Computers & Industrial Engineering 110 (2017) 75–82

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

A reinforcement learning approach to parameter estimation in dynamic job shop scheduling

Jamal Shahrabi [a,*], Mohammad Amin Adibi [b], Masoud Mahootchi [a]

[a] Faculty of Industrial Engineering and Management Systems, Amirkabir University of Technology, Tehran, Iran
[b] Faculty of Industrial and Mechanical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran

ARTICLE INFO

Article history:
Received 24 May 2013
Received in revised form 27 June 2016
Accepted 24 May 2017
Available online 26 May 2017

Keywords:
Reinforcement learning
Q–factor
Dynamic job shop scheduling
Variable neighborhood search

ABSTRACT

In this paper, reinforcement learning (RL) with a Q-factor algorithm is used to enhance performance of the scheduling method proposed for dynamic job shop scheduling (DJSS) problem which considers random job arrivals and machine breakdowns. In fact, parameters of an optimization process at any rescheduling point are selected by continually improving policy which comes from RL. The scheduling method is based on variable neighborhood search (VNS) which is introduced to address the DJSS problem. A new approach is also introduced to calculate reward values in learning processes based on quality of selected parameters. The proposed method is compared with general variable neighborhood search and some common dispatching rules that have been widely used in the literature for the DJSS problem. Results illustrate the high performance of the proposed method in a simulated environment.

1. Introduction

Selecting appropriate scheduling method or optimization parameters in the dynamic job shop scheduling (DJSS) has been noted by many researchers in recent years (Liu & Hsu, 2015; Nguyen, Zhang, Johnston, & Tan, 2015; Park et al., 2016; Scholz-Reiter, Hildebrandt, & Tan, 2013), because dynamic environment of problem due to changes in shop floor condition, causes a scheduling method not to be the best all over the scheduling horizon.

The DJSS problem emerges when real time events such as random job arrivals and machine breakdowns are taken into account in the ordinary JSS problem. In the DJSS problem, one or more conditions of the job shop scheduling (JSS) problem like the number of jobs or the number of operable machines are changed by a random event. In addition to the scheduling problem of DJSS, it is necessary to address two new issues; "when" and "how" to react to dynamic events in DJSS problems.

In some researches published regarding the DJSS problem, a scheduling method with predefined parameters is used at any rescheduling point (Chryssolouris & Subramanian, 2001; Dominic, Kaliyamoorthy, & Saravana, 2004; Fatemi Ghomi & Iranpoor, 2010; Liu, Ong, & Ng, 2005; Qi, Burns, & Harrison, 2000; Wang, Xiao, Li, & Wang, 2010; Yang, Yu-si, & Hong-yn, 2009; Zhou, Nee, & Lee, 2009). However, due to changes in problem conditions during the planning horizon, using a scheduling method with fixed parameters can reduce the performance of the method. Preventing such a problem, Aydin and Oztemel (2000) used reinforcement learning agents to select appropriate dispatching rules for scheduling according to the shop floor conditions in real time. Wang and Usher (2005) used reinforcement learning agents to select an appropriate dispatching rule for a single machine dynamic scheduling problem.

Although their work is a successful use of RL in DJSS, dispatching rules do not have good quality to meet optimization objectives. Sha and Liu (2005) presented a model that incorporates a data mining tool for mining the knowledge of job scheduling regarding 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.

Adibi, Zandieh, and Amiri (2010) used a trained artificial neural network (ANN) to update parameters of a metaheuristic method at any rescheduling point in a DJSS problem according to the problem condition. In their proposed method, offline training was used and no adaptation was taken during the scheduling horizon. Chen, Hao, Lin, and Murata (2010) proposed a rule driven dispatching method based on data envelopment analysis and reinforcement learning for the multi objective dynamic scheduling problem. Vinod and Sridharan (2011) studied the effects of due date assignment methods and scheduling rules on the performance of a job shop produc-

* Corresponding author at: 424 Hafez Ave., Tehran 1591634311, Iran.
E-mail addresses: jamalshahrabi@aut.ac.ir (J. Shahrabi), adibi@qiau.ac.ir (M.A. Adibi), mmahootchi@gmail.com (M. Mahootchi).

http://dx.doi.org/10.1016/j.cie.2017.05.026
0360-8352/

tion system in a dynamic environment. Kapanoglu and Alikalfa (2011) introduced an unsupervised learning scheduling system that builds state and priority rule pairs depending on intervals of queue by using a rule based GA for the DJSS problem. Adibi and Shahrabi (2014) presented a clustering-based modified variable neighborhood search algorithm for DJSS problem. They used K-means clustering method to improve local search in the meta-heuristic method. Oktaviandri, Hassan, and Shaharoun (2016) developed a decision support tool based on promising artificial intelligent that was able to use appropriately dispatching rules for DJSS problem.

