problems have been reported in the literature, over the last decades, there has been an explosion of interest in using Artificial Neural Networks (ANNs) for the solution of various scheduling problems. This is mainly after the success of the use of Hopfield and Tank's network (Hopfield & Tank, 1985) in solving the Traveling Salesman Problem. The authors showed that if an optimization problem can be represented by an energy function, then a Hopfield network that corresponds to this energy function can be used to minimize this function to provide an optimal or near-optimal solution. Since then, a variety of scheduling problems are solved using Hopfield type networks (Chen & Dong; 1999; Foo et al. 1995; Liansheng et al., 2000; Lo & Bavarian, 1993; Satake et al. 1994; Vaithyanathan & Ignizio, 1992; Willems & Brandts; 1995; Zhou et al., 1991).

But a few papers are proposed for the solution of parallel machine scheduling problem using ANNs. Park et al. (2000) presented a backpropagation network for solving identical parallel machine scheduling problems with sequence dependent set up times. They tried to find the sequence of jobs processed on each machine with the objective of minimizing weighted tardiness. Hamad et al. (2003) dealt with the non-identical parallel machines problem with the sum of earliness and tardiness cost minimization and proposed a way of representing the problem to be fed into a backpropagation network. Akyol & Bayhan (2005) proposed a coupled gradient network approach for solving the earliness and tardiness scheduling problem involving sequence dependent setup times.

The objective of this research is to apply ANNs to the identical parallel machine scheduling problem for minimizing the makespan. We employ a dynamical gradient network approach to attack the problem and this work is an extension of the work of Akyol & Bayhan (2006) where they consider only a small sized scheduling problem and analyze the effect of 5 different initial conditions on the solutions. In this study, after the appropriate energy function is constructed by using a penalty function approach, the dynamics are defined by steepest gradient descent on the energy function. In order to overcome the tradeoff problem encountered in using the penalty function approach, a time varying penalty coefficient methodology is proposed to be used. By performing simulation experiments, we analyze the impact that the initial conditions of the network have on the performance on 5 different data sets by running each data set 20 times (20 different initial conditions) for different sizes of jobs and machines.

2. Problem Statement

Consider a set J of n jobs J_i ,$i=1,...,n$ to be processed, each of them on one machine, on a set M of m machines M_j , $j=1,...,m$. All the jobs can be processed on any of the m machines. We consider identical machines models, for which the processing times of each job, p_i, are machine independent. The objective is to find an appropriate allocation of jobs to machines

that would optimize a performance criterion. We are interested in the makespan criterion (maximum completion time), *Cmax*.
The following notations are used throughout the rest of this paper.
J_i : job *i*, $i \in N=\{1,...,n\}$
M_j : machine *j*, $j \in M=\{1,...,m\}$
p_i: processing time of J_i
C_i: completion time of J_i
C_{max}: makespan, the maximum completion time of all jobs
$C_{max} = \max\{C_1, C_2, ...,C_n\}$
x_{ij} : 0/1 assignment variable = $\begin{cases} 1 & \text{if job } i \text{ is assigned to machine } j \\ 0 & \text{otherwise} \end{cases}$
A MIP formulation of the minimum makespan problem can be defined as follows:
min C_{max}
subject to

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

$$C\max - \sum_{i=1}^{n} p_i x_{ij} \ge 0 \qquad 1 \le j \le m \tag{2}$$

The first constraint given in (1) ensures that each job is assigned to only one machine. The second constraint given in (2) ensures that the makespan is at least the completion time of each machine.

3. Design of the Proposed Dynamical Gradient Network

In this section, we describe how dynamical gradient networks can be used to solve the considered problem presented in the previous section. The proposed approach is an extension of the original formulation given in (Hopfield, 1984; Hopfield & Tank, 1985). Firstly the network architecture is explained, and then derivation of the energy function representing the proposed network, and dynamics and proof of the convergence of the proposed network are given. Finally, the proposed penalty parameter determination method is illustrated with an example.

3.1 The Network Architecture

The proposed gradient network has two types of neurons: a continuous type neuron to represent real valued variable Cmax, and discrete types of neurons to represent binary valued variables $X_{11},\ldots, X_{1m;}\ X_{21},\ldots, X_{2m}; X_{n1},\ldots,X_{nm}$. UX_{ij} symbolizes the input to the neuron for job i and resource j, and UCmax denotes the input to the neuron representing Cmax. The dynamics of the gradient net will be defined in terms of these input variables.
VX_{ij} designates the output of the neuron for job i and resource j. This neuron will be activated if job i is allocated to resource j. VCmax depicts the output of the neuron representing Cmax. We use a linear type activation function for neuron Cmax. Neurons with sigmoidal nonlinearity are used to represent discrete variables X_{ij}, so that the activation function for discrete neurons will take the usual sigmoidal form with slopes λ_X. Here, we

use a log-sigmoid function to convert discrete neurons to continuous ones and its functional form is shown in Figure 1.

3.2 The Energy Function

Instead of using linear programming or the *k-out-of-N* rules to develop the energy function, we directly formulate the cost function according to the constraints term by term. The energy function for this network is constructed using a penalty function approach. That is the energy function E consists of the objective function C_{max} plus a penalty function to enforce the constraints. For the problem considered, the penalty function P(X, Cmax) will include three penalty terms: P1, P2 and P3.

The first term P1 adds a positive penalty if the solution does not satisfy any of the equality constraints given in (3). In other words, the first term attempts to ensure that each job is allocated to one only one machine.

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

In this case, P1 = $\sum_{i=1}^{n}(\sum_{j=1}^{m} X_{ij} - 1)^2$. This term yields zero when these equality constraints are satisfied. P2 adds a positive penalty if the solution does not satisfy any of the inequality constraints given in (4).

$$C\,\mathbf{max} - \sum_{i=1}^{n} p_i x_{ij} \ge 0 \qquad 1 \le j \le m \cdot \tag{4}$$

In accordance with this constraint, P2 will take the following form $\sum_{j=1}^{m} v(\sum_{i=1}^{n} p_i X_{ij} - C_{\max})$ where v represents the penalty function. $v(\varepsilon) = \varepsilon^2$ *for all* $\varepsilon > 0$ *and* $v(\varepsilon) = 0$ *for all* $\varepsilon \le 0$ (Watta & Hassoun, 1996) and the functional form of this function is shown in Figure 2.

