Expert Systems with Applications 37 (2010) 282–287

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Multi-objective scheduling of dynamic job shop using variable neighborhood search

M.A. Adibi [a], M. Zandieh [b,*], M. Amiri [c]

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

ARTICLE INFO

Keywords:
Dynamic job shop
Multi-objective scheduling
Variable neighborhood search
Artificial neural networks

ABSTRACT

Dynamic job shop scheduling that considers random job arrivals and machine breakdowns is studied in this paper. Considering an event driven policy rescheduling, is triggered in response to dynamic events by variable neighborhood search (VNS). A trained artificial neural network (ANN) updates parameters of VNS at any rescheduling point. Also, a multi-objective performance measure is applied as objective function that consists of makespan and tardiness. The proposed method is compared with some common dispatching rules that have widely used in the literature for dynamic job shop scheduling problem. Results illustrate the high effectiveness and efficiency of the proposed method in a variety of shop floor conditions.

1. Introduction

The job shop scheduling (JSS) problem has attracted many optimization methods because it still exists in most of manufacturing systems in various forms. JSS problem is well-known to be NP-hard and various methods like mathematical techniques, dispatching rules, artificial intelligence, artificial neural networks, neighborhood searches, fuzzy logic, and etc. are introduced to obtain an optimum (or a near to optimum) solution. But these methods are usually designed to address static JSS problem and real time events such as random job arrivals and machine breakdowns are ignored. Tacking into account these events, JSS problem shifts to a new kind of problem that is well-known as dynamic job shop scheduling (DJSS) problem. In 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 feasible longer. So in addition to scheduling problem, it is needed to deal with dynamic events in DJSS problems.

In DJSS problem, due to changes in problem condition during planning horizon, using a scheduling method with parameters set at their optimum value as used in most researches (Chryssolouris & Subramanian, 2001; Dominic, Kaliyamoorthy, & Saravana Kumar, 2004; Qi, Burns, & Harrison, 2000; Zhou, Nee, & Lee, 2008) can reduce performance of the selected method. Preventing such this problem, Aydin and Oztemel (2000) used reinforcement learning agents to select proper rule for scheduling according to the shop floor conditions in real time. Shugang, Zhiming, and Xiaohong (2005) have used a heuristic obtained by training a neural network offline with the genetic algorithm. Sha and Liu (2005a, 2005b) 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. Zandieh and Adibi (2009) used artificial neural network (ANN) to estimate proper parameters of their scheduling method at any rescheduling point for minimizing mean flow time. ANN models have been studied since early 1980s with an objective of achieving human like performance in many fields of science and are intended for modeling the organizational principles of nervous system (Bose & 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 scheduling method at rescheduling point using a pattern extracted from learning sample. Compared with the traditional pattern recognition, ANN can provide an exact description for multidimensional and non-linear problems (Yating, Bin, Lei, & Wenbin, 2008).

In this study, to address multi-objective DJSS problem, variable neighborhood search (VNS) (Mladenovic & 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, and etc. (Zandieh & Adibi, 2009; Hansen, Mladenovic, & Moreno Perez, 2007). VNS is a new neighborhood search metaheuristic that has widely used to combinatorial optimization problems in recent years. In this paper, to enhance the efficiency and effectiveness of

* Corresponding author.
E-mail address: m_zandieh@sbu.ac.ir (M. Zandieh).

0957-4174/$ - see front matter © 2009 Published by Elsevier Ltd.
doi:10.1016/j.eswa.2009.05.001

VNS, its parameters are updated at any rescheduling point by ANN (Zandieh & Adibi, 2009). Multi-objective performance measure that contains both makespan and total tardiness is also applied in the scheduling process.

The rest of the paper is organized as follows: in Section 2, the multi-objective job shop scheduling problem is defined in details. Variable neighborhood search algorithm is argued in Section 3. Artificial neural network is presented in Section 4. The dynamic job shop scheduling problem is defined in Section 5. Simulation study is presented in Section 6 and conclusion is located in Section 7.