Obviously, reinforcement learning techniques in which agents have to take into account reinforcement signals against their actions, are widely employed to address DJSS problem in which learning is required to its solve methods. However, it has been used only to select appropriate scheduling method.

On the other hand, Q-learning is the most well-known reinforcement learning algorithm which is used in literature. It works in such a way that the agent gains experience by trial and error throughout the execution of the actions. During this process, the agent learns how to assign credit or blame to each of its actions in order to improve its behavior (Aydin & Oztemel, 2000).

In this study, 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, and generality (Hansen, Mladenovic, & Moreno Perez, 2007). To enhance the efficiency and effectiveness of VNS, its parameters are updated at any rescheduling point by RL. Using few effective parameters during the optimization process, VNS is a more suitable scheduling method because using a scheduling method with fewer parameters decreases attempts to train agents in the learning process in the new so called RLVNS approach. In fact, the proposed method for DJSS problem or RLVNS uses RL to determine value of effective parameters of the VNS according to problem situation dynamically. This determination causes an automatic coordination between problem situation and the VNS setting over time. The proposed method is tested in a challenging simulated environment and it is compared to some benchmark methods reported in the literature for DJSS problem.

The rest of the paper is organized as follows: In Section 2 the dynamic job shop scheduling problem is defined in detail and then VNS is explained in Section 3. Reinforcement learning and Q-factor algorithm is presented in Section 4. A framework for the proposed method is introduced in Section 5. Simulation results are presented in Section 6 and the conclusion is given in Section 7.

2. Dynamic job shop scheduling problem

In the general job shop scheduling problem, n jobs should be processed on m machines while minimizing a function of completion time of jobs is considered along with the following technological constraints and assumptions:

(1) each machine can perform only one operation at a time on any job,
(2) an operation of a job can be performed by only one machine at a time,
(3) once an operation has begun on a machine it must not be interrupted,
(4) an operation of a job cannot be performed until its preceding operations are completed,

Initialization: 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 $k \leftarrow k+1$;

Fig. 1. Steps of the basic VNS.

(5) there are no alternate routings, i.e. an operation of a job can be performed by only one type of machine, and
(6) operation processing time and number of operable machines are known in advance.

In the dynamic job shop scheduling problem, some of the constraints and assumptions in the ordinary job shop scheduling problem are changed in order to consider more reality in job shop scheduling problem. So, the following time-dependent mathematical model can be proposed to dynamic job shop scheduling problem based on the mathematical model of classic job shop scheduling presented by Blazewicz, Domschke, and Pesch (1996). In the proposed model which is expressed in Eqs. (1)–(4), t represents the time in which a real time event takes place. In that time, the parameters of the model will vary. The set of parameters includes number of jobs, number of operable machines, processing time for operations, and Precedence relationships among operations. So, the optimum schedule will vary also at time t. In the mathematical model, $V^t = \{0, 1, \ldots, n^t, n^t + 1\}$ denotes the set of operations at time t. Here, 0 and $n^t + 1$ are dummy operations representing "start" and "end". M^t denotes the set of m^t machines A^t is the set of ordered pairs of operations constrained by the precedence relations for each job at time t. E_k^t is the set of all pairs of operations to be performed on machine k at time t. p_i^t is processing time of operation i. c_i^t is also optimization variable which specifies start time of operation i since time t. Therefore, considering deterministic processing time for the operations, the objective function in Eq. (1) assures minimizing mean flow time measure. Constraints in Eq. (2) guarantee consideration of the processing sequence of operations in the jobs and those in Eq. (3) ensure that there is only one job on each machine at an instance.

$$\min \frac{1}{n^t}\sum_{i=1}^{n^t+1} c_i^t \tag{1}$$

s.t.

$$c_j^t - c_i^t \geqslant p_i^t; \quad \forall (i,j) \in A^t \tag{2}$$

$$c_j^t - c_i^t \geqslant p_i^t \text{ or } c_i^t - c_j^t \geqslant p_j^t; \quad \forall (i,j) \in E_k^t, \ \forall k \in M^t \tag{3}$$

$$c_i^t \geqslant 0; \quad \forall i \in V^t \tag{4}$$

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.