We require that X_{ij} ϵ{0,1}. These constraints will be captured by P3 which adds a positive penalty if the binary constraints X_{ij} ϵ{0,1} are violated. In Fig. 3, the functional form of this penalty term is shown. It can be seen that the penalty will be zero at either $X_{ij} = 0$ or $X_{ij} = 1$.

P3 = $\sum_{i=1}^{n}\sum_{j=1}^{m} X_{ij}(1 - X_{ij})$ and correspondingly, the total penalty function P (X, Cmax) with all constraints can be induced as follows.

$$\min\ B\sum_{i=1}^{n}(\sum_{j=1}^{m} X_{ij} - 1)^2 + C\sum_{j=1}^{m} v(\sum_{i=1}^{n} p_i X_{ij} - C_{\max}) + D\sum_{i=1}^{n}\sum_{j=1}^{m} X_{ij}(1 - X_{ij})$$

The complete energy function can thus be written as:

$$\min\ AC_{\max} + B\sum_{i=1}^{n}(\sum_{j=1}^{m} X_{ij} - 1)^2 + C\sum_{j=1}^{m} v(\sum_{i=1}^{n} p_i X_{ij} - C_{\max}) + D\sum_{i=1}^{n}\sum_{j=1}^{m} X_{ij}(1 - X_{ij})$$

where A, B, C and D are positive penalty coefficients.

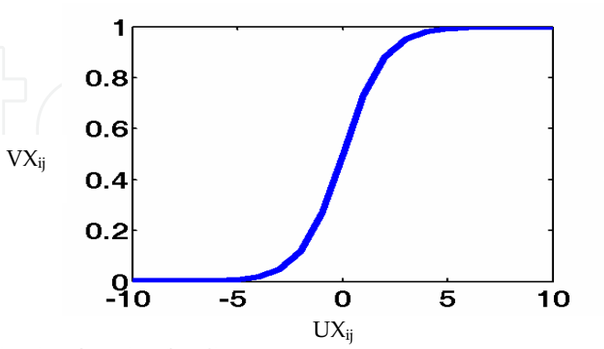


Figure 1. Activation function for discrete neurons

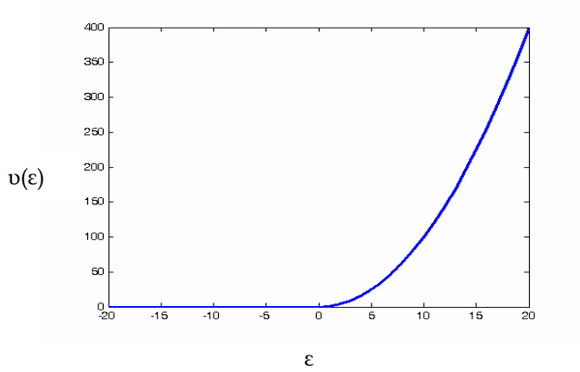


Figure 2. Penalty function for enforcing inequality constraints

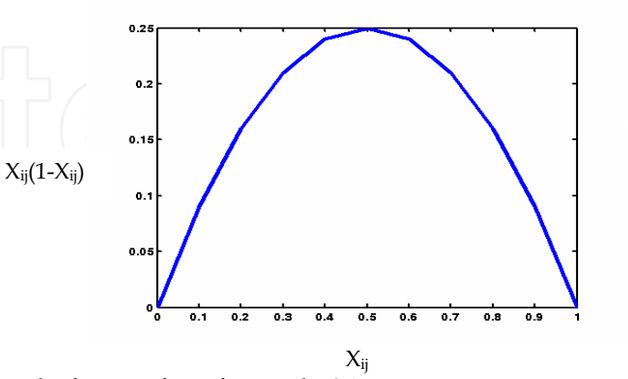


Figure 3. Penalty function for enforcing the 0,1 constraints

3.3 The Dynamics

In addition to defining the energy function to be employed, we need to consider the equation of motion of the neuron input. The dynamics for the gradient network are obtained by gradient descent on the energy function. The equations of motion are obtained as follows.

$$\frac{dUC_{\max}}{dt} = -\frac{\partial E}{\partial C_{\max}} = -A - (-1)C\sum_{j=1}^{m} \upsilon'[\sum_{i=1}^{n} P_i X_{ij} - C_{\max}] \tag{5}$$

$$\frac{dUX_{ij}}{dt} = -\frac{\partial E}{\partial X_{ij}} = -2B[\sum_{k=1}^{m} X_{ik} - 1] - C(Pi)\upsilon'[\sum_{l=1}^{n} p_l X_{lj} - C_{\max}] - D(1 - 2X_{ij}) \tag{6}$$

where η_{Cmax} and η_X are positive coefficients which will be used to scale the dynamics of the network, and ν′ is the derivative of the penalty function ν.

$$\upsilon'(\varepsilon) = 2\varepsilon \ \textit{for all} \ \varepsilon \rangle 0 \ \textit{and} \ \upsilon'(\varepsilon) = 0 \ \textit{for all} \ \varepsilon \le 0$$

The computation is performed in all neurons at the same time so that the network operates in a fully parallel mode.

The solution of equations of motion may be accomplished via the use of Euler's approximation. The states of the neurons are updated at iteration *k* as follows.

$$UC_{\max}{}^{k} = UC_{\max}{}^{k-1} + \eta_{C\max} \frac{dUC\max}{dt} \tag{7}$$

$$UX_{ij}{}^{k} = UX_{ij}{}^{k-1} + \eta_X \frac{dUX_{ij}}{dt} \tag{8}$$

Neuron outputs are calculated by V=g (U), where g (.) is the activation function, U is the input and V is the output of the neuron.

VCmax=g(UCmax) = UCmax (a linear function)

VX_{ij} = g(UX_{ij}) = logsig ($\lambda_X \times UX_{ij}$) (a log-sigmoid function)

where λ_X is the slope of the activation function and logsig(n) = 1 / (1 + exp(-n)).

3.4 Proof of Convergence

In order to use the proposed Hopfield-like dynamical network for the solution of the problem, we have to prove the convergence of the network. To do so, we have to show that energy does not increase along the trajectories, energy is bounded below, the solutions are bounded and time derivative of the energy is equal to zero only at equilibria.

