International Journal of Production Research

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tprs20

Dynamic job shop scheduling using variable neighbourhood search

M. Zandieh [a] & M.A. Adibi [b]

[a] Management and Accounting Faculty, Department of Industrial Management, Shahid Beheshti University, G.C., Tehran, Iran

[b] Faculty of Industrial and Mechanical Engineering, Qazvin Azad University, Qazvin, Iran

Version of record first published: 22 Apr 2009.

To cite this article: M. Zandieh & M.A. Adibi (2010): Dynamic job shop scheduling using variable neighbourhood search, International Journal of Production Research, 48:8, 2449-2458

To link to this article: http://dx.doi.org/10.1080/00207540802662896

International Journal of Production Research
Vol. 48, No. 8, 15 April 2010, 2449–2458

Taylor & Francis
Taylor & Francis Group

Dynamic job shop scheduling using variable neighbourhood search

M. Zandieh[a]* and M.A. Adibi[b]

[a]*Management and Accounting Faculty, Department of Industrial Management, Shahid Beheshti University, G.C., Tehran, Iran;* [b]*Faculty of Industrial and Mechanical Engineering, Qazvin Azad University, Qazvin, Iran*

(*Received 3 September 2008; final version received 26 November 2008*)

In this paper a scheduling method based on variable neighbourhood search (VNS) is introduced to address a dynamic job shop scheduling problem that considers random job arrivals and machine breakdowns. To deal with the dynamic nature of the problem, an event-driven policy is selected. To enhance the efficiency and effectiveness of the scheduling method, an artificial neural network with a back propagation error learning algorithm is used to update parameters of the VNS at any rescheduling point according to the problem condition. The proposed method is compared with some common dispatching rules that have been widely used in the literature for the dynamic job shop scheduling problem. Results illustrate the high efficiency and effectiveness of the proposed method in a variety of shop floor conditions.

Keywords: dynamic job shop scheduling; variable neighbourhood search; artificial neural networks

1. Introduction

The job shop scheduling (JSS) problem is one of the most interesting subjects for researchers and practitioners. It is because the JSS problem exists in various forms in most manufacturing systems. The JSS problem is well-known to be NP-hard and various methods like mathematical techniques, dispatching rules, artificial intelligence, artificial neural networks, neighbourhood searches, fuzzy logic, etc. are introduced to obtain an optimum (or a near to optimum) solution. But these methods are usually designed to address the static JSS problem and real time events such as random job arrivals and machine breakdowns are ignored. Taking these events into account, the JSS problem shifts to a new kind of problem that is well-known as the dynamic job shop scheduling (DJSS) problem. In the DJSS problem, one or more conditions of the problem like number of jobs or number of operable machines are changed by any new event. Therefore, the solution before the event is not good or even any longer feasible. So in addition to the scheduling problem, it is necessary to address two new issues '*when*' and '*how to*' react to dynamic events in DJSS problems.

In most research published around the DJSS problem, a scheduling method with predefined parameters is used in any rescheduling point (Qi *et al*. 2000, Chryssolouris and Subramanian 2001, Dominic *et al*. 2004, Liu *et al*. 2005, Zhou *et al*. 2008), but due to changes in a problem condition during the planning horizon, using a scheduling method

*Corresponding author. Email: m_zandieh@sbu.ac.ir

ISSN 0020–7543 print/ISSN 1366–588X online

DOI: 10.1080/00207540802662896
http://www.informaworld.com

with a fixed parameter can reduce the performance of the method. Preventing such a problem, Aydin and Oztemel (2000) used reinforcement learning agents to select an appropriate rule for scheduling according to the shop floor conditions in real time. Sha and Liu (2005) presented a model that incorporates a data mining tool for mining the knowledge of job scheduling about due date assignment in a dynamic job shop environment to adjust an appropriate parameter according to the condition of the shop floor at the instant of job arrival.