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 $f(x'') - f(x') \leq dr$ then $x' \leftarrow x''$; else $u \leftarrow |u - 1|$.
   (c) $q \leftarrow q + 1$.
4. $x'' \leftarrow x'$

Fig. 2. Steps of local search.

1. Set $Q(i,a) = 0, V(i,a) = 0; \forall i \in S, a \in A(i)$.
2. Set $k = 0$, k_{max}, $A = constant$.
3. Determine initial state i.
4. Select action a according to a selection strategy at state i.
5. Execute action a and calculate corresponding reward $r(i,a,j)$ and do following updates: $V(i,a) \leftarrow V(i,a) + 1$, $\alpha = \frac{A}{V(i,a)}$.
6. Update Q-factor associate to state i and action a as: $Q(i,a) \leftarrow (1-\alpha)Q(i,a) + \alpha[r(i,a,j) + \lambda max_{b \in A(j)} Q(j,b)]$.
7. Set $k = k + 1$, $i = j$. If $\leq k_{max}$, return to the step 4, else go to the step 8.
8. Determine optimum decision at each state $a^*(i)$ as: $a^*(i) \in \arg max_{b \in A(j)} Q(j,b)$.

Fig. 3. Steps of Q-learning algorithm.

Table 1
Defined states and their interpretations.

State	Number of jobs in the shop floor (n)	Mean Operations Processing Time (MOPT)
1	$2 \leqslant n \leqslant 3$	$MOTP < 25$
2	$2 \leqslant n \leqslant 3$	$25 \leqslant MOTP < 50$
3	$2 \leqslant n \leqslant 3$	$50 \leqslant MOTP < 75$
4	$2 \leqslant n \leqslant 3$	$MOTP \geqslant 75$
5	$4 \leqslant n \leqslant 6$	$MOTP < 25$
6	$4 \leqslant n \leqslant 6$	$25 \leqslant MOTP < 50$
7	$4 \leqslant n \leqslant 6$	$50 \leqslant MOTP < 75$
8	$4 \leqslant n \leqslant 6$	$MOTP \geqslant 75$
9	$7 \leqslant n \leqslant 10$	$MOTP < 25$
10	$7 \leqslant n \leqslant 10$	$25 \leqslant MOTP < 50$
11	$7 \leqslant n \leqslant 10$	$50 \leqslant MOTP < 75$
12	$7 \leqslant n \leqslant 10$	$MOTP \geqslant 75$
13	$11 \leqslant n \leqslant 15$	$MOTP < 25$
14	$11 \leqslant n \leqslant 15$	$25 \leqslant MOTP < 50$
15	$11 \leqslant n \leqslant 15$	$50 \leqslant MOTP < 75$
16	$11 \leqslant n \leqslant 15$	$MOTP \geqslant 75$
17	$n \geqslant 16$	$MOTP < 25$
18	$n \geqslant 16$	$25 \leqslant MOTP < 50$
19	$n \geqslant 16$	$50 \leqslant MOTP < 75$
20	$n \geqslant 16$	$MOTP \geqslant 75$

Table 2
Set of actions and corresponding VNS parameters.

Action	VNS parameters
1	$N = 10$, $q_{max} = 50$, $dr = 4$
2	$N = 10$, $q_{max} = 50$, $dr = 6$
3	$N = 20$, $q_{max} = 50$, $dr = 4$
4	$N = 20$, $q_{max} = 50$, $dr = 6$
5	$N = 50$, $q_{max} = 50$, $dr = 4$
6	$N = 50$, $q_{max} = 50$, $dr = 6$
7	$N = 100$, $q_{max} = 50$, $dr = 4$
8	$N = 100$, $q_{max} = 50$, $dr = 6$

3. Variable neighborhood search

The Variable neighborhood search algorithm is a metaheuristic optimization method that can be used to solve combinatorial problems. The algorithm searches the neighborhood of a solution until another solution better than the current solution is found and then moves to the new one.

An operation based representation method (Amirthagadeswaran & Arunachalam, 2006) is also selected in this paper. This representation method encodes a schedule as a sequence of numbers. Each number stands for one operation. The specific operation represented by a number is interpreted according to the order of numbers in the string. Each of the n numbers, representing n jobs, will appear m times spread over the entire string.