Consider the time derivative of the energy function E given below.

$$\begin{aligned}\frac{dE}{dt} &= \sum_{i=1}^{n}\sum_{j=1}^{m}\frac{\partial E}{\partial VX_{ij}}\frac{\partial VX_{ij}}{dt}+\frac{\partial E}{\partial VC\max}\frac{\partial VC\max}{dt}\\ &= \sum_{i=1}^{n}\sum_{j=1}^{m}-\frac{dUX_{ij}}{dt}\frac{dVX_{ij}}{dt}+\frac{\partial E}{\partial VC\max}\frac{\partial VC\max}{dt}\\ &= \sum_{i=1}^{n}\sum_{j=1}^{m}\left(-\frac{dVX_{ij}}{dt}\right)\frac{dUXij}{dVX_{ij}}\left(\frac{dVX_{ij}}{dt}\right)+\left(\frac{-dUC\max}{dt}\right)\frac{dUC\max}{dt}\\ &= \sum_{i=1}^{n}\sum_{j=1}^{m}-\left(\frac{dVX_{ij}}{dt}\right)^{2}\frac{dUX_{ij}}{dVX_{ij}}-\left(\frac{dVC\max}{dt}\right)^{2}\end{aligned} \tag{9}$$

Since $\frac{dUX_{ij}}{dVX_{ij}} = [g^{-1}(VX_{ij})]' \geq 0$ (monotone increasing) for log-sigmoid function, the right-hand side of the equation given in (9) will be obviously negative. This ensures that the energy does not increase along trajectories, so we can write $\frac{dE}{dt} \leq 0$.

$\frac{dE}{dt} = 0$ implies that $\frac{dVX_{ij}}{dt} = 0 \quad for\ all\ i, j$ and $\frac{dVC\max}{dt} = 0$. In other words, $\frac{dE}{dt} = 0$ at the equilibrium points.

Since X_{ij}s are binary variables, they are bounded but we have to check the boundedness of Cmax. If we rewrite the motion equation for Cmax, we obtain the following:

$$\begin{aligned}\frac{dUC_{\max}}{dt} &= -\frac{\partial E}{\partial VC_{\max}}\\ &= -A-(-1)C\sum_{j=1}^{m}\upsilon'[\sum_{i=1}^{n}P_iVX_{ij}-VC_{\max}]\end{aligned}$$

There may be different possible cases

Case 1: Assume that $\sum_{i=1}^{n}P_iVX_{ij}-VC_{\max}=0 \rightarrow \frac{dUC_{\max}}{dt}=-A$

which means that UCmax=VCmax will decrease.

This will cause

$$\sum_{i=1}^{n}P_iVX_{ij}-VC_{\max} > 0.$$

Case 2: Assume that $\sum_{i=1}^{n} P_i VX_{ij} - VC_{\max} < 0 \rightarrow \frac{dUC_{\max}}{dt} = -A$

which means that UCmax=VCmax will decrease.
This will cause

$$\sum_{i=1}^{n} P_i VX_{ij} - VC_{\max} > 0 .$$

Therefore we have to consider Case 3 in which we assume $\sum_{i=1}^{n} P_i VX_{ij} - VC_{\max} > 0$

$$\text{If } \sum_{i=1}^{n} P_i VX_{ij} - VC_{\max} > 0$$

$$\frac{dUC_{\max}}{dt} = \frac{dVC_{\max}}{dt} = -\frac{\partial E}{\partial VC_{\max}}$$

$$= -A - (-1)C\sum_{j=1}^{m} \upsilon'[\sum_{i=1}^{n} P_i VX_{ij} - VC_{\max}]$$

$$= -A - 2CVC_{\max} + 2C\sum_{i=1}^{n} P_i VX_{ij}$$

$$\text{Let r(z)} = -A + 2C\sum_{i=1}^{n} P_i VX_{ij}$$

$$\frac{dVC_{\max}}{dt} = -2CVC_{\max} + r(z)$$

If we multiply both sides by e^{Ct},

$$\text{we get} \quad e^{Ct} \cdot \frac{dVC_{\max}}{dt} = -2CVC_{\max} \cdot e^{Ct} + r(z) \cdot e^{Ct}$$

$$e^{Ct} \cdot \frac{dVC_{\max}}{dt} + 2CVC_{\max} \cdot e^{Ct} = r(t) \cdot e^{Ct}$$

$$= \frac{d}{dt}[e^{Ct} VC_{\max}] = r(t)e^{Ct}$$

$$\int_{VC\max(0)e^{0.C}}^{VC\max(t)e^{Ct}} d(VC_{\max}(t).e^{Ct}) = \int_0^t e^{Cz} r(z)dz$$

$$= VC_{\max}(t)e^{Ct} - VC_{\max}(0) = \int_0^t e^{Cz} r(z)dz$$

$$VC_{\max}(t) = e^{-Ct} VC_{\max}(0) + \int_0^t e^{-Ct} e^{Cz} r(z)dz$$

$$= e^{-Ct} VC_{\max}(0) + e^{-Ct} \int_0^t e^{Cz} r(z)dz$$

$$\left|VC_{\max}(t)\right| = \left| e^{-Ct} VC_{\max}(0) + e^{-Ct} \int_0^t e^{Cz} r(z)dz \right|$$

$$\leq \left| e^{-Ct} \cdot VC_{\max}(0) \right| + \left| e^{-Ct} \cdot \int_0^t e^{Cz} r(z)dz \right|$$

$$\leq e^{-Ct} \left|VC_{\max}(0)\right| + e^{-Ct} \cdot \left| \int_0^t e^{Cz} r(z)dz \right|$$

We can write

$$\left|VC_{\max}(t)\right| \leq e^{-Ct} \left|VC_{\max}(0)\right| + e^{-Ct} \cdot \left| \int_0^t e^{Cz} r(z)dz \right|$$

$$\leq e^{-Ct} \left|VC_{\max}(0)\right| + e^{-Ct} \cdot \int_0^t \left|e^{Cz}\right| \left|r(z)\right| dz$$