2. The multi-objective job shop scheduling problem

The ordinary job shop scheduling model considers n jobs to be process on m machines ($n \times m$ operations) while minimizing some function of completion time of jobs subject to following technological constraints and assumptions;

- Each machine can perform only one operation at a time on any job.
- An operation of a job can be performed by only one machine at a time.
- Once an operation has begun on a machine, it must not be interrupted.
- An operation of a job cannot be performed until its preceding operations are completed.
- There are no alternate routings, i.e. an operation of a job can be performed by only one type of machine.
- Operation processing time and number of operable machines are known in advance.

In this research, a multi-objective performance measure is applied as the objective function to construct schedules. The objective function consists of makespan and tardiness. Makespan is defined as the total time that is required to process a group of jobs. Tardiness is defined as the difference between the completion time and due date for each job in which the completion time occurs after the due date. It is observed that the variance of the makespan is much smaller than that of tardiness and noted that this can have a detrimental effect on the quality of solutions (Rangsaritratsamee, Ferrell, & Kurz, 2004). So Eq. (1) is used to be minimized in this research.

$$f = 5 \times Makespan + 2 \times Tardiness = 5[\underset{alli}{Max}(d_i) - \underset{alli}{Min}(s_i)] + 2\sum_{alli} \Psi_i(d_i - DD_i), \quad (1)$$

where d_i, departure time of job i; DD_i, due date time of job i; s_i, starting time of job i; $\Psi_i = 1$ if $(d_i - DD_i) > 0; = 0$ otherwise.

Operation-based representation and deduction of schedule (Amirthagadeswaran & 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 totally nine operations, but three different integers, each is 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.

3. Variable neighborhood search

Variable neighborhood search is a new local search technique that tries to escape from local optimum by changing the neighborhood structure (NS) that is the manner in which the neighborhood is defined. In its basic form, VNS explores a set of neighborhoods of the current solution, makes a local search from a neighbor solution to a local optimum, and moves to it only if there has been an improvement (Perez, Vega, & Martin, 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', respectively. The outer loop works as a refresher reiterating the inner loop. Local search explores for an improved solution within the local neighborhood, while shake diversifies the solution by switching to another local neighborhood. 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 are illustrated in Fig. 1.

In Fig. 1, $N_k(k = 1, 2 \ldots, k_{max})$ denote neighborhood structures for both shake and local search functions. To avoid costing too much computational time, the best number of neighborhood structure is often two (Liao & Cheng, 2007) which is followed by our algorithm for each function (shake and local search). The two neighborhoods employed in our algorithm are *Insertion* and *Swap*. *Insertion* randomly identifies two particular operations and places one operation in the position that directly precedes the other operation. *Swap* randomly identifies two particular operations and places each operation in the position previously occupied by the other operation.

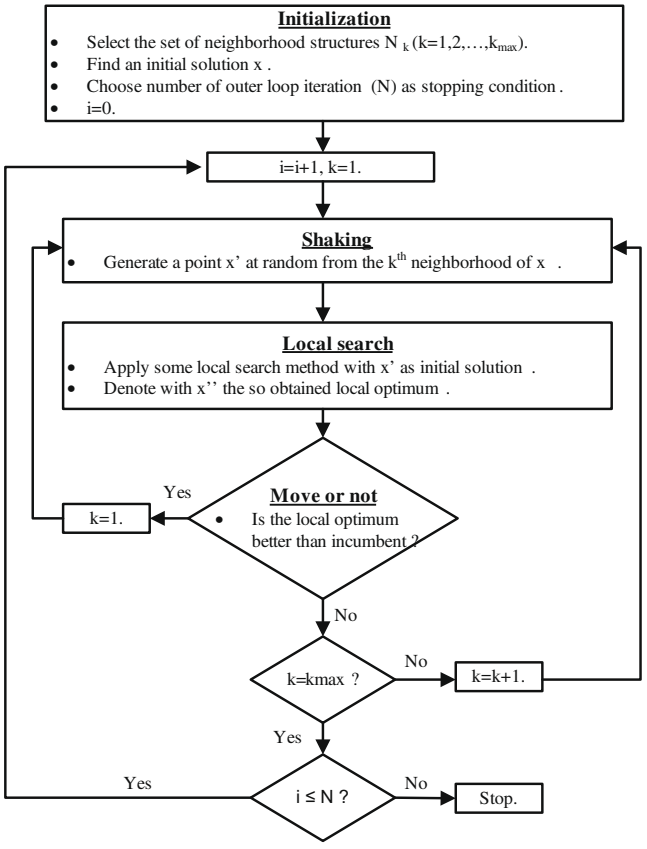

Fig. 1. VNS algorithm.

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

4. Artificial neural network

The fundamental unit of ANN is the neurons which are arranged in layers and are categorized 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 connections known as synapses. Each of synapses is characterized 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 modeling 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, (e) simulation and prediction by using the trained ANN.

In this paper, probable effective characteristics for a learning sample on optimization process (input values) can be number of jobs (n), mean initial solution cost ($MISC$), mean operation processing time ($MOPT$), operation processing time variance ($OPTV$), and mean due date (MDD). Since any learning sample has five characteristics, ANN consists of five neurons in input layer ($x_i, \quad i = 1,2,3,4,5$). The number of neurons in the hidden layer is supposed r, the output of which is categorized as $y_j, \quad j = 1,2,\ldots,r$. The output layer has three neurons ($z_l, l = 1,2,3$) which denote number of outer loop iteration (N), number of inner loop iteration ($q_{\max}$), and dr. It is assumed that the connection weight matrix between input and hidden layers is V and the connection weight matrix between hidden and output layers is W (Fig. 3).

The learning phase steps in the proposed ANN are as follows:

- Step 1: Choose learning coefficient (η).
- Step 2: Choose beginning 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:

$$Y_j = \sum_{i=1}^{5} x_i V_{ij}; \quad j = 1,2,\ldots,r, \tag{2}$$

$$y_j = f(Y_j); \quad j = 1,2,\ldots,r, \tag{3}$$

  (b) Output layer:

$$Z_l = \sum_{j=1}^{r} y_j W_{jl}; \quad l = 1,2,3, \tag{4}$$

$$z_l = f(Z_l); \quad l = 1,2,3. \tag{5}$$

  Sigmoid function (Eq. (6)) is used as transfer characteristic for every neuron in the hidden and output layers:

$$f(t) = \frac{1}{1+\exp(-t)}; \quad t \in [-\infty, +\infty]. \tag{6}$$

- Step 4: Calculate the error (E) using Eq. (7):

$$E = \sum_{p=1}^{P\max} \left[\frac{1}{2} \sum_{l=1}^{3} (d_{pl} - z_{pl})^2 \right], \tag{7}$$

  where d_{pl} and z_{pl} are the desired output and actual output of ANN for the p^{th} learning sample, respectively. $P_{\max}$ denotes the number of the learning samples.
- Step 5: Calculate error signal of every neuron layer by layer using Eqs. (8) and (9).

$$\delta_l(z) = (d_l - z_l) z_l (1 - z_l); \quad l = 1,2,3 \tag{8}$$

$$\delta_j(y) = y_j (1 - y_j) \sum_{l=1}^{3} \delta_l(z).W_{jl}; \quad j = 1,2,\ldots,r \tag{9}$$

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

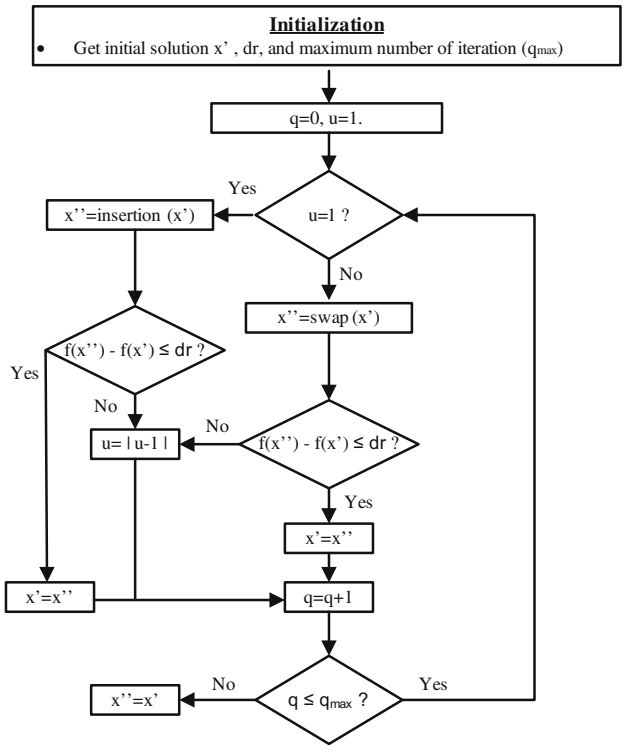

Fig. 2. Local search algorithm.

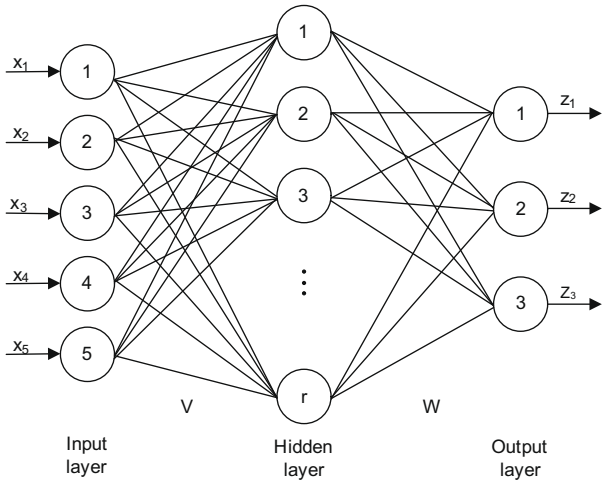

Fig. 3. The structure of a three-layer back propagation network.

- Step 6: update the weight matrixes using Eqs. (10) and (11).

$$W_{jl} \leftarrow W_{jl} + \eta\delta(z_l)y_j; \quad j = 1,2,\ldots,r, \quad l = 1,2,3 \tag{10}$$

$$V_{ij} \leftarrow V_{ij} + \eta\delta(y_j)x_i; \quad i = 1,2,3,4,5, \quad j = 1,2,\ldots,r. \tag{11}$$

- Step 7: Turn to step 3 until stopping condition (number of iteration) is met.

5. The dynamic job shop scheduling

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

In the job shop, the distribution of the job arrival process closely follows the Poisson distribution. Hence, the time between arrivals of jobs is distributed exponentially (Rangsaritratsamee et al., 2004; Sha & Liu, 2005a, 2005b; Vinoda & Sridharan, 2007) and mean time between job arrivals (*MTBJA*) controls new job arrival rate. 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 dynamic job shop framework, considering an event-driven policy (Sabuncuoglu & 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 breakdowns. At that time, we consider a new job with remained 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 VNS are determined using weights obtained from ANN. This optimum solution will be used as production schedule until next event takes place. The strategy explained above is illustrated in Fig. 4.

6. Simulation study

It has been widely reported in literature that a job shop with more than 6 machines presents the complexity involved in large dynamic job shop scheduling (Chryssolouris & Subramanian, 2001). In this paper, to demonstrate the performance of the proposed method, a job shop consist of 10 machines is simulated. Hence, to train 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 total work content (TWK) rule is selected to generate due date for each job. Using TWK rule, the due date of each job equals the sum of the job arrival time and a multiple of the total job processing time. The multiple is related to job characteristics and is called the tightness factor. Hence, the due date of job i is computed as Eq. (12).

$$DD_i = a_i + K\sum_{j=1}^{m} t_{ij}; \quad i = 1,2,\ldots,n, \tag{12}$$

where a_i is arrival time of job i and K is the tightness factor. In this research, the tightness factor is assigned from a normal distribution with a mean of 10. Furthermore, since it is recommended that the minimum value of the tightness factor be 2, the standard deviation of the distribution is set to 2 so meeting this minimum value is assured with near certainty (Rangsaritratsamee et al., 2004).

The optimum number of neurons in hidden layer is determined by the actual training effect. In order to find the desired value, the number of neurons in hidden layer is varied from 5 to 15. The effect of the different number of the hidden layer neurons on E is calculated 10 times at each value and the mean values are shown in Fig. 5. It is obvious that the E is minimum when the number of neurons in the hidden layer is 13. In this process, η is assumed 0.9 and number of iteration is selected 100000 as stopping condition.

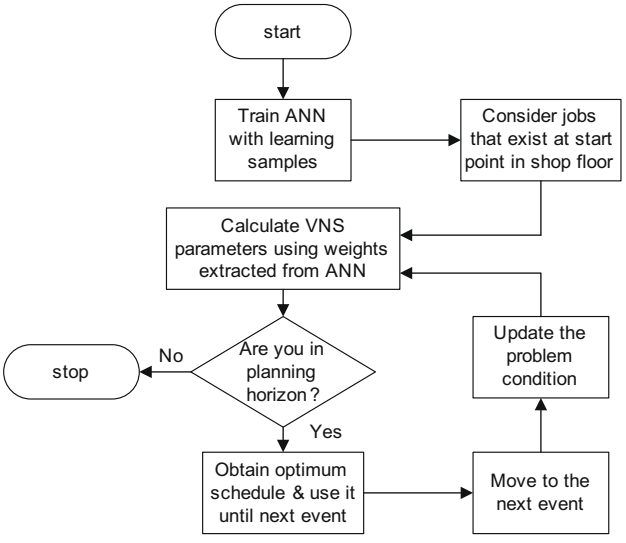

Fig. 4. Proposed strategy for dynamic job shop scheduling problem.

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

	Learning sample characteristics					Desired output		
	n	*MOPT*	*OPTV*	*MISC*	*MDD*	*N*	q_{max}	*dr*
Learning sample								
Amn1*	5	51.94	915.07	4093.0	5289.0	25	100	40
Amn2*	5	54.50	680.62	4431.0	5650.0	25	100	35
Orb1	10	54.09	823.07	7550.0	5137.0	50	200	50
Abz6	10	59.46	541.48	6970.0	4440.0	50	200	40
La21	15	53.29	709.85	7946.8	4679.0	100	250	65
La22	15	48.81	781.80	7286.2	4615.7	100	250	60
La26	20	52.57	668.17	8939.8	6533.8	150	300	55
La27	20	54.16	695.26	9036.0	5462.7	150	300	65
La32	30	55.23	756.39	13652.6	6351.3	200	400	80
La33	30	49.89	740.84	12000.6	4371.8	200	400	60

* Generated by the authors.

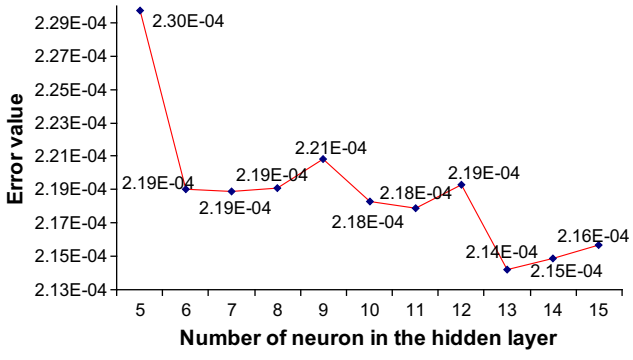

Fig. 5. Effect of number of neuron in the hidden layer on error value, $\eta = 0.9$, number of iteration = 100,000.

Weights obtained from ANN with 13 neurons in hidden layer will be used to determine 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 literature. In order to illustrate the potential of the proposed method for DJSS problem, it is compared with some common dispatching rules that widely used in literature (Pierreval & Mebabki, 1997; Holthaus, 1999; Qi et al., 2000; Dominic et al., 2004; Sha & Liu, 2005a, 2005b). A list of these rules is as follows:

- The shortest processing time (SPT) dispatching rule.
- The first in first out (FIFO) dispatching rule.
- The last in first out (LIFO) dispatching rule.
- The earliest due date (EDD) dispatching rule.

All machines have the same *MTTR* and *MTBF*. $A_g = MTTR/(MTBF + MTTR)$ 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 (approximately $5 \times MOPT$). 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 *MTBJA* and considering constant values of *MTBF* and *MTTR* leads to an increase in 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 job arrived at shop floor reaches 1200. Mean value of objective function for the latest 1000 rescheduling points during planning horizon is selected as the performance measure of each scheduling method. Results are illustrated in Fig. 6.

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

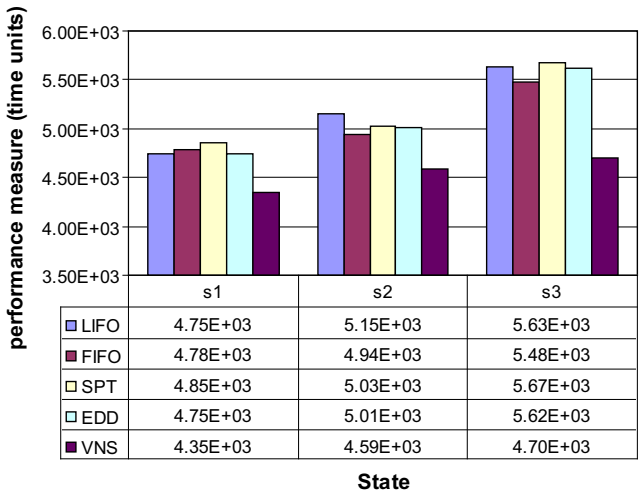

Fig. 6. Comparison of the scheduling rules and the proposed method based on VNS.

As illustrated in Fig. 6, presented method for DJSS problem dramatically improves performance measure. Furthermore, decreasing *MTBJA* that leads to an increase in number of jobs in the shop floor at rescheduling point, more improvement in performance measure is obtained.

7. Conclusion

In this research a scheduling method based on variable neighborhood search (VNS) is proposed for dynamic job shop scheduling problem with random job arrivals and machine breakdowns. Multi-objective performance measure that contains both makespan and total tardiness is also applied in the scheduling process. At any rescheduling point, using weights obtained from artificial neural network, proper parameters for VNS are calculated that significantly enhances 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., & 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*, 532–540.

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

Bose, N. K., & Liang, P. (1996). *Neural network fundamentals with graph, algorithms, and application*. Tata McGraw Hill.

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

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

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

Hansen, P., Mladenovic, N., & 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*, 137–162.

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

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

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

Pierreval, H., & 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., & Harrison, D. K. (2000). The application of parallel multipopulation genetic algorithms to dynamic job-shop scheduling. *International Journal of Advanced Manufacturing Technology, 16*, 609–615.

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

Sabuncuoglu, I., & 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., & Liu, C. H. (2005a). Using data mining for due date assignment in a dynamic job shop environment. *International Journal of Advanced Manufacturing Technology, 25*, 1164–1174.

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

Shugang, L., Zhiming, W., & Xiaohong, P. (2005). Job shop scheduling in real time cases. *International Journal of Advanced Manufacturing Technology, 26*, 870–875.

Vinoda, V., & Sridharan, R. (2007). Scheduling a dynamic job shop production system with sequence-dependent setups: An experimental study. *Robotics and Computer-Integrated Manufacturing*, doi:10.1016/j.rcim.2007.05.001.

Yating, W., Bin, S., Lei, L., & Wenbin, H. (2008). Artificial neural network modeling of plating rate and phosphorus content in the coatings of electroless nickel plating. *Journal of Materials Processing Technology, 205*, 207–213.

Zandieh, M., Adibi, M. A. (2009). Dynamic job shop scheduling using variable neighbourhood search. *International Journal of Production Research*, doi:10.1080/00207540802662896.

Zhou, R., Nee, A. Y. C. & 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.