For instance, consider matrix $[123;231;321]$ as sequence of jobs 1, 2, and 3 on machines 1, 2, and 3. We are given a solution like [213233112], where $\{1,2,3\}$ represents $\{job_1, job_2, job_3\}$ respectively. Obviously, there are a total of nine operations, but three different integers are repeated three times each. 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 [213233112] 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 the ith job.

VNS tries to escape from the local optimum by changing the neighborhood structure (NS) that is the manner in which the neighborhood is defined. Neighborhoods are usually ranked in such a way that solutions increasingly far from the current one are explored (Perez, vaga, & 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 structure. 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 it until the termination condition is met. The steps of the VNS structure can be summarized as Fig. 1. In this figure, $N_k, k = 1, 2, \ldots, k_{max}$, denote neighborhood structures for both shake and local search functions. To avoid taking up too much computational time, the best number of the 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 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 & Ferland, 2007) based on the two mentioned neighborhood structures. Con-

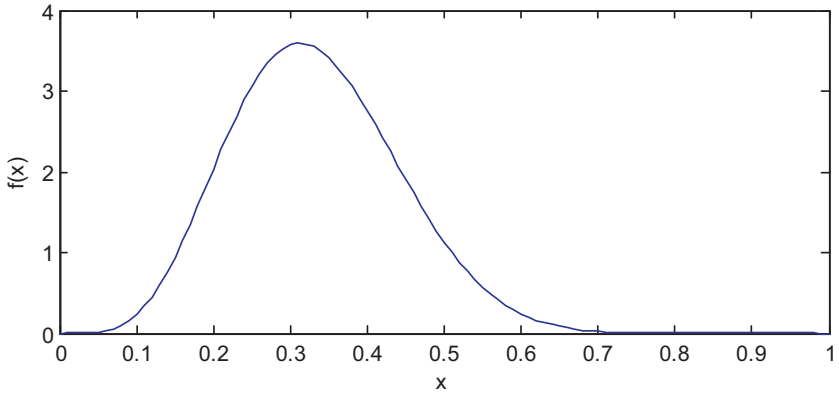

Fig. 4. The curve corresponding to Eq. (5), $u = 6, w = 12$.

Table 3
Conversion rules corresponding to reward and punishment.

$f(x_0)$	Corresponding reward
$f(x_0) \leqslant 0.1$	-1
$0.1 < f(x_0) \leqslant 1.0$	0
$1 < f(x_0)$	1

trary to simple hill climbing, in the threshold accepting method, which is an iterative procedure, $x' \leftarrow x''$ when $f(x'') - f(x') \leqslant dr$, where dr is an acceptance level. The value of dr is a predetermined constant in all iterations. Local search function steps are illustrated in Fig. 2.

To use the VNS, some parameters including N, q_{max}, and dr should be initially specified. Value of each parameter has significant effect on the performance of the VNS. On the other hand, the best value for each parameter depends on the problem which is being solved. In the next section, reinforcement learning is introduced which will be used to determine appropriate value for the parameters of the VNS according to the scheduling problem over time.

4. Reinforcement learning

4.1. Q-Learning

Q-learning, one of the most popular algorithms of reinforcement learning, which is used in this paper, is presented in this section. In the Q-learning algorithm, the goal is to find state–action pair value, $Q(i,a)$, which represents the long–term expected reward for each pair of state and action. The optimal state–action values for a system, proven to converge to the optimal state–action values, represent the optimal policy that an agent intends to learn. The used procedure of the Q-learning algorithm is as shown in Fig. 3 (Wang & Usher, 2005). In this figure, $Q(i,a)$ is state–action pair value, $V(i,a)$ is the number of selecting action a at state i. S is set of states and $A(i)$ is set of all admissible actions at state i. k_{max} is maximum number of learning iteration that is equal to the number of new jobs arrived at shop in this paper. A is selected as 1 and λ is discounting rate set 0.8 in this paper. In this paper, ε-greedy is used as the selection strategy. In ε-greedy method, if a random number is less than the predefined value, ε, an action will be selected at random from admissible action set at state i, otherwise the action with the most Q-factor will be selected (Gosavi, 2003).

4.1.1. State and action definition

In the proposed approach, state is defined as the shop floor condition influencing the scheduling problem optimization performance represented by two variables: number of job in shop floor (n) and mean processing time of current operations ($MOPT$). MOPT is simply calculated based on processing time of all operations which are ready at the scheduling time. All possible conditions are classified as 20 states which are listed in Table 1. Eight actions are also defined. Each action contains VNS algorithm parameters including number of outer loop iteration (N), number of inner loop iteration (q_{max}) and acceptance threshold (dr). All these actions and their different values are presented in Table 2.