Assume that $\left|r(z)\right| \leq M < \infty$

$$\left|VC_{\max}(t)\right| \leq \left|e^{-Ct}\right| \left|VC_{\max}(0)\right| + e^{-Ct} \cdot M \int_0^t e^{Cz} dz$$

$$\leq e^{-Ct} \left|VC_{\max}(0)\right| + e^{-Ct} \cdot M \cdot \frac{1}{c} [e^{Ct} - 1]$$

$$\leq e^{-Ct} \left|VC_{\max}(0)\right| + \frac{M}{c} [1 - e^{-Ct}]$$

Since $\left|e^{-Ct}\right| \leq 1$ and $e^{-Ct} \to 0$ as $t \to \infty$, then

$$\left|VC_{\max}(t)\right| \leq \infty$$

$$r(z) = -A + 2C\sum_{i=1}^{n} P_i VX_{ij}$$

$$\left|r(z)\right| = \left|-A + 2C\sum_{i=1}^{n} P_i VX_{ij}\right|$$

$$\leq \left|-A\right| + \left|2C\sum_{i=1}^{n} P_i VX_{ij}\right|$$

$$\leq A + 2C\sum_{i=1}^{n} P_i VX_{ij}$$

Since A>0 and $2C\sum_{i=1}^{n} P_i VX_{ij} > 0$

$$\left|r(z)\right| < \infty \text{ and } \frac{dUC_{\max}}{dt} < \infty$$

and we can conclude that the solutions are bounded.
Combining this fact with the fact that the energy E is bounded (since the cost is always greater than zero), we conclude that the network converges to a stable state which is a local minimum of E(X,Cmax). In other words, the time evolution of the network is a motion in space tends to that minimum point as *t* goes to infinity.

3.5 Selection of Parameters

In order to simulate the proposed network for solving the problem described by the dynamics given in Section 3.3, some parameters should be determined. These are the penalty parameters A, B, C and D; the activation slopes $\lambda_{X;}$ the step sizes η_{Cmax}, η_X and the initial conditions.
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 empirically. That is simulation runs are conducted, and optimality and/or feasibility of the resulting equilibrium points of the system are observed. The network can be initialized to small random values, and then synchronous or asynchronous updating of the network will allow a minimum energy state to be attained. In order to ensure smooth convergence, step size must be selected carefully (Watta & Hassoun, 1996).
The dynamics of the proposed Hopfield-like gradient network will converge to local minima of the energy function E. Since the energy function includes four terms, competing to be minimized, there are many local minima and a tradeoff among the terms. An infeasible solution may be obtained when at least one of the constraint penalty terms is non-zero. In

this case, the objective function term will generally be quite small but the solution will not be feasible. Alternatively, a local minimum, which causes a feasible but not a good solution, may be encountered even if all the constraints are satisfied. In order to satisfy the each penalty term, its associated penalty parameter can be increased but this results an increase in other penalty terms and a tradeoff occurs. The penalty parameters that result a feasible and a good solution, which minimizes the objective function, should be found.

Determining the appropriate values of the penalty parameters, network parameters and initial states are so critical issues associated with gradient type networks that by adjusting the parameters, the convergence performance to valid solutions can be improved. It is clear that solving scheduling problems represented by many constraints will cause a tradeoff among the penalty terms to be minimized.

Due to the problems of Hopfield like NNs in solving optimization problems, various modifications are proposed to improve the convergence of the Hopfield network. While several authors modified the energy function of the Hopfield network to improve the convergence to valid solutions (Aiyer, Niranjan, & Fallside, 1990; Brandt, Wang, Laub & Mitra, 1988; Van Den Bout & Miller, 1988) many others studied the same formulation with different penalty parameters (Hedge, Sweet, & Levy, 1988; Kamgar-Parsi & Kamgar-Parsi, 1992; Lai & Coghill, 1992). In recent years, time based penalty parameters are proposed to overcome the tradeoff problems encountered in using penalty function approach. Wang (1991) used monotonically time-varying penalty parameters for solving convex programming problems. Dogan & Guzelis (2006) proposed linearly increasing time-varying penalty parameters for solving clustering problems. Here, we propose to use time varying penalty parameters that take zero values as a starting value and then are increased in a linear fashion in a stepwise manner to reduce the feasible region and also by updating all the neurons synchronously, better simulation results are obtained.

The proposed gradient network algorithm can be summarised by the following pseudo-code.

Step 1. Construct an energy function for the considered problem using a penalty function approach.

Step 2. Initialize all neuron states to random values.

Step 3. Select the slope of the activation function (λ) and step sizes (η).

Step 4. Determine penalty parameters

Step 4.1 Select C (the coefficient of the inequality constraint) and assign zero as initial value to other penalty parameters A, B and D. If the constraint associated with parameter C is satisfied, proceed to Step 4.2 otherwise go back to Step 4.1.

Step 4.2 Select D (a higher value than C to increase the effect of equality constraint), and use the predetermined value of C (without taking into consideration of the effect of parameter A and B) to check whether both of the constraints associated with these terms are satisfied. If yes go to step 4.3, otherwise to step 4.4.

Step 4.3. Select B (a higher value than D), assign 1 to A, and use the predetermined values of C, D together with B to check whether all of the constraints associated with these terms are satisfied. If yes go to step 5, otherwise to step 4.4.

Step 4.4 Increase the value of parameter whose associated constraint is not satisfied.

Step 5. Repeat n times:

Step 5.1. Update U using equations (7) and (8), and then compute V by V=g (U).

Step 6. If the energy has converged to local minimum proceed to step 7, otherwise go back to step 5.

Step 7. Examine the final solution to determine feasibility and optimality.
Step 8. Adjust parameters A, B, C, D if necessary to obtain a satisfactory solution, reinitialize neuron states and repeat from step 5.