In recent years, the technological advancements in computer science have encouraged new application tools such as artificial neural networks to be applied as efficient approaches in a variety of engineering issues. Artificial neural network (ANN) models have been studied since the early 1980s, with an objective of achieving human like performance in many fields of science and are intended for modelling the organisational principles of the nervous system (Bose and Liang 1996). In ANNs, a network of processing elements is designed and mathematics carry out information processing for problems whose solutions require knowledge that is difficult to describe. ANNs can be used to predict appropriate parameters of a scheduling method at a rescheduling point using a pattern extracted from a learning sample. Compared with the traditional pattern recognition, ANN can provide an exact description for a multidimensional and non-linear problem (Yating *et al.* 2008).

In this study, variable neighbourhood search (Mladenovic and Hansen 1997) is selected as a scheduling method at any rescheduling point because it brings together a lot of desirable properties for a metaheuristic such as simplicity, efficiency, effectiveness, generality, etc. (Hansen *et al.* 2007). Variable neighbourhood search (VNS) is a new neighbourhood search metaheuristic that has been widely used to combinatorial optimisation problems in recent years. To enhance the efficiency and effectiveness of VNS, its parameters are updated at any rescheduling point by the ANN. Using a few effective parameters during the optimisation process, VNS is a more suitable scheduling method because using a scheduling method with fewer parameters increases prediction accurateness.

The rest of the paper is organised as follows: in Section 2, the dynamic job shop scheduling problem is defined in detail and the proposed method is introduced. A simulation study is presented in Section 3 and the conclusion is located in Section 4.

2. The dynamic job shop scheduling problem

2.1 *The job shop scheduling problem*

The general job shop scheduling model considers n jobs to be processed on m machines ($n \times m$ operations) while minimising some function of completion time of jobs, subject to the following technological constraints and assumptions: (i) each machine can perform only one operation at a time on any job; (ii) an operation of a job can be performed by only one machine at a time; (iii) once an operation has begun on a machine, it must not be interrupted; (iv) an operation of a job cannot be performed until its preceding operations are completed; (v) there are no alternate routings, i.e., an operation of a job can be performed by only one type of machine; and (vi) operation processing time and number of operable machines are known in advance.

In order to use a scheduling method to obtain an optimum solution for a JSS problem, mean flow time is used as an optimisation objective. Also, operation-based representation

and deduction of schedule (Amirthagadeswaran and Arunachalam 2006) is used in this article. This representation method encodes a schedule as a sequence of operations. Each number stands for one operation. The specific operation represented by a number is interpreted according to the order of the numbers in the string. Each of the n jobs will appear m times spread over the entire string. For instance, we are given a state of [2 1 3 2 3 3 1 1 2], where $\{1, 2, 3\}$ represents $\{job_1, job_2, job_3\}$ respectively. Obviously, there are in total nine operations, but three different integers are each repeated three times. The first integer, 2, represents the first operation of the second job, O_{21}, to be processed first on the corresponding machine. Likewise, the second integer, 1, represents the first operation of the first job, O_{11}. Thus, the set of [2 1 3 2 3 3 1 1 2] is understood as $[O_{21}, O_{11}, O_{31}, O_{22}, O_{32}, O_{33}, O_{12}, O_{13}, O_{23}]$, where O_{ij} stands for the jth operation of ith job.

2.2 *Variable neighbourhood search*

Variable neighbourhood search is a new local search technique that tries to escape from local optimum by changing the neighbourhood structure (NS), that is the manner in which the neighbourhood is defined. In its basic form, VNS explores a set of neighbourhoods of the current solution, makes a local search from a neighbour solution to a local optimum, and moves to it only if there has been an improvement (Perez *et al.* 2003).

The VNS algorithm that is used in this article begins with an initial solution, $x \in S$, where S is the set of search space, and uses a two-nested loop. The inner one alters and explores using two main functions namely 'shake' and 'local search'. The outer loop works as a refresher reiterating the inner loop. The local search explores for an improved solution within the local neighbourhood, while shake diversifies the solution by switching to another local neighbourhood. The inner loop iterates as long as it keeps improving the solutions, where an integer, k, controls the length of the loop. Once an inner loop is completed, the outer loop reiterates until the termination condition is met. The steps of VNS structure are illustrated in Figure 1.

Let N_k $(k = 1, 2, \ldots, k_{\max})$ denote neighbourhood structures for both shake and local search functions. To avoid costing too much computational time, the best number of

Initialisation: Select the set of neighborhood structures $N_k\,(k = 1,2...,k_{\max})$, which will be used in the search; find an initial solution x; choose a stopping condition;

Repeat the following until the stopping condition (N = number of iteration) is met:

(1) Set $k \leftarrow 1$;

(2) Until $k = k_{\max}$, repeat the following steps:

(a) *Shaking*. Generate a point x' at random from the k^{th} neighborhood of x ($x' \in N_k(x)$);

(b) *Local search*. Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;

(c) More or not. If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $N_1(k \leftarrow 1)$; otherwise, set $k \leftarrow k + 1$;

Figure 1. Steps of the basic VNS.

1. Get initial solution $x' \in S$, and dr.
2. Set $q \leftarrow 0$ and $u \leftarrow 1$
3. While $q < q_{\max}$ do
   (a) If $u = 1$ then $x'' \in S \leftarrow$ Insertion (x'); else $(u = 0)\, x'' \in S \leftarrow$ Swap (x')
   (b) If $f(x'') - f(x') \le dr$ then $x' \leftarrow x''$; else $u \leftarrow |u - 1|$.
   (c) $q \leftarrow q + 1$.
4. $x'' \leftarrow x'$

Figure 2. Steps of the local search.

neighbourhood structures is often two (Liao and Cheng 2007), which is followed by our algorithm for each function (shake and local search). The two neighbourhoods employed in our algorithm are defined below:

(1) *Insertion*: randomly identifies two particular operations and places one operation in the position that directly precedes the other operation.
(2) *Swap*: randomly identifies two particular operations and places each operation in the position previously occupied by the other operation.

In this paper, imposing more diversification, the local search is developed as a threshold accepting method (Bouffard and Ferland 2007) based on the two mentioned NSs. Contrary to simple hill climbing, in a threshold accepting method, that is an iterative procedure, $x' \leftarrow x''$ when $f(x'') - f(x') \le \mathrm{d}r$, where d$r$ is an acceptance level. In this paper, the value of dr is a predetermined constant in all iterations. The local search function steps are illustrated in Figure 2.

2.3 *Artificial neural networks*

The fundamental units of an ANN are the neurons which are arranged in layers and are categorised as the input (I), hidden (H) and output (O) neurons depending on in which layer they are located. Neurons in each layer are linked to each of those in the layers immediately next to it through the connections known as synapses. Each of the synapses is characterised by a weight factor which can be adjusted to target the desired output signal. A back propagation neural network is adopted in this study in which signals are passed from the input layer to the output layer through a hidden layer and learning is done by adjusting the connection weights by an algorithm that involves back propagating the error to previous layers.

The procedure of ANN modelling is usually within the following contents: (a) choosing the parameters of ANN; (b) collecting of data; (c) pre-processing of database; (d) training of ANN; and (e) simulation and prediction by using the trained ANN. The probable effective characteristics for a learning sample on an optimisation process (input values) can be number of jobs (n), mean initial solution cost (MISC), mean operation processing time (MOPT), and operation processing time variance (OPTV). So any learning sample has four characteristics and ANN consists of four neurons x_i, $i = 1, 2, 3, 4$. The number of neurons in the hidden layer is supposed r, the output of which is categorised as y_j, $j = 1, 2, \ldots, r$. The output layer has three neurons which denote number of outer loop iteration (N), number of inner loop iteration ($q_{\max}$), and dr. It is assumed that the

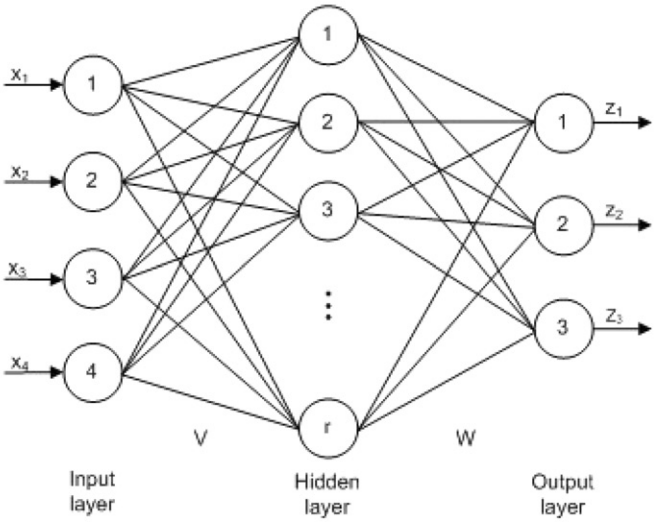

Figure 3. The structure of three layers of back propagation network.

connection weight matrix between input and hidden layers is V and the connection weight matrix between hidden and output layers is W (Figure 3). The learning phase steps are as follows:

Step 1: Choose learning coefficient (η) between 0 and 1.

Step 2: Choose initial V and W.

Step 3: Input the learning samples, and calculate the input and output net values of every neuron layer by layer, $E = 0$:

(a) Hidden layer:

$$Y_j = \sum_{i=1}^{4} x_i V_{ij}, \quad j = 1, 2, \ldots, r \tag{1}$$

$$y_j = f(Y_j), \quad j = 1, 2, \ldots, r. \tag{2}$$

(b) Output layer:

$$Z_l = \sum_{j=1}^{r} y_j W_{jl}, \quad l = 1, 2, 3. \tag{3}$$

$$z_l = f(Z_l), \quad l = 1, 2, 3. \tag{4}$$

Sigmoid function (Equation (5)) is used as transfer characteristic for every neuron in the hidden and output layers:

$$f(t) = \frac{1}{1 + \exp(-t)}, \quad t \in [-\infty, +\infty]. \tag{5}$$

Step 4: Calculate the error (E) using Equation (6):

$$E = \sum_{p=1}^{P_{\max}} \left[\frac{1}{2} \sum_{l=1}^{3} \left(d_{pl} - z_{pl} \right)^2 \right], \tag{6}$$

where: d_{pl} and z_{pl} are the desired output and actual output of ANN for the pth learning sample respectively; and $P_{\max}$ denotes the number of the learning samples.

Step 5: Calculate error signal of every neuron layer by layer using Equations (7) and (8):

$$\delta(z_l) = (d_l - z_l) z_l (1 - z_l), \quad l = 1, 2, 3 \tag{7}$$

$$\delta_j(y) = y_j (1 - y_j) \sum_{l=1}^{3} \delta_l(z) \cdot W_{jl}, \quad j = 1, 2, \ldots, r. \tag{8}$$

where $\delta(z)$ and $\delta(y)$ are error signals for output and hidden layers, respectively.

Step 6: Update the weight matrixes using Equations (9) and (10):

$$W_{jt} \leftarrow W_{jl} + \eta \delta(z_l) y_j, \quad \begin{matrix} j = 1, 2, \ldots, r \\ l = 1, 2, 3 \end{matrix} \tag{9}$$

$$V_{ij} \leftarrow V_{ij} + \eta \delta(y_j) x_i, \quad \begin{matrix} i = 1, 2, 3, 4 \\ j = 1, 2, \ldots, r. \end{matrix} \tag{10}$$

Step 7: Return to Step 3 until stopping criteria is met.

2.4 *The dynamic job shop scheduling problem*

In order to consider more reality in the job shop, some of the constraints and assumptions in the ordinary job shop scheduling problem are changed in this paper. Random job arrivals and machine breakdowns that are categorised in job related and resource related real time events, respectively, are taken into account.

In job shops, the distribution of the job arrival process closely follows a Poisson distribution. Hence, the time between arrivals of jobs is distributed exponentially (Rangsaritratsamee *et al*. 2004, Sha and Liu 2005, Vinod and Sridharan 2007). The time between two breakdowns and repair time are assumed to follow an exponential distribution too. So the mean time between failure (MTBF) and the mean time to repair (MTTR) are two parameters related to machine breakdown.

In a dynamic job shop framework, considering an event-driven policy (Sabuncuoglu and Kizilisik 2003) in which rescheduling is triggered in response to a dynamic event that changes the current condition, a static job shop problem is generated whenever a new job arrives or a machine breaks down. At that time, we consider a new job with remaining operation of previous jobs or a broken machine that needs time to repair.

The new static problem is fed to the scheduling method (VNS) to produce an optimum or a near to optimum schedule. At any rescheduling point, the parameters of the VNS are determined using weights obtained from the ANN. This optimum solution will be used as production schedule until the next event takes place. The strategy explained above is illustrated in Figure 4.

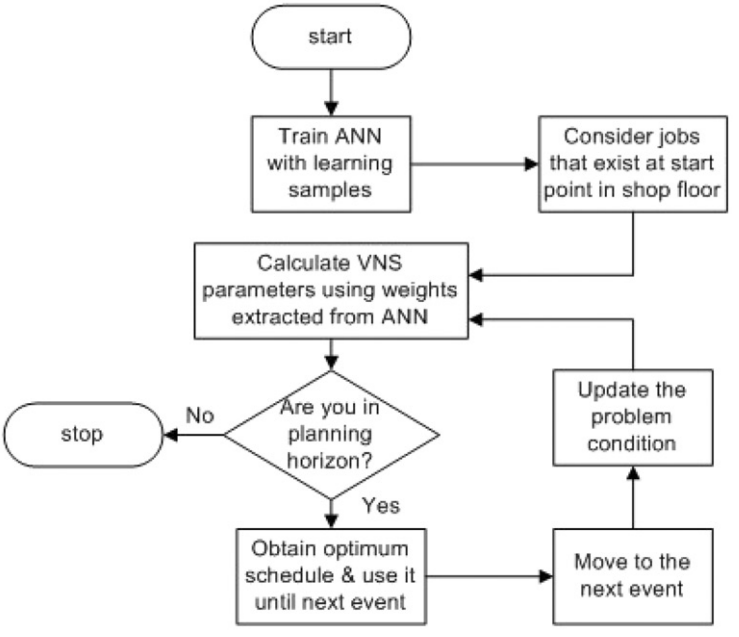

Figure 4. Proposed strategy for dynamic job shop scheduling problem.

3. Simulation study

It has been widely reported in the literature that a job shop with more than six machines presents the complexity involved in large dynamic job shop scheduling (Chryssolouris and Subramanian 2001). In this paper, to demonstrate the performance of the proposed method, a job shop consisting of 10 machines is simulated. Hence, to train an ANN, a number of learning samples with various job numbers and $m = 10$ are required. These learning samples and related desired VNS parameters obtained by experimental study are presented in Table 1.

The optimum number of neurons in the hidden layer is determined by the actual training effect. In order to find the desired one, the number of neurons is varied from three to 10 in the hidden layer. The effect of the different number of the hidden layer neurons on E is calculated 10 times and the mean values are shown in Figure 5. It is obvious that E is minimum when the number of neurons in the hidden layer is eight. During the above process, it is assumed $\eta = 0.95$.

Weights obtained from the ANN with eight neurons in the hidden layer will be used to determine the VNS parameters at any rescheduling point that is equal to a random event.

Simulation starts with a 10×10 static job shop problem (Abz6) extracted from the literature. In order to illustrate the potential of the proposed method for the DJSS problem, it is compared with some common dispatching rules that have been widely used in the literature (Pierreval and Mebabki 1997, Holthaus 1999, Qi *et al*. 2000, Dominic *et al*. 2004, Sha and Liu 2005). A list of these rules is as follows:

(1) The shortest processing time (SPT) dispatching rule.
(2) The first in first out (FIFO) dispatching rule.
(3) The last in first out (LIFO) dispatching rule.

Table 1. Learning samples and desired output used to train ANN.

		Learning sample characteristics				Desired output		
		n	MOPT	OPTV	MISC	N	q_{max}	dr
Learning sample	Amn1*	5	53.44	801.63	682.30	20	100	4
	Amn2*	5	54.50	680.62	701.12	20	100	4
	Orb1	10	54.09	823.07	1129.40	25	150	8
	Abz6	10	59.46	541.48	1017.80	25	150	6
	La21	15	53.29	709.85	1183.10	75	200	6
	La22	15	48.81	781.80	1041.60	75	200	5
	La26	20	52.57	668.17	1300.60	150	300	4
	La27	20	54.16	695.26	1290.50	150	300	6
	La32	30	55.23	756.39	1817.70	200	400	6
	La33	30	49.89	740.84	1666.50	200	400	5

Note: *Generated by the authors.

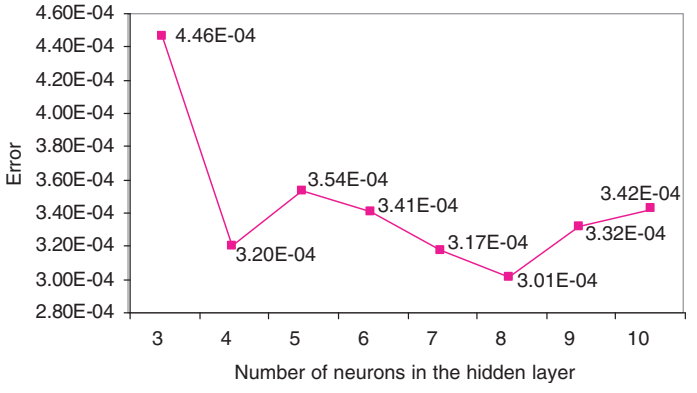

Figure 5. Effect of number of neurons in hidden layer on error.

All machines have the same MTTR and MTBF. $A_g = \text{MTTR}/(\text{MTBF} + \text{MTTR})$ denotes the breakdown level of the shop floor that is the percentage of time the machines have failures. In this article, it is assumed $A_g = 0.05$ and MTTR = 300 time units ($5 \times$ MOPT). So MTBF = 5700 is obtained. Thus, on an average of 5700 time units a machine is available and then breaks down with a mean time to repair of 300 time units. Decreasing mean time between job arrivals (MTBJA) and considering constant values of MTBF and MTTR leads to an increase in the number of jobs that concurrently exist in the shop floor at the rescheduling point. So simulation has repeated at three states of MTBJA to do a better comparison. The states are presented in Table 2.

Simulation for each state continues until the number of new jobs that have arrived at the shop floor reaches 1000. Mean flow time of all jobs that leave the shop floor during the planning horizon is selected as the performance measure. Results are illustrated in Figure 6.

As illustrated in Figure 6, the presented method for the DJSS problem dramatically improves the performance measure. Furthermore, when decreasing MTBJA that leads to

Table 2. Values of different dynamic parameters.

		Dynamic parameter (time units)		
		MTBJA	MTBF	MTTR
State	S1	400	5700	300
	S2	300	5700	300
	S3	200	5700	300

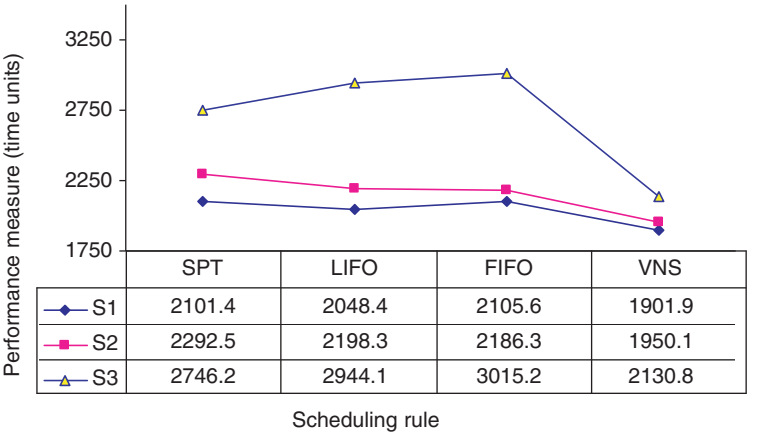

Figure 6. Comparison of the scheduling rules and the proposed method based on VNS.

an increase in the number of jobs in the shop floor at a rescheduling point, more improvement in the performance measure is obtained.

4. Conclusion

In this paper a scheduling method based on variable neighbourhood search (VNS) was proposed for the dynamic job shop scheduling problem with random job arrivals and machine breakdowns. At any rescheduling point, using weights obtained from an artificial neural network, proper parameters for VNS are calculated that significantly enhance the performance of the scheduling method. The proposed method was compared to some common dispatching rules using a simulated job shop under varied conditions. Results indicate that the performance of the proposed method is significantly better than those of the common dispatching rules.

References

Amirthagadeswaran, K.S. and Arunachalam, V.P., 2006. Improved solutions for job shop scheduling problems through genetic algorithm with a different method of schedule deduction. *International Journal of Advanced Manufacturing Technology*, 28 (5–6), 532–540.

Aydin, M.E. and Oztemel, E., 2000. Dynamic job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems*, 33 (2), 169–178.

Bose, N.K. and Liang, P., 1996. *Neural network fundamentals with graph, algorithms, and application*. New Delhi, India: Tata McGraw Hill.

Bouffard, Y. and Ferland, J.A., 2007. Improving simulated annealing with variable neighborhood search to solve the resource-constrained scheduling problem. *Journal of Scheduling*, 10 (6), 375–386.

Chryssolouris, G. and Subramanian, E., 2001. Dynamic scheduling of manufacturing job shops using genetic algorithm. *Journal of Intelligent Manufacturing*, 12 (3), 281–293.

Dominic, P.D.D., Kaliyamoorthy, S., and Kumar, M., 2004. Efficient dispatching rules for dynamic job shop scheduling. *International Journal of Advanced Manufacturing Technology*, 24 (1–2), 70–75.

Hansen, P., Mladenovic, N., and Moreno Perez, J.A., 2007. Variable neighborhood search. *European Journal of Operational Research*, 191 (3), 593–595.

Holthaus, O., 1999. Scheduling in job shops with machine breakdowns: an experimental study. *Computers & Industrial Engineering*, 36 (1), 137–62.

Liao, C.J. and Cheng, C.C., 2007. A variable neighborhood search for minimizing single machine weighted earliness and tardiness with common due date. *Computers & Industrial Engineering*, 52 (4), 404–413.

Liu, S.Q., Ong, H.L., and Ng, K.M., 2005. Metaheuristics for minimizing the makespan of the dynamic shop scheduling problem. *Advances in Engineering Software*, 36 (3), 199–205.

Mladenovic, N. and Hansen, P., 1997. Variable neighborhood search. *Computers & Operation Research*, 24 (11), 1097–1100.

Perez, J.A.M., Vega, J.M.M., and Martin, I.R., 2003. Variable neighborhood tabu search and its application to the median cycle problem. *European Journal of Operational Research*, 151 (2), 365–378.

Pierreval, H. and Mebabki, N., 1997. Dynamic selection of dispatching rules for manufacturing system scheduling. *International Journal of Production Research*, 35 (6), 1575–1591.

Qi, J.G., Burns, G.R., and Harrison, D.K., 2000. The application of parallel multipopulation genetic algorithms to dynamic job-shop scheduling. *International Journal of Advanced Manufacturing Technology*, 16 (8), 609–615.

Rangsaritratsamee, R., Ferrell, W.G., and Kurz, M.B., 2004. Dynamic rescheduling that simultaneously considers efficiency and stability. *Computers & Industrial Engineering*, 46 (1), 1–15.

Sabuncuoglu, I. and Kizilisik, O.B., 2003. Reactive scheduling in a dynamic and stochastic FMS environment. *International Journal of Production Research*, 41 (17), 4211–4231.

Sha, D.Y. and Liu, C.H., 2005. Using data mining for due date assignment in a dynamic job shop environment. *International Journal of Advanced Manufacturing Technology*, 25 (11–12), 1164–1174.

Vinod, V. and Sridharan, R., 2007. Scheduling a dynamic job shop production system with sequence-dependent setups: an experimental study. *Robotics and Computer – Integrated Manufacturing*, 24 (3), 435–449.

Yating, W., *et al*., 2008. Artificial neural network modeling of plating rate and phosphorus content in the coatings of electroless nickel plating. *Journal of Materials Processing Technology*, 205 (1–3), 207–213.

Zhou, R., Nee, A.Y.C. and Lee, H.P., 2008. Performance of an ant colony optimisation algorithm in dynamic job shop scheduling problems. *International Journal of Production Research*, doi: 10.1080/00207540701644219. [In press].