4.1.2. Reward

At any rescheduling point, according to problem conditions, a state will be identified. Then the RL agent, based on selection strategy, selects an action that determines VNS operational parameters. After the optimization process, a vector containing objective function values during the last optimization effort will be obtained. Plotting the matrix, a decreasing curve will appear. Appearance of the curve can give a lot of information about the optimization

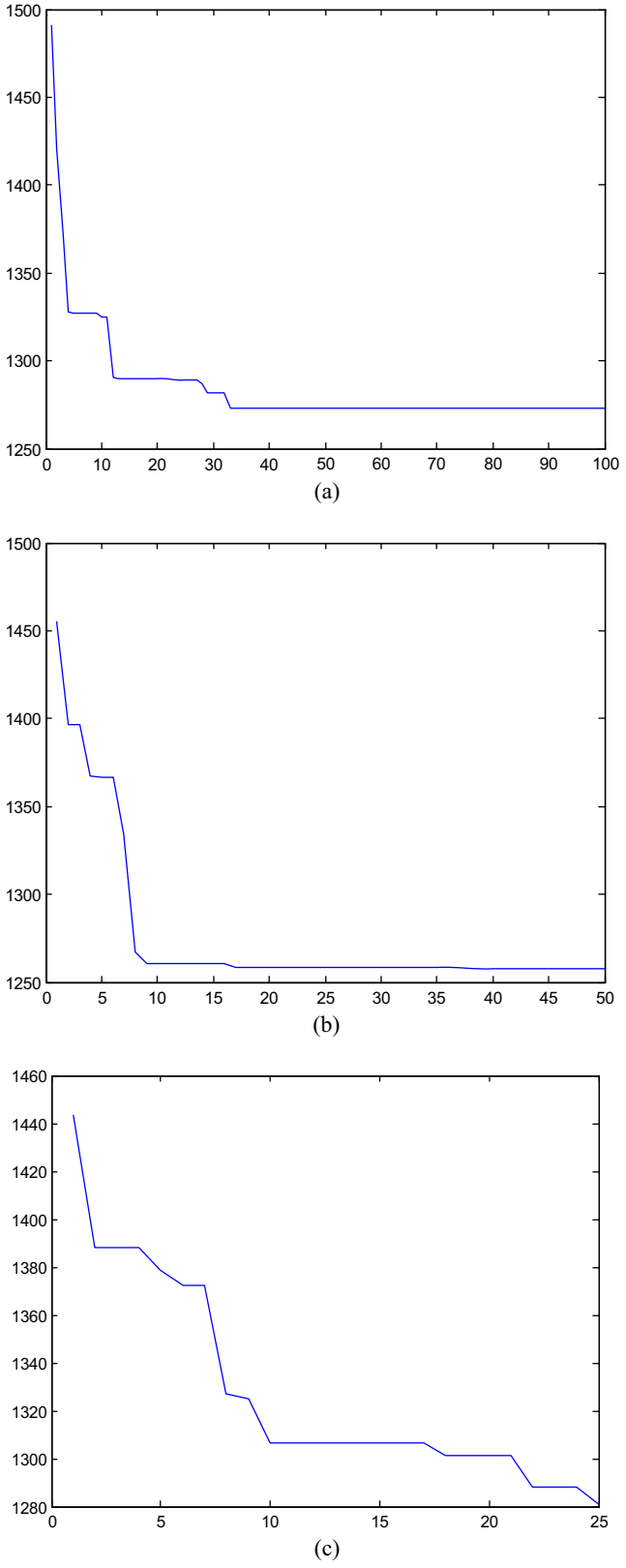

Fig. 5. The curves corresponding to three different sets of VNS operational parameters. 3(a): $N = 100$, $q_{max} = 20$, $dr = 5$, $N_{min} = 34$, 3(b): $N = 50$, $q_{max} = 20$, $dr = 5$, $N_{min} = 37$, 3(c): $N = 25$, $q_{max} = 20$, $dr = 5$, $N_{min} = 24$.

process goodness which highly depends on VNS algorithm parameters.