3.6 An Example

We explain the procedure with a 5-job 3-machine identical parallel machine scheduling problem. After constructing the energy function for this problem, all neuron states are initialized to random values chosen uniformly from the interval [0,1]. In the proposed approach, we firstly suggest to satisfy the inequality constraint by penalizing it. In the first phase of the simulation (for the first 2000 iterations), initial value of the penalty parameter C is chosen as 8. Because other penalty parameters are not taken into consideration, they are equal to zero. Since this inequality constraint is satisfied after 2000 iterations, it is decided to proceed to the next phase. In the second phase (for iterations from 2001 to 4000), one of the equality constraints (binary constraints) is taken into consideration, and its associated parameter D is chosen as 20, a value greater than C. The predetermined value of C, 8, is used to penalize the inequality constraint. Both of the constraints are satisfied. Thus, it is decided to proceed to the next phase (for iterations from 4001 to 5000). In this phase, all of the constraints are tried to be satisfied. Together with the predetermined values of C and D, the penalty parameter B belonging to the assignment constraint is chosen as 100 (a value greater than other parameters). Since A belongs to the original objective function, it is not penalized, and we assign 1 to A. After running simulations with all these 4 penalty terms, the feasibility and optimality of the final solution is checked. It is seen that except the inequality constraint, being violated with a small percentage error, all of the constraints are satisfied. Therefore, it is decided to enhance the weight of this constraint, and then value of its parameter, C, is increased to 600. Optimal solution is found at iteration 5100. All of the constraints were met satisfactorily, and the cost value is 3.1. In Table 1, values of penalty parameters used during the solution of the problem considered are displayed.

Penalty Coef. / Iterations	A	B	C	D
1:2000	0	0	8	0
2001:4000	0	0	8	20
4001:5000	1	100	8	20
5001:5100	1	1	600	1

Table 1. Penalty parameter values in four phases of simulation

4. Simulation Results

A simulation experiment was conducted to test the effectiveness of the proposed gradient network approach in terms of solution quality. The initial conditions of the network and the processing times of jobs were chosen randomly from uniform distribution in an interval [0,1], and [1,3], respectively. In tables 2-11, penalty coefficients of the proposed gradient network and other parameters which were determined empirically by running trial simulations are given.

For each problem size, the gradient network was run for 20 different initial conditions on 5 different datasets. It is to be noted that the same set of penalty parameters are tried to be found for all the test sets of each problem size during simulations. By tuning the parameters for each dataset, it is possible to improve the performance of the proposed network.

Penalty Coef. / Iterations	A	B	C	D
1:2000	0	0	8	0
2001:4000	0	0	8	20
4001:5000	1	100	8	20
5001:5100	1	1	600	1

Table 2. Penalty coefficients during four phases of simulations for n=5 m=3

Penalty Coef. / Iterations	A	B	C	D
1:2000	0	0	8	0
2001:4000	0	0	8	20
4001:5000	1	100	8	20
5001:5100	1	1	600	1

Table 3. Penalty coefficients during four phases of simulations for n=10 m=3

Penalty Coef. / Iterations	A	B	C	D
1:2000	0	0	8	0
2001:4000	0	0	8	20
4001:5000	1	100	8	20
5001:5100	1	1	600	1

Table 4. Penalty coefficients during four phases of simulations for n=20 m=3

Penalty Coef. / Iterations	A	B	C	D
1:2000	0	0	8	0
2001:4000	0	0	8	20
4001:5000	1	100	8	20
5001:5100	1	1	600	1

Table 5. Penalty coefficients during four phases of simulations for n=50 m=3

Penalty Coef. / Iterations	A	B	C	D
1:2000	0	0	8	0
2001:4000	0	0	8	20
4001:5000	1	100	8	20
5001:5100	1	1	600	1

Table 6. Penalty coefficients during four phases of simulations for n=100 m=3

Penalty Coef. / Iterations	A	B	C	D
1:2000	0	0	8	0
2001:4000	0	0	8	20
4001:5000	1	100	8	20
5001:5100	1	1	600	1

Table 7. Penalty coefficients during four phases of simulations for n=10 m=5

Penalty Coef. / Iterations	A	B	C	D
1:2000	0	0	10	0
2001:4000	0	0	10	30
4001:5000	1	100	10	30
5001:5100	1	1	600	1

Table 8. Penalty coefficients during four phases of simulations for n=20 m=5

Penalty Coef. / Iterations	A	B	C	D
1:2000	0	0	10	0
2001:4000	0	0	10	30
4001:5000	1	100	10	30
5001:5100	1	1	600	1

Table 9. Penalty coefficients during four phases of simulations for n=50 m=5

Penalty Coef. / Iterations	A	B	C	D
1:2000	0	0	10	0
2001:4000	0	0	10	30
4001:5000	1	100	10	30
5001:5100	1	1	600	1

Table 10. Penalty coefficients during four phases of simulations for n=100 m=5

m	n	η_{Cmax}	η_X	λ_X
3	5	0.001	0.1	1
3	10	0.001	0.1	1
3	20	0.001	0.1	1
3	50	0.001	0.1	1
3	100	0.001	0.1	1
5	10	0.001	0.01	1
5	20	0.0008	0.01	1
5	50	0.0008	0.1	1
5	100	0.0008	0.1	1

Table 11. Other Parameters used in the simulation

The proposed procedure was implemented in Matlab language (Version 6.5) and run on a PC with a Pentium IV, 2.6 GHz processor having a 512 MB of RAM.

In tables 12-20, the solutions obtained by the gradient network using the determined parameters are compared with those of the well known LPT heuristic and with the optimum solutions found by Lingo (version 8.0), a linear programming software package, in terms of Best Cmax (cost of the best solution obtained by the gradient network), Avg. Cmax (cost of the average solution obtained by the gradient network), Worst Cmax (cost of the worst solution obtained by the gradient network), and % deviations. Columns (6) and (7) represent the % deviations of the proposed gradient network solution from the LPT rule solution and from the optimal solution, respectively. The % deviations reported in Columns (6) and (7) are given by

$$\%\ deviation\ from\ LPT = \frac{Avg.C\max(Gradient\ network) - C\max(LPT)}{C\max(LPT)} * 100\%$$

$$\%\ deviation\ from\ the\ optimal = \frac{Avg.C\max(Gradient\ network) - C\max(optimal)}{C\max(optimal)} * 100\%$$

where Avg. Cmax(Gradient network) is the average gradient network solution of the 20 runs, Cmax(LPT) is the LPT solution and Cmax(optimal) is the optimal solution obtained by the linear programming solver. The percentage of times, which resulted in a feasible solution by the network, was also displayed in the last columns of these tables. It is obvious that the negative % deviation values from the LPT dispatching rule represent the % improvement realized by the gradient network.

As our primary goal was to compare the proposed network solution with the LPT rule and with the optimal solutions, in terms of solution quality, the CPU times required for solving

each data set are not given. But from the simulation experiments, it is seen that when compared with the very long solution times needed to obtain the optimal solutions by the Lingo software, the proposed network could converge to valid solutions in reasonable times between 13.18 seconds (for n=3 m=5) and 203.57 seconds (for n=100 m=5). Obviously, by the implementation of the proposed network in a dedicated hardware, significant reductions can be obtained in running times.