Particularly, a reasonable decrease that gets flat near the end of the iteration number represents a good parameter selection. On the other hand, a steep decrease that gets flat immediately indicates that excessive search is done and also a decreasing curve that ends very near to the iteration number (N) represents a few number of executed searches. In this paper −1, 0, and +1 are considered as reward when the curve appearance is bad, not good, and good, respectively. To do so in this paper, a bell shape function is used as presented in Fig. 4. This function can be obtained by a special combination of Eq. (5) parameters, u and w.

$$f(x) = \frac{1}{\beta(u,w)} x^{u-1}(1-x)^{w-1}; \quad 0 < x < 1 \tag{5}$$

In Eq. (5), $\beta(u,w)$ is defined as Eq. (6).

$$\beta(u,w) = \int_0^1 x^{u-1}(1-x)^{w-1}dx \tag{6}$$

For $u = 6$ and $w = 12$, the shape is illustrated in Fig 4. In the proposed approach $x = 1 - N_{min}/N$ in which N is number of iteration and N_{min} ($N_{min} < N$) is the number that the curve gets flat at it. As illustrated in Fig. 4, the curve is not symmetric. This is because getting a good solution is more important than CPU time reduction. Note that other combinations of values for u and w can be used to produce such shape. To convert a continuous value of $f(x_0)$, $0 < x_0 < 1$, to discrete values −1, 0, and +1, the conversion rules presented in Table 3 are used.

For example, consider three sets of VNS parameters that yield the curves illustrated in Fig 5. In Table 4, steps of reward calculation for each curve are presented. All curves are related to optimization processes of a $m = 10$ and $n = 10$ JSS problem.

5. The proposed framework

In the 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 (state), a static job shop problem is generated whenever a new job arrives or a machine breaks down. At that time we consider the new job and the remaining operation of previous jobs or the broken machine that needs time to be repaired.

The new static problem is fed to the scheduling method (VNS) to produce an optimum or a near to optimum schedule. At any

Table 4
Steps of reward calculation for the curves in Fig. 5.

Curve	N, q_{max}, dr	N_{min}	x	$f(x)$	Reward
3(a)	100, 20, 5	34	0.66	0.0653	−1
3(b)	50, 20, 5	37	0.26	3.2148	1
3(c)	25, 20, 5	24	0.04	0.0049	−1

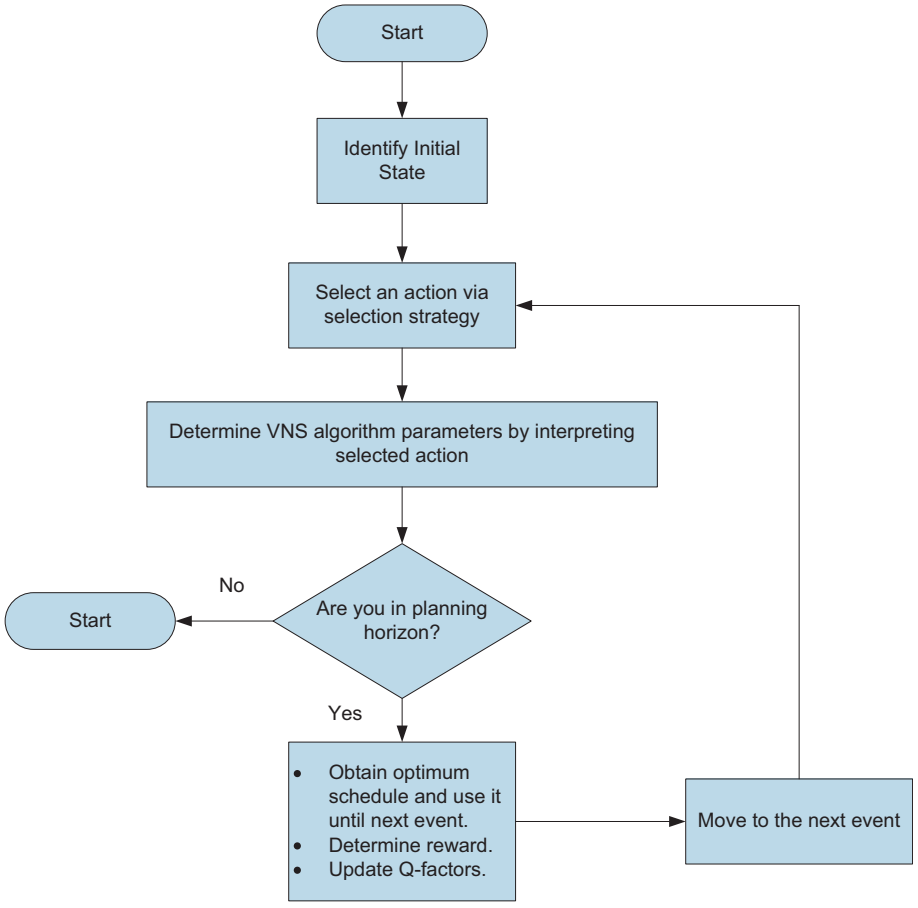

Fig. 6. Proposed strategy for dynamic job shop scheduling problem using RL to estimate optimization method parameters.

rescheduling point, the parameters of VNS are determined using interpreting selected action by the RL agent. This optimum solution will be used as a production schedule until the next event takes place.

A reward or punishment is determined at the end of each rescheduling and Q-factors are updated. The strategy explained above is illustrated in Fig. 6.

6. Simulation results

In job shops, the distribution of the job arrival process is usually assumed to closely follow the Poisson distribution. Hence, the time between arrivals of jobs is distributed exponentially (Adibi et al., 2010; Rangsaritratsamee, Ferrell, & Kurz, 2004; Sha & Liu, 2005; Vinod & Sridharan, 2007). The time between two breakdowns and repair time are also assumed to follow an exponential distribution. Thus, the mean time between failure (MTBF) and the mean time to repair (MTTR) are two parameters related to machine breakdown.

It has been widely reported in the 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 consisting of 10 machines is simulated.

To determine the VNS parameters at any rescheduling point that is equal to a state, ε - greedy strategy is used to select the proper action. When an action is selected, the corresponding VNS algorithm parameters are obtained. Thus, a static JSS problem is ready to be solved by the VNS algorithm.

Simulation starts with Abz6 10×10 static job shop problem extracted from the literature (Adams, Balas, & Zawack, 1988). In order to illustrate the potential of the proposed method for the DJSS problem, it is compared with some common dispatching rules that are widely used in the literature (Adibi et al., 2010; Dominic et al., 2004; Holthaus, 1999; Pierreval & Mebabki, 1997; Qi et al., 2000; Sha & Liu, 2005). 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 proposed method is also compared to the general VNS (GVNS). In GVNS all parameters are assumed fixed during the planning horizon. In order to make a fair comparison, GVNS parameters are selected so that the mean CPU time over all jobs is the same as the proposed method RLVNS.

It is assumed that all machines have the same mean time to repair (MTTR) and mean time between failures (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, $A_g = 0.05$ and $MTTR = 5 \times MOPT = 300$ time units are assumed. So $MTBF = 5700$ is obtained. Thus an average of 5700 time units of a machine is available and then breaks down with a mean time to repair 300 time units.

Simulation continues until the number of new jobs arrived at the shop floor reaches 2200. Mean flow time of the last 2000 jobs that leave the shop floor during the planning horizon is selected as the performance measure. Results are illustrated in Fig. 7.

As illustrated in Fig. 7, the presented method for the DJSS problem improves the performance measure. Although dispatching rules have less CPU time because of their simplicity, they produce low quality solution as is presented in Fig. 7. Using the RL permits appropriate selection of values for the VNS parameters over time. In fact, it guarantees necessary and sufficient computational attempts for each VNS run.

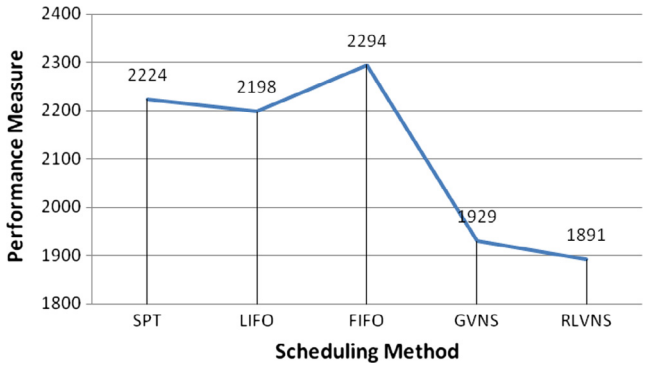

Fig. 7. Comparison of proposed method based on RL and other common scheduling methods for DJSS problem.

7. Conclusion

In this paper, a scheduling method based on variable neighborhood search (VNS) is proposed for the dynamic job shop scheduling problem with random job arrivals and machine breakdowns. Considering an event driven policy, to obtain proper parameters for the VNS at any rescheduling point, reinforcement learning with the Q-learning algorithm is used. In the proposed method, an action is selected using the ε - greedy strategy according to shop floor condition which are defined as a state. This approach significantly enhances the performance of the scheduling method. The proposed method was compared to some common dispatching rules and general VNS using a simulated job shop. Results indicate that the performance of the proposed method is significantly better than those of the common dispatching rules and GVNS. The most significant advantage of the proposed method is its ability to update optimum strategies in the form of Q-factor during time. Thus, the proposed method is not only a solution to the dynamic nature of the real production environment, but optimum strategies are also updated dynamically.

References

Adams, J., Balas, E., & Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management Science, 34*(3), 391–401.

Adibi, M. A., & Shahrabi, J. (2014). A clustering-based modified variable neighborhood search algorithm for a dynamic job shop scheduling problem. *The International Journal of Advanced Manufacturing Technology, 70*(9–12), 1955–1961.

Adibi, M. A., Zandieh, M., & Amiri, M. (2010). Multi-objective scheduling of dynamic job shop using variable neighborhood search. *Expert Systems with Applications, 37*, 282–287.

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.

Blazewicz, J., Domschke, W., & Pesch, E. (1996). The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research, 93*(1), 1–33.

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.

Chen, X., Hao, X., Lin, H. W., & Murata, T. (2010). Rule driven multi objective dynamic scheduling by data envelopment analysis and reinforcement learning. In *IEEE international conference on automation and logistics* (Art no. 5585316, pp. 396–401).

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.

Fatemi Ghomi, S. M. T., & Iranpoor, M. (2010). Earliness – tardiness – lost sales dynamic job – shop scheduling. *Production Engineering Research Development, 4* (2), 221–230.

Gosavi, A. (2003). *Simulation-based optimization: Parametric optimization technique and reinforcement learning*. Kluwer Academic Publishers.

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.

Kapanoglu, M., & Alikalfa, M. (2011). Learning IF – THEN priority rules for dynamic job shops using genetic algorithms. *Robotics and Computer-Integrated Manufacturing, 27*(1), 47–55.

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.

Liu, C. H., & Hsu, C. I. (2015). Dynamic job shop scheduling with fixed interval deliveries. *Production Engineering, 9*(3), 377–391.

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

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

Nguyen, S., Zhang, M., Johnston, M., & Tan, K. C. (2015). Automatic programming via iterated local search for dynamic job shop scheduling. *IEEE Transactions on Cybernetics, 45*(1), 1–14.

Oktaviandri, M., Hassan, A., & Shaharoun, A. M. (2016), Generation of look-up tables for dynamic job shop scheduling decision support tool. In *IOP conference series: materials science and engineering, February* (Vol. 114(1), p. 012067).

Park, J., Mei, Y., Nguyen, S., Chen, G., Johnston, M. & Zhang, M. (2016). Genetic programming based hyper-heuristics for dynamic job shop scheduling: cooperative coevolutionary approaches. In *European conference on genetic programming. March* (pp. 115–132).

Perez, J. A. M., vaga, 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.

Scholz-Reiter, B., Hildebrandt, T., & Tan, Y. (2013). Effective and efficient scheduling of dynamic job shops—combining the shifting bottleneck procedure with variable neighbourhood search. *CIRP Annals - Manufacturing Technology*. http://dx.doi.org/10.1016/j.cirp.2013.03.047.

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

Vinod, V., & 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.

Vinod, V., & Sridharan, R. (2011). Simulation modeling and analysis of due – data assignment methods and scheduling decision rules in a dynamic job shop production system. *International Journal of Production Economics, 129*(1), 127–146.

Wang, S., Xiao, X., Li, F., & Wang, C. (2010). Applied research of improved hybrid discrete PSO for dynamic job – shop scheduling problem. In *IEEE proceedings of the 8th world congress on intelligent control and automation* (Art no. 5553799, pp. 4065–4068).

Wang, Y., & Usher, J. M. (2005). Application of reinforcement learning for agent–based production scheduling. *Engineering Applications of Artificial Intelligence, 18*, 73–82.

Yang, G., Yu-si, D., & Hong-yn, Z. (2009). Job–shop scheduling considering rescheduling in uncertain dynamic environment. In *International conference on management science and engineering – 16th annual conference proceedings, ICMSE* (Art. no. 5317409, pp. 380–384).

Zhou, R., Nee, A. Y. C., & Lee, H. P. (2009). Performance of an ant colony optimization algorithm in dynamic job shop scheduling problems. *International Journal of production Research, 47*(11), 2903–2920.