Gradient Network			LPT (4)	Optimum (5)	Deviation (%) from the LPT solution (6)	Deviation (%) from the optimal solution (7)	Percent Feasibility of Computed Solutions (8)
Best Cmax (1)	Avg. Cmax (2)	Worst Cmax (3)					
3.1	3.1	3.1	3.1	3.1	0.00	0.00	100%
4.69	4.69	4.69	4.69	4.69	0.00	0.00	100%
3.55	3.55	3.55	3.55	3.55	0.00	0.00	100%
2.98	2.98	2.98	2.98	2.98	0.00	0.00	100%
3.02	3.02	3.02	3.02	3.02	0.00	0.00	100%

Table 12. Results for m=3, n=5 over 5 problems

Gradient Network			LPT (4)	Optimum (5)	Deviation (%) from the LPT solution (6)	Deviation (%) from the optimal solution (7)	Percent Feasibility of Computed Solutions (8)
Best Cmax (1)	Avg. Cmax (2)	Worst Cmax (3)					
7.33	7.54	7.67	7.59	7.21	-0.66	4.57	100 %
6.97	7.21	7.47	7.45	6.92	-3.22	4.19	100 %
7.28	7.56	7.72	7.69	7.2	-1.69	5	100 %
6.79	7.11	7.30	7.46	6.72	-4.69	5.80	100 %
6.77	7.01	7.31	7.44	6.72	-5.78	4.31	100 %

Table 13. Results for m=3, n=10 over 5 problems

Gradient Network			LPT (4)	Optimum (5)	Deviation (%) from the LPT solution (6)	Deviation (%) from the optimal solution (7)	Percent Feasibility of Computed Solutions (8)
Best Cmax (1)	Avg. Cmax (2)	Worst Cmax (3)					
13.24	13.53	13.85	13.37	13.05	1.19	3.68	100 %
13.84	14.24	14.46	14.01	13.74	1.64	3.64	100 %
13.03	13.42	13.63	13.40	12.92	0.15	3.87	100 %
14.25	14.54	14.76	14.60	14.05	-0.41	3.48	100 %
13.35	13.60	13.82	13.46	13.12	1.04	3.66	100 %

Table 14. Results for m=3, n=20 over 5 problems

Gradient Network			LPT (4)	Optimum (5)	Deviation (%) from the LPT solution (6)	Deviation (%) from the optimal solution (7)	Percent Feasibility of Computed Solutions (8)
Best Cmax (1)	Avg. Cmax (2)	Worst Cmax (3)					
33.53	33.84	34.07	33.70	33.34	0.41	1.50	100 %
30.58	30.95	31.14	30.75	30.36	0.65	1.94	100 %
31.47	31.85	32.15	31.65	31.38	0.63	1.49	100 %
34.53	35.41	35.77	35.32	34.92	0.25	1.40	100 %
34.68	35.10	35.30	34.88	34.51	0.63	1.71	100 %

Table 15. Results for m=3, n=50 over 5 problems

Gradient Network			LPT (4)	Optimum (5)	Deviation (%) from the LPT solution (6)	Deviation (%) from the optimal solution (7)	Percent Feasibility of Computed Solutions (8)
Best Cmax (1)	Avg. Cmax (2)	Worst Cmax (3)					
70.00	70.28	70.58	70.55	69.91	-0.38	0.53	100 %
66.65	66.94	67.14	67.09	66.45	-0.22	0.73	100 %
68.42	68.85	69.10	69.04	68.39	-0.27	0.67	100 %
66.11	66.73	66.52	66.73	66.09	0.00	0.97	100 %
65.85	66.15	66.33	66.35	65.69	-0.30	0.70	100 %

Table 16. Results for m=3, n=100 over 5 problems

Gradient Network			LPT (4)	Optimum (5)	Deviation (%) from the LPT solution (6)	Deviation (%) from the optimal solution (7)	Percent Feasibility of Computed Solutions (8)
Best Cmax (1)	Avg. Cmax (2)	Worst Cmax (3)					
3.43	3.53	3.68	3.43	3.43	2.91	2.91	100 %
3.38	3.76	3.97	3.79	3.38	-0.79	11.24	100 %
3.64	3.85	3.97	3.68	3.57	4.35	7.56	100 %
4.03	4.16	4.24	4.03	4.03	3.22	3.22	100 %
3.57	3.67	3.73	3.53	3.53	3.97	3.97	100 %

Table 17. Results for m=5, n=10 over 5 problems

Gradient Network			LPT (4)	Optimum (5)	Deviation (%) from the LPT solution (6)	Deviation (%) from the optimal solution (7)	Percent Feasibility of Computed Solutions (8)
Best Cmax (1)	Avg. Cmax (2)	Worst Cmax (3)					
7.43	7.78	7.91	7.37	7.28	5.56	6.87	100 %
7.68	7.95	8.08	7.62	7.49	4.33	6.14	100 %
8.13	8.24	8.37	7.8	7.76	5.64	6.18	100 %
7.79	7.98	8.13	7.69	7.51	3.77	6.26	100 %
8.55	8.77	8.92	8.29	8.18	5.79	7.21	100 %

Table 18. Results for m=5, n=20 over 5 problems

Gradient Network			LPT (4)	Optimum (5)	Deviation (%) from the LPT solution (6)	Deviation (%) from the optimal solution (7)	Percent Feasibility of Computed Solutions (8)
Best Cmax (1)	Avg. Cmax (2)	Worst Cmax (3)					
20.49	20.86	21.09	20.28	20.22	2. 86	3.16	100 %
21.70	22.17	22.42	21.55	21.49	2.88	3.16	100 %
18.69	18.94	19.15	18.42	18.40	2.82	2.93	100 %
20.71	21.11	21.33	20.37	20.33	3.63	3.83	100 %
19.79	20.01	20.24	19.43	19.41	2.98	3.09	100 %

Table 19 Results for m=5, n=50 over 5 problems

Gradient Network			LPT	Optimum	Deviation (%) from the LPT solution	Deviation (%) from the optimal solution	Percent Feasibility of Computed Solutions
Best Cmax (1)	Avg. Cmax (2)	Worst Cmax (3)	(4)	(5)	(6)	(7)	(8)
41.65	41.87	42.06	41.24	41.20	1.53	1.63	100 %
40.16	40.56	40.74	39.78	39.77	1.96	1.99	100 %
41.90	42.12	42.28	41.36	41.34	1.84	1.89	100 %
40.20	40.55	40.69	39.83	39.82	1.80	1.83	100 %
41.54	41.89	42.06	41.19	41.15	1.70	1.8	100 %

Table 20. Results for m=5, n=100 over 5 problems

To interpret the findings in a table, let us consider Table 12. For all the 5 data sets, 20 out of the 20 runs of the proposed network resulted in a feasible solution, that is percent feasibility is 100 %. The average, worst and the best cost of the 20 feasible solutions for the first dataset is 3.1, which is equal to the global optimal solution value, therefore the percent above the optimal solution and LPT result is 0.0. Similarly, if we consider Table 20, for the first dataset, again, 100 % of the runs resulted in a feasible solution by the proposed network. The average Cmax of the feasible solutions is 41.87, which is 1.53 % more costly than the result of LPT rule, and 1.63 % more costly than the global optimal solution. The best makespan value produced by the gradient network is 41.65, which is 0.99 % ([(41.65-41.24)*100]/41.24) above than the LPT result and 1.09 % ([(41.65-41.20)*100]/41.20) above the global optimal solution.
According to these findings, it is clear that the initial conditions of the network appear to have a serious impact on the solution quality. For example in Table 17, for n=10 and m=5, although the proposed network results in gaps between 2.91 and 4.35 % from the LPT solution, on average, it outperforms the LPT heuristic for one of the datasets. In the same table, if the results obtained using the first data set are considered, it is seen that although the average makespan from the 20 different initial runs is found as 3.53, the best makespan out of the 20 runs, produced by the proposed network is 3.43, which is equal to the optimal solution. In addition, although the average Cmax results obtained by the proposed network are above the LPT results for the 4 data sets, the best Cmax results outperform the LPT rule in 4 data sets.
In all the simulations carried out to show the performance of the network, convergence to valid schedules is achieved and better results are obtained for small number of machines and large number of jobs. If all the test cases are considered, the proposed network is, on average, able to produce a solution with a makespan value, which is 1.14 % above of cost of the LPT result. By tuning the penalty coefficients for each dataset, it is possible to improve the convergence and the optimality of the solutions. On the other hand, besides its convergence to valid schedules, convergence to good quality solutions of the proposed network points out its general applicability in other scheduling environments.

5. Conclusions and Future Research

This study has presented a dynamical gradient network for solving the identical parallel machine scheduling problem with the makespan criterion which is known to be NP-hard even for the case of two identical parallel machines. Focus of this paper has been on demonstrating the optimization capabilities of the proposed network by solving a set of randomly generated problems. The proposed Hopfield-like network uses time-varying penalty parameters that start from zero and increase in a stepwise manner during iterations to overcome the tradeoff problem of the penalty function method, one of the important drawbacks of the penalty function approach.. To analyse the performance of the network, it is compared with the well-known LPT heuristic commonly used to solve the problem under study, and also with the optimal solutions in terms of the solution quality. The simulation experiments demonstrated that the proposed network generated feasible solutions in all the cases, and, in some of the data sets it found smaller makespan compared to LPT. In general, for all the instances, the average deviation percentage of the proposed network is 1.14 % from the LPT heuristic.

By conducting several simulation experiments, the influence of different initializations schemes was investigated on the solutions of the problem considered. The analysis results showed that the percent error of the network is very sensitive to the selection of the starting points and the choice of the parameters used in simulation.

The contribution of this paper is two fold. We propose to use a novel time varying penalty method that guarantees feasible and near optimal solutions for solving the identical parallel machine scheduling problem with the makespan criterion. Although a large body of literature exists for solving identical parallel machine scheduling problem with the makespan minimization criterion, to the best of our knowledge, there is no previously published article that tried to solve this NP-hard problem using neural networks, so that this study will also make a contribution to the scheduling literature.

Several issues are worthy of future investigations. First, further studies will be focused on selecting the parameters of the network automatically rather than choosing by trial and error, which is one of the drawbacks of neural networks. Second, extension of the results to large size problems will be worthwhile. Finally, extension of the results to different manufacturing scheduling environments is important for industrial applications, and implementation of the network in hardware can make progress in computational efficiency.

6. References

Aiyer, S.V.B.; Niranjan, M. & Fallside, F. (1990). A theoretical investigation into the performance of the Hopfield model. *IEEE Transactions on Neural Networks*. 1, 204-215.

Akyol, D.E. & Bayhan, G.M. (2005). A Coupled Gradient Network Approach for the Multi Machine Earliness and Tardiness Scheduling Problem, *Lecture notes in computer science*, 3483, 596-605.

Akyol, D.E. & Bayhan, G.M. (2006). Minimizing Makespan on Identical Parallel Machines using Neural Networks, *Lecture notes in computer science*, 4234, 553-562.

Brandt, R.D.; Wang, Y.; Laub, A.J. & Mitra. S.K. (1988). Alternative Networks for Solving the Travelling Salesman Problem and the List-Matching Problem. In *Proceedings of the International Conference on Neural Networks*, 2, 333-340.

Chen, M. & Dong, Y. (1999). Applications of neural networks to solving SMT scheduling problems-a case study. *International Journal of Production Research*, 37, 4007-4020.

Cheng, T. & Sin, C. (1990). A State-of-the-Art Review of Parallel-Machine Scheduling Research. *European Journal of Operational Research*, 47, 271-292.

Coffman, E.G., Garey, M.R. & Johnson, D.S. (1978). An application of bin-packing to multi-processor scheduling, *SIAM Journal of Computing*, 7, 1-17.

Dogan, H. & Guzelis, C. (2006). Robust and Fuzzy Spherical Clustering by a Penalty Parameter Approach. *IEEE Transactions on Circuits and Systems-II. 53*(8), 637-641.

Foo, S.Y.; Takefuji, Y. & Szu, H. (1995). Scaling properties of neural networks for job-shop scheduling, *Neurocomputing*, 8, 79-91.

Frangioni, A.; Scutella, M.G. & Necciari, E. (1999). Multi-exchange algorithms for the minimum makespan machine scheduling problem, *Technical Report: TR-99-22*.

Friesen, D.K. & Langston, M.A. (1986). Evaluation of a MULTIFIT based scheduling algorithm, *J. Algorithm*, 7, 35-59.

Friesen, D.K. (1987). Tighter bounds for LPT scheduling on uniform processors. *SIAM J. Computing*. 16, 554-560.

Garey, M.R. & Johnson, D.S. (1979). Computer and intractability: a guide to the theory of NP completeness, (W.H Freeman, San Francisco).

Graham, R.L. (1969). Bounds on multiprocessor timing anomalies. *SIAM Journal of Applied Mathematics*, 17, 416-429.

Graham, R.L.; Lawler, E.L.; Lenstra, J.K. & Rinnooy Kan, A.H.G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5, 287–326.

Hamad, A.; Sanugi, B. & Salleh, S. (2003). A neural network model for the common due date job scheduling on unrelated parallel machines, *International Journal of Computer Mathematics*, 80, 845-851.

Hedge, S.; Sweet, J. & Levy, W. (1988). Determination of parameters in a Hopfield/Tank computational network. *In Proc. IEEE International Conference on Neural Networks*, 2, 291-298.

Hochbaum, D.S. & Shmoys, D.B. (1987). Using dual approximation algorithms for scheduling problems: Practical and theoretical results. *Journal of the Association for Computing Machinery*. 34, 144-162.

Hopfield, J. (1984). Neurons with graded response have collective computational properties like of two-state neurons. *In Proc. of the National Academy of Sciences of the USA*, 81, 3088-3092.

Hopfield, J. & Tank, T.W. (1985). Neural computation of decisions in optimization problems. *Biological Cybernetics*, 52, 141-152.

Hübscher, R. & Glover, F. (1994). Applying Tabu Search with influential diversification to multiprocessor scheduling, *Computers and Operations Research*, 8, 877-884.

Jozefowska, J.; Milka, M.; Rozycki, R.; Waligora, G. & Weglarz, J. (1998). Local search metaheuristics for discrete-continuous problems. *European Journal of Operational Research*, 107, 354-370.

Kamgar-Parsi, B. & Kamgar-Parsi, B. (1992). Dynamical Stability and Parameter Selection in Neural Optimization. P*roc. of International Joint Conference on Neural Networks*, 4, 566-571.

Lai, W.K. & Coghill, G.G. (1992). Genetic Breeding of Control Parameters for the Hopfield/Tank Neural Net. *Proc. of the International Joint Conference on Neural Networks*, 4, 618-623.

Leung, J.Y.-T. (1989). Bin packing with restricted piece sizes, *Information Processing Letters*, 31, 145-149.

Liansheng, G.; Gang, S. & Shuchun, W. (2000). Intelligent scheduling model and algorithm for manufacturing, *Production Planning and Control*, 11, 234-243.

Lo, Z.P. & Bavarian, B. (1993). Multiple job scheduling with artificial neural networks. *Computers and Electrical Engineering*, 19, 87-101.

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

Park, Y.; Kim, S. & Lee, Y.H. (2000). Scheduling jobs on parallel machines applying neural network and heuristic rules. *Computers and Industrial Engineering*, 38, 189-202.

Sahni, S.K. (1976). Algorithms for scheduling independent tasks. *J. Assoc. Comput mach.*, 23, 116-127.

Satake, T.; Morikawa, K. & Nakamura, N. (1994). Neural network approach for minimizing the makespan of the general job-shop. *International Journal of Production Economics*. 33, 67-74.

Sethi, R. (1977). On the complexity of mean flow time scheduling. *Mathematics of Operations Research*. 2, 320-330.

Vaithyanathan, S. & Ignizio, J.P. (1992). A stochastic neural network for resource constrained scheduling. *Computers and Operations Research*, 19, 241-254.

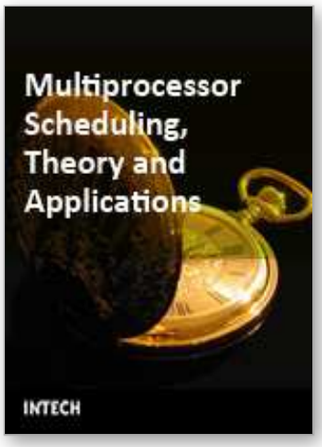
Van Den Bout, D.E. & Miller, T.K. (1988). A Traveling Salesman Objective Function that Works. *In Proc. of IEEE International Conference on Neural Networks*, 2, 299-303.

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

Watta, P.B. & Hassoun, M.H. (1996). A Coupled Gradient Network Approach for Static and Temporal Mixed-Integer Optimization. *IEEE Transactions on Neural Networks*, 7, 578-593.

Willems, T.M. & Brandts, E.M.W. (1995). Implementing heuristics as an optimization criterion in neural networks for job-shop scheduling. *Journal of Intelligent Manufacturing*, 6, 377-387.

Zhou, D.N.; Cherkassy, V.; Baldwin, T.R. & Olson, D.E. (1991). A Neural Network Approach to Job-Shop Scheduling. *IEEE Transactions on Neural Networks*, 2, 175-179.



Multiprocessor Scheduling, Theory and Applications

Edited by Eugene Levner

ISBN 978-3-902613-02-8

Hard cover, 436 pages

Publisher I-Tech Education and Publishing

Published online 01, December, 2007

Published in print edition December, 2007

A major goal of the book is to continue a good tradition - to bring together reputable researchers from different countries in order to provide a comprehensive coverage of advanced and modern topics in scheduling not yet reflected by other books. The virtual consortium of the authors has been created by using electronic exchanges; it comprises 50 authors from 18 different countries who have submitted 23 contributions to this collective product. In this sense, the volume can be added to a bookshelf with similar collective publications in scheduling, started by Coffman (1976) and successfully continued by Chretienne et al. (1995), Gutin and Punnen (2002), and Leung (2004). This volume contains four major parts that cover the following directions: the state of the art in theory and algorithms for classical and non-standard scheduling problems; new exact optimization algorithms, approximation algorithms with performance guarantees, heuristics and metaheuristics; novel models and approaches to scheduling; and, last but least, several real-life applications and case studies.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen