Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational Design and Engineering 4 (2017) 46–59

www.elsevier.com/locate/jcde

A single-machine scheduling problem with multiple unavailability constraints: A mathematical model and an enhanced variable neighborhood search approach

Maziar Yazdani[a,*], Seyed Mohammad Khalili[a], Mahla Babagolzadeh[b], Fariborz Jolai[a,1]

[a]*School of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran*
[b]*Department of Industrial Engineering, Ferdowsi University of Mashhad, Mashhad, Iran*

Received 22 January 2016; received in revised form 5 July 2016; accepted 9 August 2016
Available online 3 October 2016

Abstract

This research focuses on a scheduling problem with multiple unavailability periods and distinct due dates. The objective is to minimize the sum of maximum earliness and tardiness of jobs. In order to optimize the problem exactly a mathematical model is proposed. However due to computational difficulties for large instances of the considered problem a modified variable neighborhood search (VNS) is developed. In basic VNS, the searching process to achieve to global optimum or near global optimum solution is totally random, and it is known as one of the weaknesses of this algorithm. To tackle this weakness, a VNS algorithm is combined with a knowledge module. In the proposed VNS, knowledge module extracts the knowledge of good solution and save them in memory and feed it back to the algorithm during the search process. Computational results show that the proposed algorithm is efficient and effective.

Keywords: Single machine scheduling; Availability constraint; Variable neighborhood search; Knowledge module

1. Introduction

Generally in scheduling problems it is assumed that machines are continuously available over the planning horizon. However, this assumption may not be true in many practical situations. For instance, a machine may not be available during the planning horizon due to maintenance activities [1], tool changes [2], or breakdowns. Since, machines require preventive and curative maintenance [3], operators take breaks, and worn out parts require changing. Not only managers are increasingly faced with the costs caused by the temporary unavailability of resources, but also they are constantly concerned with difficult decisions regarding balancing resources' unavailability and production. Nowadays, accounting for machine unavailability has become a promising research area. For instance, in the airline industry, scheduled maintenance has reduced production time by about 15% [4]. Low et al. [5] mentioned two applications related to the aerospace industry where micro drilling tools need to be changed periodically and the machine cannot be used during this time. They have emphasized the wide applicability of the problem in real manufacturing environments such as computer centers, NC-machines and IC-testing industries. Rapine et al. [6] considered the case of an automated machine which requires the intervention of an auxiliary resource (i.e., an operator that removes jobs or adds chemicals) whose unavailability blocks the machine. Accordingly, managers have to schedule their machines effectively in order to maximize their profits while avoiding conflicts between scheduled maintenance and planned production.

In this paper, we study a scheduling problem on a single machine with multiple unavailability periods and distinct due

*Corresponding author. Fax: +98 21 88013102.
E-mail addresses: Maziyar.yazdani@ut.ac.ir (M. Yazdani), M.Khalili@ut.ac.ir (S.M. Khalili), Mahla_golzade@yahoo.com (M. Babagolzadeh).
[1]fjolai.ut.ac.ir
Peer review under responsibility of Society for Computational Design and Engineering.

http://dx.doi.org/10.1016/j.jcde.2016.08.001

dates where the objective is to minimize the sum of maximum earliness and tardiness of jobs. It is shown that this problem is strongly NP-hard, and in order to optimize the problem exactly, a mathematical model is proposed. In addition, with respect to high complexity of this problem, metaheuristic algorithms are proposed to obtain optimal or near-optimal solutions for the considered problem. Variable neighborhood search (VNS) is a powerful metaheuristic algorithm for solving complex combinatorial optimization problems which was first introduced by Mladenovic [7]. VNS, through using a systematic change of neighborhood structures, is capable of evading local optimum. Therefore during past decade, it has been used for solving a wide spectrum of complex optimization problems such as graph coloring [8], spanning tree [9] and job shop scheduling [10]. However, it should be mentioned that VNS and other metaheuristic algorithms have some weaknesses. For instance, they guide their optimization algorithm through utilizing objective function or fitness [11]. Random nature of these algorithm's operators is another weakness of them [12]. To overcome this shortcoming, recent researches [13–17] concentrated on proposing algorithms that have emphasized on the interaction between evolution and learning. This paper intends to combine VNS algorithm with a knowledge module and proposes a knowledge-based variable neighborhood search.

The paper has the following structure. In the next section, a brief review of relevant literature is provided. Section 3 presents the mathematical formulation of the considered problem. In Section 4, we describe the proposed algorithm. Section 5 reports the experimental design. Finally, last section is devoted to conclusion and future research.

2. Literature review

As mentioned former, this paper deals with a single machine scheduling problem with multiple unavailability periods where the objective function minimizes the sum of maximum earliness and tardiness of jobs, in addition, a knowledge-based VNS is proposed as the solution method. Accordingly, the relevant literature is provided in three separate but complementary streams: single machine scheduling problems with unavailability constraints, machine scheduling problems with the focus on the maximum earliness and tardiness of jobs, and recent applications of VNS in scheduling problems.

2.1. Single machine scheduling problems with unavailability constraints

A comprehensive review of literature in scheduling problems with unavailability constraints has been conducted by Schmidt [18]. Angel-Bello et al. [19] proposed a mixed integer programming model for scheduling problem with availability constraints and sequence-dependent setup costs. Moreover, they presented a valid inequality and an efficient heuristic approach in order to lessen the computational time. Rustogi and Strusevich [20] considered single machine problems with generalized positional deterioration effects and machine maintenance where decisions are made regarding possible sequences of jobs and on the number of maintenance activities to be included into a schedule in order to minimize the overall makespan. Zammori et al. [21] focused on the single machine scheduling problem, in which jobs and maintenance tasks are jointly considered to find the optimal schedule. Wang and Liu [22] presented an integrated optimization model for production scheduling and preventive maintenance (PM) in a single machine with its time to failure has a Weibull probability distribution. Yin et al. [23] considered the problem of scheduling of independent and simultaneously available jobs without preemption on a single machine, where the machine has a fixed maintenance activity. Xu et al. [24] considered a single-machine scheduling problem with workload-dependent maintenance duration, and the objective is minimize total completion time. Cui et al. [25] addressed the problem of finding robust production and maintenance schedules for a single machine with failure uncertainty, where both production and maintenance activities occupy the machine's capacity, while production depletes the machine's reliability and maintenance restores its reliability. Luo et al. [26] considered the problem of scheduling a maintenance activity and jobs on a single machine, where the maintenance activity must start before a given deadline and the maintenance duration increases with its starting time. Hfaiedh et al. [27] aimed to minimize the maximum delivery time under the non-resumable scenario of jobs in a single machine scheduling problem with release dates and tails, provided that the machine is unavailable during a fixed interval. Bai et al. [28] studied a single machine slack due date assignment (usually referred to as SLK) scheduling problem with deteriorating jobs and a rate-modifying activity, where the deterioration effect manifest such that the job processing time is a function of its starting time in a sequence. Vahedi-Nouri et al. [29] considered a single machine scheduling problem with the learning effect and multiple availability constraints that minimizes the total completion time. Li and Zhao [30] studied single machine scheduling with a fixed non-availability interval, where the processing time of a job is a linear increasing function of its starting time, and each job has a release date. Kacem et al. [31] considered the maximization of the weighted number of early jobs on a single machine with non-availability constraints. They dealt with the resumable and the non-resumable cases. Gu et al. [32] investigated two single-machine scheduling problems with a new type of aging effect, which is dominated by the processing speed of the machine, while during the whole scheduling horizon, the machine is subject to an optional maintenance, and the duration of the maintenance depends on the length of the uptime before it. Liu et al. [33] investigated a single-machine scheduling problem with periodic maintenance, in which the pursued objective is to minimize the number of tardy jobs. Wang [34] proposed a bi-objective optimization model for the problem of production scheduling and preventive maintenance in a single-machine context with sequence-dependent setup times, while during the setup times, preventive maintenance activities are supposed to be performed simultaneously. The two objectives are to minimize the total expected completion time of jobs and to minimize the maximum of expected times

of failure of the machine at the same time. Chen et al. [35] considered the single machine scheduling problem with an operator non-availability period. The supposed operator non-availability period is an open time interval in which a job may neither start nor complete. Fan et al. [36] studied the problem of integrated scheduling of production and delivery on a single machine in which jobs in processing may be interrupted due to the availability constraint of the machine. They supposed when the machine becomes available again, the interrupted jobs can be resumed or restarted processing. Mashkani and Moslehi [37] presented a novel definition for single machine scheduling problem with flexible periodic availability constraints called bimodal availability in which, the duration of unavailability corresponding to the continuous working time of the machine changes in a discrete manner and it can adopt two different values. Nian and Mao [38] addressed three deterioration effect models for the single-machine scheduling problems with simultaneous considerations of job rejection, deterioration effects, and deteriorating multi-maintenance activities. They assumed that each machine may be subject to several maintenance activities over the scheduling horizon, and the duration of the maintenance depends on its running time. Hsu [39] explored a single-machine scheduling with aging effects and optional maintenance activity assignment. The jobs' processing time is assumed to follow a power position-dependent aging model. This study decided whether and when to implement the maintenance activity into the job sequence, how long the duration of maintenance activity is, and how to schedule so as to minimize the makespan.

2.2. Scheduling problem with sum of maximum earliness and tardiness criterion

Amin-Nayeri and Moslehi [40] were among the first people who investigated how to minimize the sum of maximum earliness and tardiness in a single-machine scheduling problem. Tavakkoli-Moghaddam et al. [41] studied the same objective function by considering an idle insert algorithm. They used various size of problems for evaluating the efficiency of suggested algorithm and they presented the optimal value related to the special case of a common due date. Afterwards, Tavakkoli-Moghaddam et al. [42] applied a branch and bound algorithm in order to solve a single-machine sequencing problem to gain an optimal sequence of job in which objective function attempts to minimize the maximum earliness and tardiness. Later in another paper, a simulated annealing and a branch and bound algorithms are used to solve a single-machine scheduling problem by Tavakkoli-Moghaddam and Vasei [43]. Based on the obtained results, they illustrated that the proposed simulated annealing has smaller error and its computational time is less than that of obtained by branch and bound method. Moslehi et al. [44] presented a branch and bound approach with proper upper and lower bound in order to solve scheduling problem of two machines flow shop. In addition, they proposed some effective lemmas to increase the efficiency of the algorithm. In the sequences, a branch and bound method was extended for a single-machine scheduling problem with unequal release times [45]. In a later work, Moslehi et al. [46] developed a branch and bound algorithm for the problem by considering useful upper and lower bound and new dominances rules. Nekoiemehr and Moslehi [47] developed an efficient lower and upper bound in addition to three dominance rules for the problem with sequence-dependent setup time. Mahnam et al. [48] presented an efficient branch and bound in order to solve the scheduling problem with unequal release time and idle insert. Furthermore, they suggested two metaheuristic algorithms; namely, genetic and a particle swarm optimization, for large job sizes and the obtained results indicated that the proposed genetic algorithm was more efficient. Recently, a mixed integer linear model for a single-machine scheduling problem is presented by Benmansour et al. [49]. They studied the problem in two particular cases. In the first case they considered that the machine was always available, while in the second one, availability constraints associated with periodic maintenance are considered for the machine.

2.3. Variable neighborhood search in scheduling problems

During the past decade, solving complex optimization problems with metaheuristic algorithms has received considerable attention among researchers [50,51]. Variable Neighborhood Search (VNS) is of these metaheuristic algorithm that has been widely used for solving a broad spectrum of combinatorial and global optimization problems. Also, this algorithm provides a framework for building heuristics, which applies the idea of neighborhood change in order to avoid the trap of local minima [52]. Reviewing the literature shows that the VNS is a common solution approach in scheduling problems. For instance, Liao and Chen [53] proposed a hybrid metaheuristic that uses tabu search within VNS for minimizing the total weighted earliness and tardiness in a single machine environment. Kirlik and Oguz [54] presented a general VNS for minimizing the total weighted tardiness with sequence dependent setup times in a single machine scheduling problem. Liu and Zhou [55] elaborated a hybrid metaheuristic, named as permutation-based harmony-VNS, for solving the single-machine earliness/tardiness scheduling problem where the common due date is specified in advance and imposed as restrictive to the schedule. Arroyo et al. [56] applied VNS algorithm to solve the single machine scheduling problem with sequence dependent setup times and distinct due windows. They considered multiple objectives including minimizing the total weighted earliness/tardiness and minimizing the total flow time, simultaneously. Wang and Tang [57] developed a population-based VNS to solve the problem of single machine scheduling with the aim of minimizing total weighted tardiness. Recently, Laalaoui and M'Hallah [4] addressed a single machine scheduling problem where jobs are weighted, machine is unavailable during one or more maintenance periods, and all jobs share a common due date. They proposed a VNS based heuristic that is dotted with two mechanisms (a linked list data structure and a dynamic threshold acceptance criterion) in order to speed up its convergence toward optimal solution.

3. Problem description and model formulation

The problem under study is described in this section. There are n jobs which are processed on a single machine. The machine is not continuously available for processing throughout the scheduling horizon, and it has multiple fixed and predefined unavailability periods. M_j is the jth unavailability period. If jobs between any two consecutive unavailability periods are considered as a batch, a schedule can be viewed as batches of jobs separated by unavailability periods. The objective of the problem is to find a schedule that minimizes the sum of maximum earliness and tardiness of jobs. Assumptions made in this paper are as follows:

- Jobs are processed without error and preemption of the jobs is not allowed.
- A machine can only process one job at a time.
- Setup times are included in the processing time.
- All data in this problem is deterministic.
- There is no precedence relationship between the jobs.
- Time intervals between two consecutive maintenance activities are identical.

Property 1. Problem $1|nr\_a|ET_{max}$ is strongly NP-hard.

Proof 1. Assume in especial case, all jobs have a common due date and due dates equal to zero. So minimizing sum of maximum earliness and tardiness become minimizing makespan. Minimizing makespan subject to multiple unavailability periods and non-resemable jobs (1|nr_pm|Cmax) [58], is strongly NP-hard. Clearly, minimizing the sum of maximum earliness and tardiness is strongly NP-hard too.

Notations which are used in the developed model, are listed as below:

Indices

i Index for jobs, $i=1,\ldots, n$.
r Index for positions, $r=1,\ldots, n$.
b Index for batches, $b=1\ldots\ k+1$.

Parameters

N Number of jobs requiring processing at time zero.
p_i Processing time of job i.
K Number of unavailability periods.
d_i Due date of job i.
S_b Start time of the bth unavailability period
F_b Finish time of the bth unavailability period
M A very large number.

Decision variables

$c_{[r]}$ Completion time of job scheduled in position r.
x_{irb} A binary variable that is equal to 1 if job i is scheduled in position r and batch b in sequence, otherwise 0.
e_i Earliness of job i.
E_{max} Maximum earliness of jobs.
t_i Tardiness of job i.
T_{max} Maximum tardiness of jobs.
y_{irb} Completion time of job i if is placed in position r on batch b (an auxiliary variable for linearizing the model).
w_r A continues auxiliary variable for linearizing the model.
z_r A binary auxiliary variable for linearizing the model.

Now, we present the model:

$$\min z = E_{max} + T_{max} \tag{1}$$

S.t.

$$\sum_{r=1}^{n}\sum_{b=1}^{k+1} x_{irb} = 1; \quad \forall i = 1,2,\ldots,n \tag{2}$$

$$\sum_{i=1}^{n}\sum_{b=1}^{k+1} x_{irb} = 1; \quad \forall r = 1,2,\ldots,n \tag{3}$$

$$\sum_{i=1}^{n}\sum_{r=1}^{n} x_{irb}p_i \le (S_b - F_{b-1}); \quad \forall b = 1,2,\ldots,k+1 \tag{4}$$

$$\sum_{i=1}^{n}\sum_{r=1}^{n} x_{irb} \le M \times \sum_{i=1}^{n}\sum_{r=1}^{n} x_{ir(b-1)}.; \quad \forall b = 2,\ldots,k+1 \tag{5}$$

$$c_{[r]} \ge c_{[r-1]} + \sum_{i=1}^{n}\sum_{b=1}^{k+1} x_{irb}p_i; \quad \forall r = 1,2,\ldots,n \tag{6}$$

$$c_{[r]} \ge \sum_{i=1}^{n}\sum_{b=1}^{k+1} x_{irb}.(p_i + F_{b-1}); \quad \forall r = 1,2,\ldots,n \tag{7}$$

$$c_{[r]} \le \sum_{i=1}^{n}\sum_{b=1}^{k+1} x_{irb} \times p_i + Max\left\{c_{[r]}, \sum_{i=1}^{n}\sum_{b=1}^{k+1} x_{irb} \times F_{b-1}\right\}; \quad \forall r = 1,2,\ldots,n \tag{8}$$

$$\sum_{r=1}^{n} c_{[r]} \times x_{irb} \le S_b; \quad \forall i = 1,2,\ldots,n, \ \ b = 1,2,\ldots,k+1 \tag{9}$$

$$t_i - e_i = \sum_{r=1}^{n}\sum_{b=1}^{k+1} c_{[r]} \times x_{irb} - d_i; \quad \forall i = 1,2,\ldots,n \tag{10}$$

$$E_{max} \ge e_i; \quad \forall i = 1,2,\ldots,n \tag{11}$$

$$T_{max} \ge t_i; \quad \forall i = 1,2,\ldots,n \tag{12}$$

$$x_{irb} \in \{0,1\}; \quad i = 1,2,\ldots,N, \quad r = 1,\ldots,N, \quad b = 1,2,\ldots,k+1 \tag{13}$$

$$c_{[r]} \ge 0; \quad r = 1,\ldots,n \tag{14}$$

$$e_i \ge 0; \quad i = 1,\ldots,n \tag{15}$$

$$t_i \ge 0; \quad i = 1,\ldots,n \tag{16}$$

The objective function (1) minimizes the sum of maximum earliness and tardiness of jobs. Constraint (2) ensures that each job must be scheduled to exactly one batch and one position. Constraint (3) ensures that only one job can be scheduled to position r. Processing time for each batch is restricted by constraint (4). Constraint (5) specifies that if no job is located into one batch, then no job could be placed into the next batch.

Completion times of jobs are computed by constraints (6)–(9). Constraint (10) determines the tardiness or earliness of each job based on its completion time. Constraints (11) and (12) calculate the maximum earliness and maximum tardiness of all jobs respectively. Constraints (13)–(16) define the types of variables. Also we assume that $C_{[0]}$ and F_0 are zero.

The proposed model is a nonlinear mixed integer programming model which its nonlinearity originates from constraints (8), (9) and (10). Due to the complexities of solving nonlinear mathematical models we propose the following constraints. To remove the nonlinear term in constraint (8) we define:

$$w_r = \left(c_{[r-1]} + \sum_{i=1}^{n}\sum_{b=1}^{k+1} x_{irb}p_i\right) - \left(\sum_{i=1}^{n}\sum_{b=1}^{k+1} x_{irb}.(p_i + F_{b-1})\right); \quad \forall b = 1,2,\ldots,k+1 \tag{17}$$

where w_r is a free of sign variable. Then we should have:

$$w_r + M \times (1 - z_r) \geq 0; \quad \forall r = 1,2,\ldots,n \tag{18}$$

$$w_r \leq M \times z_r; \quad \forall r = 1,2,\ldots,n \tag{19}$$

Similarly, to remove the nonlinear term in constraints (9) and (10) we define:

$$y_{irb} = x_{irb} \times p_i; \quad \forall i = 1,2,\ldots,n, \ \ r = 1,2,\ldots,n, \ \ b = 1,2,\ldots,k+1 \tag{20}$$

Then we should have:

$$y_{irb} - M \times (1 - x_{irb}) \leq c_{[r]}; \quad \forall i = 1,2,\ldots,n, \ \ r = 1,2,\ldots,n, \ \ b = 1,2,\ldots,k+1 \tag{21}$$

$$y_{irb} + M \times (1 - x_{irb}) \geq c_{[r]}; \quad \forall i = 1,2,\ldots,n, \ \ r = 1,2,\ldots,n, \ \ b = 1,2,\ldots,k+1 \tag{22}$$

$$y_{irb} \leq M \times x_{irb}; \quad \forall i = 1,2,\ldots,n, \ \ r = 1,2,\ldots,n, \ \ b = 1,2,\ldots,k+1 \tag{23}$$

So constraints (9) and (10) will be rewritten as (24) and (25), respectively.

$$y_{irb} \leq S_b; \quad \forall i = 1,2,\ldots,n, \ \ r = 1,2,\ldots,n, \ \ b = 1,2,\ldots,k+1 \tag{24}$$

$$t_i - e_i = \sum_{r=1}^{n}\sum_{b=1}^{k+1} y_{irb} - d_i; \quad \forall i = 1,2,\ldots,n \tag{25}$$

Finally the ultimate model contains (1)–(7), (11)–(19) and (21)–(25). The ultimate model is a linear mixed integer programming model which can be solved exactly by common commercial software packages like GAMS. However due to the NP-hard nature of the problem, the ultimate linear programming model could hardly be solved in large cases. So developing an appropriate solution algorithm to solve the problem in large cases is essential which will be done in the next section.

4. Proposed algorithm approach

Variable neighborhood search starts with an initial solution, and manipulates it through a multi-nested loop in which the core one alters and explores via two main functions so-called 'shake' and 'local search'. The outer loop plays a role as a refresher reiterating the inner loop, while the inner loop carries out the major search. Local search searches for an improved solution through 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. Once an inner loop is completed, the outer loop reiterates until the termination condition is satisfied (see Fig. 1).

Since the neighborhood structures (NS) play a key role in VNS, it should be chosen very rigorously to achieve an efficient VNS. Here, we tried to propose several neighborhood structures for generating diverse solutions in the quickest and easiest way possible. The proposed neighborhood structures are listed below. In the proposed VNS a knowledge module has been added to algorithm to increase the efficiency of basic VNS.

4.1. Solution representation

A good solution representation should be simple, reduce redundancy, show an accurate solution to the problem, and also allows algorithm to work effectively. Fig. 2 shows an

Initialization. Select the set of neighborhood structures N_k, k = 1,...,k_{max}, that will be used in the search;
find an initial solution x;
choose a stopping condition;
Repeat the following until the stopping condition is met:
(1) Set k ⟵ 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′ ∈ 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) Move or not. If this local optimum is better than the incumbent, move there (x ⟵ x"), and continue the search with N_1(k ⟵ 1); otherwise, set k ⟵k+1;

Fig. 1. Steps of the basic VNS.

example of a solution representation for the problem where number of jobs is ten.

4.2. Knowledge module

Knowledge and heuristic search modules are integrated in knowledge based optimization method where The task of first one is to obtain the necessary information from the solutions and feeds it back to the search module, and the task of second one is to search through the wide solution space and find good solutions. The idea behind implantation of knowledge module in algorithm is to keep features of previous solutions and apply them to find more appropriate solutions.

4.2.1. Memory

A sample of Memory array is shown in Fig. 3. The all rows of memory are initialized by zero value. Within the searching the knowledge module recognizes which solutions are best solutions that are obtained so far. This knowledge is stored in Memory and during search process back to VNS and guides it to search in a space that contains better solutions. Memory would be update if a new and improved solution is found. This transaction between VNS and Memory continues until the stopping condition is satisfied. The Memory initialization and its updating are described in the next two subsections.

4.2.1.1. Memory initialization. At begin, there is no knowledge and all blocks in Memory are set to 0. When an initial solution is generated randomly and considered as the best solution, this solution copy to memory completely, and the remaining blocks are set to 0. Fig. 4 represents an initialized Memory for the considered problem.

4	2	5	9	6	7	1	3	8	10

Fig. 2. A sample solution representation.

Memory

6	3	7	8	5	1	2	4	9	10
2	10	8	9	1	5	7	6	3	4

Fig. 3. A sample of memory.

Memory

0	0	0	0	0	0	0	0	0	0
5	2	1	7	6	8	9	10	3	4

Initial solution

5	2	1	7	6	8	9	10	3	4

Fig. 4. A sample of memory initialization.

4.2.1.2. Updating of memory. Memory is updated after local search by the obtained best solution. Fig. 5 shows an example of updating Memory. If the memory is full and a new and improved solution is found by using searching procedure, the improved solution is substituted for the worse best solution existing in the memory. However if the memory has an empty capacity, the improved solution is located in the empty row. As shown in this figure the first two arrays are related to Memory before updating and the third array is the improved solution obtained by local search, so the worse solution existing in the memory is replaced with the third array in order to update Memory.

4.3. Neighborhood structures

Neighborhood structure, attempts to transform the current solution into one of its neighbors to generating new solution using previous one. Numerous structures are examined in our algorithm and the best neighborhood structures are selected through them. In this research six neighborhood structures LN_1 to LN_6 are used for the local search and three neighborhood structures SN_1 to SN_3 are used for the shaking procedure. Two of these neighborhood structures are knowledge-based and use Memory to generate new solution. The details of these procedures are provided in the following.

4.3.1. Neighborhood structure $\boldsymbol{LN}_1$

In this procedure, we look for possibility improvement of objective function by exchanging the position of jobs. Two jobs are selected at random, and then are swapped (see Fig. 6).

4.3.2. Neighborhood structure $\boldsymbol{LN}_2$

In this procedure, a job is removed at random, and then relocated into another random selected position (see Fig. 7).

5	2	1	7	6	8	9	10	3	4
2	10	8	9	1	5	7	6	3	4

Improved solution

2	5	3	6	10	9	1	8	4	7

Fig. 5. A sample of memory updating.

5	4	10	1	8	6	9	3	2	7

Fig. 6. Procedure of $\boldsymbol{LN}_1$.

8	9	2	10	3	7	4	1	5	6

Fig. 7. Procedure of $\boldsymbol{LN}_2$.

4.3.3. Neighborhood structure $\boldsymbol{LN}_3$

Three jobs are selected at random, and then their position changed with each other randomly (see Fig. 8).

4.3.4. Neighborhood structure $\boldsymbol{LN}_4$

Job with highest earliness is selected, and then swap with another random selected job (see Fig. 9).

4.3.5. Neighborhood structure $\boldsymbol{LN}_5$

Job with highest tardiness is selected, and then swap with another random selected job (see Fig. 10).

4.3.6. Neighborhood structure $\boldsymbol{LN}_6$

The steps for generating neighbor solutions by LN_6 are as follows (see Fig. 11):

Step 1: A job is selected from a memory at random.
Step 2: Selected job is removed from current solution.
Step 3: The selected job at first step is inserted in same position of the memory in current solution.

Step 4: Empty position that obtained in step 2 fills with job that is not in current solution.

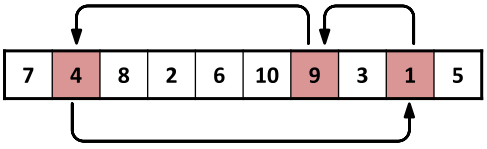

Fig. 8. Procedure of $\boldsymbol{LN}_3$.

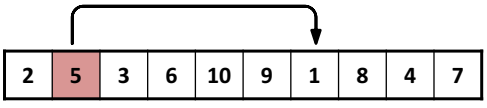

Fig. 9. Procedure of $\boldsymbol{LN}_4$.

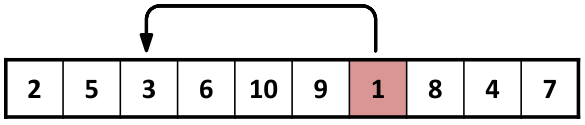

Fig. 10. Procedure of $\boldsymbol{LN}_5$.

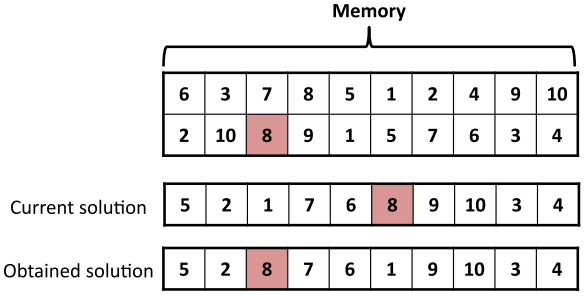

Fig. 11. Procedure of $\boldsymbol{LN}_6$.

4.3.7. Neighborhood structure $\boldsymbol{SN}_1$

The steps for generating neighbor solutions by SN_1 are as follows (see Fig. 12):

Step 1: Create empty template and it is initialized by 0 and 1 randomly.
Step 2: Copy the blocks from the current solution corresponding to the locations of the "1"s in the binary string to the same positions in new solution.
Step 3: Complete the remaining empty block locations with the unselected jobs randomly.

4.3.8. Neighborhood structure $\boldsymbol{SN}_2$

This subsection uses Opposition-Based Learning (OBL) concept for switching to another part of search space. The basic concept of Opposition-Based Learning (OBL) was firstly proposed as a machine intelligence scheme for reinforcement learning by Tizhoosh [59] in 2005, after short time, it has been utilized to enhance soft computing methods such as artificial neural networks [60–64] and fuzzy systems [65,66]. In a short period of time, some other kinds of opposition-based learning were applied to different areas of science. In recent years, OBS has been proven as an effective methodology to enhance various metaheuristic optimization methods. The achieved empirical results have shown good performance of the concept of opposition in a wide range of learning and optimization fields. OBL has been utilized to improve the success rate of various metaheuristic algorithms such as ant colony optimization [67,68], simulated annealing [69], harmony search [70], biogeography-based optimization [71], differential evolution [72–74], particle swarm optimization [75–77] and gravitational search algorithm [78]. A comprehensive review of these algorithms and also other opposition-based works can be find in [79]. In opposition, the amount of each variable defined as the mirror point of the solution from the center of the search space as below [80].

Definition 1. If $X(x_1, x_2, \ldots, x_d)$ is a point in d-dimensional search space, where $x_1, x_2, \ldots, x_d$ are real numbers and $x_i \in [a_i, b_i], i = 1, 2, \ldots, d$, its opposite point $\breve{X}(\breve{x}_1, \breve{x}_2, \ldots, \breve{x}_d)$ is defined as follows (see Fig. 13):

$$\breve{x}_i = a_i + b_i - x_i, i = 1, 2, \ldots, d.$$

To the best of authors' knowledge, most published papers on opposition-based learning are for solving problems in continuous domains. Recently, it is gradually realized that opposition-based learning can also be modified to be used for solving discrete problems, including traveling salesman

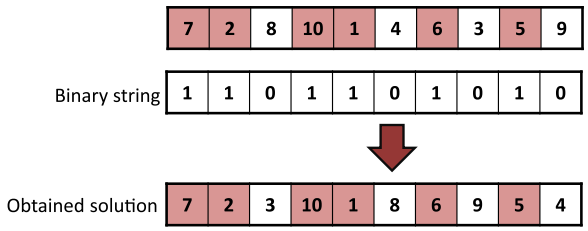

Fig. 12. Procedure of $\boldsymbol{SN}_1$.

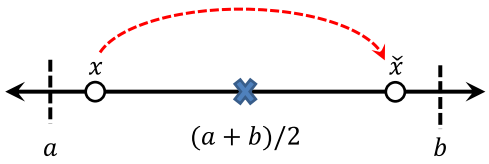

Fig. 13. Opposite point defined in domain [a, b].

problem. It's clear that main problem is that how to define and evaluate the opposite numbers in discrete domain [79]. For example, the attempting to utilize opposition concept by reversing an order in some problems such as TSP really is a wasteful attempt. Ergezer et al. [71], define opposite path as :

Definition 2. Let n be the number of nodes in a graph and $P = [1, 2\ldots n]$ is considered as an even node cycle. The clockwise opposite path, P_o^{CW}, is as follow:

$$P_o^{CW} = \left[1, 1+\frac{n}{2}, 2, 2+\frac{n}{2}, \ldots, \frac{n}{2}-1, n-1, , \frac{n}{2}, n\right]$$

According to above equation, opposite point cannot be used if n is odd. If n is odd then an auxiliary node will be added to the end of the original path. Next, the clockwise opposite path of candidate solution is calculated by above equation, and then the added number from the end of opposite path will be removed final. According to aforementioned fact, Procedure of SN_2 is schematically as follow: (Fig. 14)

4.3.9. Neighborhood structure SN_3

The steps for generating neighbor solutions by SN_3 are as follows (see Fig. 15):

Step 1: Choose a sequence from memory.
Step 2: Create empty template and assign a randomly generated number 0 and 1 to each cell.
Step 3: Copy the blocks from the selected sequence in step 1 corresponding to the locations of the "1"s in the binary string to the same positions in new solution.
Step 4: The blocks that have already been selected from the memory are deleted from the current solution, so that the repetition of a block in the new string is avoided.
Step 5: Complete the remaining empty block locations with the undeleted blocks that remain in the step 4 by preserving their block sequence.

4.4. Intensification phase

Intensification is the search in the current solution for constructing better solutions closer to optimum. The proposed variable neighborhood search uses a straightforward procedure as its intensification sub-algorithm. In the intensification phase, the algorithm exchanges the location of the first two adjacent jobs and evaluates the fitness value of the new sequence. If the fitness value of the new sequence has improved by this swap, algorithm accepts this exchange and restarts the exchange sub-procedure. However, if this exchange does not improve the objective function of the solution, the two jobs will move back to their original

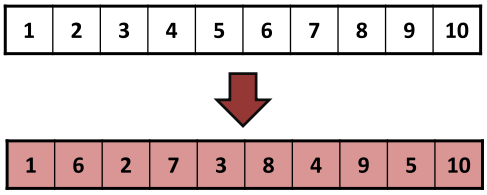

Fig. 14. Procedure of SN_2.

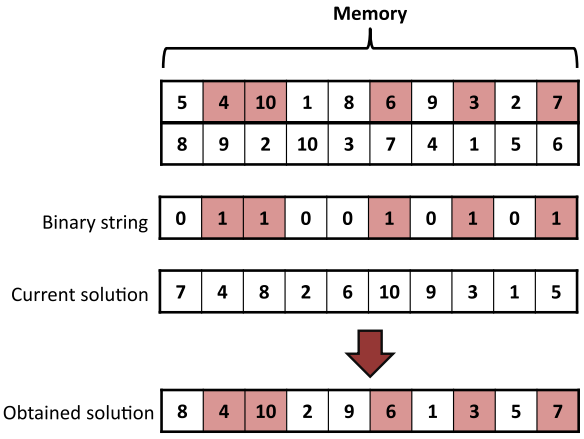

Fig. 15. Procedure of SN_3.

```
π = current sequence
For i=1:n-1 (n is number of jobs)
    Two adjacent jobs i and i+1 are selected randomly from current sequence π.
    π′= swap positions of these jobs.
    If objective function (π′)<objective function (π )
      π=π′,
      i=1,
    End
End
```

Fig. 16. The structure of Intensification phase.

locations and exchange will be applied to the next two adjacent jobs. Intensification phase is provided as follow: (Fig. 16)

4.5. Proposed algorithm

The structure of proposed VNS algorithm is shown in Fig. 17. Proposed VNS is made of shaking and local search procedures. At the first step, the required shaking, local search procedures and stop condition are determined. Next, an initial solution y is randomly generated through the search space and then is considered as the best solution. Generate a neighbor solution y' from the best solution y by shaking procedure SN_k. Set initial value of k equal to 1. After the y' is obtained, local search is applied to improve the quality of obtained solution. At the each iteration a move is selected randomly from all local searches. The probability of selection of each local search is proportional to the number of iteration that it makes an improvement in solutions. We apply this feature in the algorithm using roulette wheel technique. After the local search, the current best solution y is replaced with the local optimum y' if y' is equal or better than y and then k is set to 1 and memory is updated. Otherwise k is increased by one. When k reach to 3, k is reset to 1.

Select set of the neighborhood structures that will be used in the shaking procedure ($SN_k, k = 1,2,3$) and local search ($LN_i, i = 1,2,\dots,6$);
define stopping conditions;
β: total number of local searches, $\beta = 6$;
γ_i:the number of iterations which neighborhood LN_i has improved the solution;
P_i: the probability of chosen move LN_i;
Generate an initial memory and initialize it by zero
Generate an initial solution y randomly and consider it as the best solution.
Locate the initial solution y in the memory
Repeat the following sequence until the stopping condition is met:
Set k ⟵ 1;
While ($k \le k_{max}$) **do**
    Shaking: generate a solution y′ at random from k^{th} neighborhood structure SN_k
    Local search
    l =1;
    While ($l \le N$) **do**
        k: rand (0,1);
        $P_i = {}^{\gamma_i}/_{\sum_{i=1}^{\beta}\gamma_i} \qquad i = 1,\dots,\beta$
        Select move LN_i and generate a neighboring solution y" from the current solution y′
        if y" is better than y′ **then**
            Set y′ ⟵ y";
        $\gamma_i = \gamma_i + 1$;
        End if
    End while
    Apply intensive phase
Move or not : if local optimum y' is better than the best solution y then y ⟵ y',
Update the memory;
Set k⟵1;
otherwise k⟵k+1;
End while
End repeat

Fig. 17. The structure of proposed VNS.

Searching procedure is repeated for N times. The algorithm is repeated until a stop condition is met.

5. Experimental design and data setting

5.1. Data generation

In this part of the paper, of the proposed mathematical model is validated and the performance of the developed variable neighborhood search is investigated. In this regard, a set of test problems are produced randomly. It should be mentioned that each parameter of test problems are generated randomly based on special distribution function that are shown in Table 1. Basically, test problems are adopted from the literature, and as mentioned in the assumptions, no release dates are considered. Based on Table 1, the size of test problems are varied from 8 to 100 jobs. The processing times are generated by discrete uniform distribution within [1,50] [81]. Duration of availability (AV) and unavailability period (UN) for each instance are constant and are generated based on Uniform [150,200] and Uniform [5,20] for availability and unavailability periods, respectively [81]. Due date is another parameter that is generated for each job by Uniform $[d_{min} - \lambda, d + d_{max}]$, where $d_{min} = P(1 - TEF), P = round\left(\sum_{i=1}^{n} p_i + \left(\sum_{i=1}^{n} p_i/AV\right) \times UN\right)$ and $\lambda = P(RDD/2)$. *TEF* is the tardiness/earliness, and *RDD* is relative range of due dates. *RDD* gets the values of 0.2, 0.5 and 0.8. Besides, *TEF* gets the values of 0.2, 0.35 and 0.5.

Table 1
Test problem characteristics.

Factor	Level
Number of jobs	8,10,12,15,20,50,70 and 100
Number of batches	3
Processing times	Uniform [1,50]
Duration of each unavailability for each instance (UN)	Uniform [5,20]
Duration of each availability for each instance (AV)	$\text{Max}\left\{\left(\frac{\sum_{i=1}^{n} p_i}{3}\right),\right.$
TEF	0.2, 0.35 and 0.5
RDD	0.2, 0.5 and 0.8

Seventy-two test problems based on aforementioned parameters are generated. For each size of test problems, nine instances are considered. Name of generated instances with their relevant parameters are introduced in Table 2.

5.2. Experimental results

Since no similar work has not been seen in the literature as the benchmark for the considered problem, first of all the proposed mathematical model is validated through some test

Table 2
Name of problems generated and their relevant parameters.

Number of jobs	*TEF*=0.2			*TEF*=0.35			*TEF*=0.5		
	RDD=0.2	*RDD*=0.5	*RDD*=0.8	*RDD*=0.2	*RDD*=0.5	*RDD*=0.8	*RDD*=0.2	*RDD*=0.5	*RDD*=0.8
8	JG01	JG02	JG03	JG04	JG05	JG06	JG07	JG08	JG09
10	JG10	JG11	JG12	JG13	JG14	JG15	JG16	JG17	JG18
12	JG19	JG20	JG21	JG22	JG23	JG24	JG25	JG26	JG27
15	JG28	JG29	JG30	JG31	JG32	JG33	JG34	JG35	JG36
20	JG37	JG38	JG39	JG40	JG41	JG42	JG43	JG44	JG45
50	JG46	JG47	JG48	JG49	JG50	JG51	JG52	JG53	JG54
70	JG55	JG56	JG57	JG58	JG59	JG60	JG61	JG62	JG63
100	JG64	JG65	JG66	JG67	JG68	JG69	JG70	JG71	JG72

problems. After that, the obtained results from the proposed meta-heuristic algorithm are compared with the results obtained from mathematical model. It should be remained that the proposed mathematical model is solved exactly via GAMS optimization software. As described in Section 3, the memory module is used in the search process and the intensive phase strategy is developed in the proposed algorithm to improve the efficiency of the variable neighborhood search. In order to examine neighborhood structures that are made based on memory and intensive phase, we performed computational experiment, and the obtained results are introduced in Table 3. As we can see in this table, four different variable neighborhood search approaches are considered, i.e. VNS I, VNS II, VNS III and VNS IV. In VNS I intensive phase and both neighborhood structures that are based on memory are imbedded. In VNS II the proposed algorithm is run without intensive phase. In neither VNS III nor VNS IV neighborhood structures with memory are not considered. However in VNS IV intensive phase is considered similar to VNS II.

Since stopping criteria can have effect on runtime and the quality of the solutions, we have done some tests to reach good value of them. To gain the best value of stopping condition parameters, we utilized one-third of instances of different sizes as test problems and evaluated the behavior of them. At first, in order to tuned the stopping condition of inner loop, we have considered fourth different values; namely, $1/2N$, N, $3/2N$, $2N$, for it, where N is the number of jobs and also we have used maximum number of function evaluation ($3000 \times N$ objective function evaluations, where N is the number of jobs) as a stopping condition of algorithm. According to the obtained results, N was the best value for stopping criteria of inner loop. Moreover, results indicated that objective function had not significant changes after $2000 \times N$ objective function evaluations. In fact, algorithm converges into the optimal solution or near the optimal solution after $2000 \times N$ iterations. Therefore, we have considered $2000 \times N$ as the stopping condition of algorithm in order to solve problem and obtain best solution in a reasonable time. In order to conduct a fair comparison, these criteria were used for all algorithms.

The proposed VNSs are coded using MATLAB. All of experiments are run on Intel Core i7 processor running at 2 GHz with 4 GB of RAM. Each test was repeated for 10 times and the average of these runs is reported. The results of the test problems, including objective function and run times are recorded in Table 3 for each algorithm. An index called the "% error" is defined as follows:

$$\%\text{error} = \frac{C-A}{A}$$

A is the best value that are obtained by metaheuristic algorithms or GAMS, and C is the average of 10 runs value obtained by respective solution method. Consider the "**" in columns indicate that this solution is best among other solutions obtained by other methods. Furthermore, "*" in column indicate that this solution is the optimum solution. Also, Rank column returns rank of a solution method among other methods.

As mentioned former, validation of the proposed variable neighborhood search method is investigated via comparison of results by the results of the exact solution of mathematical model in GAMS. As far as Table 3 shows, error index is small for most of test problems, and it seems reasonable to trust to the proposed meta-heuristic algorithm. It should be mentioned that in medium-size test problems (i.e. instances with 15 to 20 jobs) large-size test problems (i.e. instances with more than 20 jobs) that GAMS fails to obtain a solution in reasonable time, meta-heuristic algorithms are applied to get solutions which can be trusted to be near optimum. As seen in Table 3, the best performing algorithm based on average error is VNS I with average error of 1.5%. The second best is VNS II with average error of 5.2%. VNS III has the worst performance with average error of 8.1% (see Fig. 18).

On the other hand, according to the rank of the proposed algorithms among other methods, VNS I has best performance in 51 out of 72 test problems. The second best is VNS II with 27 out of 72 test problems. VNS IV and VNS III stand on next ranks with 25 and 25 out of 72 test problems, respectively. Finally, based on the achieved results, it can be in inferred that both of intensive phase and especially local searches that are based on memory can increase performance of the proposed variable neighborhood search significantly.

6. Conclusion

This paper deals with a single-machine scheduling problem in which jobs have distinct due dates, and machine has some

Table 3
Comparison of results of the model solved by GAMS with the VNSs.

	GAMS				VNS I				VNS II				VNS III				VNS IV			
	A	*T*	*R*	%E	*A*	*T*	*R*	*%E*	*A*	*T*	*R*	*%E*	*A*	*T*	*R*	*%E*	*A*	*T*	*R*	*%E*
JG01	170**	4.4	1	0.0%	170**	2.304	1	0.0%	170**	2.664	1	0.0%	170**	2.499	1	0.00%	170**	2.482	1	0.0%
JG02	212**	9.3	1	0.0%	212**	2.352	1	0.0%	212**	2.336	1	0.0%	212**	2.397	1	0.00%	212**	2.499	1	0.0%
JG03	100**	2.9	1	0.0%	100**	2.175	1	0.0%	100**	2.1	1	0.0%	100**	2.431	1	0.00%	100**	2.397	1	0.0%
JG04	191**	6	1	0.0%	191**	2.288	1	0.0%	191**	2.256	1	0.0%	191**	2.465	1	0.00%	191**	2.448	1	0.0%
JG05	85**	0.8	1	0.0%	85**	2.288	1	0.0%	85**	2.16	1	0.0%	85**	2.431	1	0.00%	85**	2.55	1	0.0%
JG06	28**	0.2	1	0.0%	28**	2.235	1	0.0%	28**	2.25	1	0.0%	28**	2.414	1	0.00%	28**	2.499	1	0.0%
JG07	176**	8.9	1	0.0%	176**	2.272	1	0.0%	176**	2.175	1	0.0%	176**	2.533	1	0.00%	176**	2.397	1	0.0%
JG08	152**	4.7	1	0.0%	152**	2.384	1	0.0%	152**	2.205	1	0.0%	152**	2.397	1	0.00%	152**	2.55	1	0.0%
JG09	62**	1.8	1	0.0%	62**	2.205	1	0.0%	62**	2.19	1	0.0%	62**	2.38	1	0.00%	62**	2.465	1	0.0%
JG10	257**	29.3	1	0.0%	257**	2.84	1	0.0%	257**	2.774	1	0.0%	257**	2.982	1	0.00%	257**	3.087	1	0.0%
JG11	153**	2.2	1	0.0%	153**	2.755	1	0.0%	153**	2.831	1	0.0%	153**	2.961	1	0.00%	153**	2.982	1	0.0%
JG12	123**	12.8	1	0.0%	123**	2.793	1	0.0%	123**	2.812	1	0.0%	123**	3.15	1	0.00%	123**	2.982	1	0.0%
JG13	175**	24.5	1	0.0%	175**	2.98	1	0.0%	175**	2.831	1	0.0%	175**	3.129	1	0.00%	175**	3.087	1	0.0%
JG14	142**	15.3	1	0.0%	142**	2.755	1	0.0%	142**	2.66	1	0.0%	142**	3.024	1	0.00%	142**	3.108	1	0.0%
JG15	73**	21.2	1	0.0%	73**	2.717	1	0.0%	73**	2.682	1	0.0%	73**	3.129	1	0.00%	73**	3.129	1	0.0%
JG16	192**	18.9	1	0.0%	192**	3	1	0.0%	192**	2.812	1	0.0%	192**	3.108	1	0.00%	192**	3.003	1	0.0%
JG17	111**	5.5	1	0.0%	111**	2.755	1	0.0%	111**	2.85	1	0.0%	111**	3.003	1	0.00%	111**	2.961	1	0.0%
JG18	83**	4.2	1	0.0%	83**	2.793	1	0.0%	83**	2.812	1	0.0%	83**	2.94	1	0.00%	83**	3.045	1	0.0%
JG19	283**	339	1	0.0%	283**	3.408	1	0.0%	284.4	3.504	5	0.5%	283.8	3.9	4	0.30%	283**	3.77	1	0.0%
JG20	178**	10.5	1	0.0%	178**	3.45	1	0.0%	178.2	3.243	4	0.1%	178**	3.744	1	0.00%	178.7	3.744	5	0.4%
JG21	91**	2.3	1	0.0%	91**	3.266	1	0.0%	91.3	3.404	5	0.3%	91.2	3.848	3	0.20%	91.2	3.796	3	0.2%
JG22	252*	3600	1	0.0%	252.5	3.456	2	0.2%	253.5	3.528	4	0.6%	252.5	3.666	3	0.20%	255.5	3.9	5	1.4%
JG23	177**	163.1	1	0.0%	177**	3.552	1	0.0%	177.5	3.408	3	0.3%	177.9	3.666	5	0.50%	177.5	3.666	3	0.3%
JG24	135**	126.1	1	0.0%	135**	3.381	1	0.0%	135.4	3.212	4	0.3%	135.5	3.692	5	0.40%	135.1	3.874	3	0.1%
JG25	283**	1136	1	0.0%	283**	3.576	1	0.0%	283**	3.576	1	0.0%	283.6	3.718	5	0.20%	283.3	3.874	4	0.1%
JG26	211**	1684	1	0.0%	211.8	3.48	3	0.4%	212.8	3.243	5	0.9%	211.2	3.848	2	0.10%	211.8	3.744	4	0.4%
JG27	108**	110	1	0.0%	108.0	3.45	2	0.0%	108.4	3.45	4	0.4%	108.4	3.77	5	0.40%	108.2	3.64	3	0.2%
JG28	260.0	3600	2	2.2%	254.2*	4.38	1	0.0%	260.0	4.38	2	2.2%	260.3	4.736	4	2.40%	260.3	4.512	4	2.4%
JG29	206**	88.3	1	0.0%	206**	4.35	1	0.0%	208.0	4.06	5	1.0%	206**	4.672	1	0.00%	206.2	4.736	4	0.1%
JG30	251**	350.9	1	0.0%	251**	4.205	1	0.0%	252.0	4.321	5	0.4%	251.8	4.48	4	0.30%	251.3	4.672	3	0.1%
JG31	332.0	3600	5	16.0%	286.8	4.38	3	0.2%	286.2*	4.5	1	0.0%	286.6	4.768	2	0.10%	287.4	4.64	4	0.4%
JG32	284.0	3600	5	7.4%	264.5*	4.321	1	0.0%	274.4	4.35	4	3.7%	265.3	4.672	2	0.30%	271.5	4.512	3	2.7%
JG33	138**	340.5	1	0.0%	138.5	4.147	5	0.3%	138**	4.06	1	0.0%	138.3	4.828	4	0.20%	138**	4.672	1	0.0%
JG34	340.0	3600	2	1.5%	335*	4.2	1	0.0%	343.0	4.23	5	2.4%	340.3	4.851	3	1.60%	341.4	4.608	4	1.9%
JG35	275.0	3600	2	1.0%	272.2*	4.292	1	0.0%	296.0	4.06	5	8.7%	275	4.608	2	1.00%	275.3	4.704	4	1.1%
JG36	209.0	3600	2	8.6%	192.5*	4.118	1	0.0%	209.4	4.116	4	8.8%	209	4.64	2	8.60%	235.2	4.512	5	22.2%
JG37	275*	3600	1	0.0%	373.5	5.64	2	35.8%	406.2	5.85	5	47.7%	401.6	6.321	3	46.00%	402.0	6.106	4	46.2%
JG38	311.0	3600	3	0.2%	310.3*	5.616	1	0.0%	324.0	5.358	5	4.4%	310.3*	6.063	1	0.00%	311.6	6.45	4	0.4%
JG39	193.0	3600	5	23.9%	155.7*	5.513	1	0.0%	184.4	5.4	3	18.4%	184	6.106	2	18.20%	184.6	6.063	4	18.5%
JG40	406*	3600	1	0.0%	406.4	5.76	2	0.1%	415.1	5.577	5	2.2%	406.4	6.02	2	0.10%	412.2	6.235	4	1.5%
JG41	340*	3600	1	0.0%	351.0	5.85	3	3.2%	357.4	5.624	5	5.1%	341	6.278	2	0.30%	352.1	6.106	4	3.5%
JG42	278.0	3600	5	22.5%	238.5	5.32	3	5.0%	235.5	5.476	2	3.7%	227*	6.235	1	0.00%	252.3	6.278	4	11.1%
JG43	367.0	3600	5	1.0%	363.5*	5.8	1	0.0%	366.4	5.538	4	0.8%	365	6.321	2	0.40%	365.0	6.106	2	0.4%
JG44	358.0	3600	5	20.8%	296.2*	5.499	1	0.0%	322.3	5.472	4	8.8%	303.3	6.407	3	2.40%	296.2*	6.02	1	0.0%
JG45	203.0	3600	4	3.9%	195.3*	5.513	1	0.0%	195.3*	5.04	1	0.0%	202.6	6.106	3	3.70%	214.2	6.45	5	9.6%
JG46	–	–	–	–	897.4*	14.6	1	0.0%	957.9	13.97	3	6.7%	946.9	16.5	2	5.50%	979.9	16.06	4	9.2%
JG47	–	–	–	–	690.4	13.97	2	2.6%	695.8	13.34	3	3.4%	705.8	16.28	4	4.90%	673*	15.73	1	0.0%
JG48	–	–	–	–	692.0	13.21	3	5.5%	656*	12.78	1	0.0%	699.8	15.7	4	6.70%	681.7	16.5	2	3.9%
JG49	–	–	–	–	1143.7*	14.41	1	0.0%	1,170.5	13.73	4	2.3%	1166.60	15.84	3	2.00%	1159.5	15.84	2	1.4%
JG50	–	–	–	–	850.2	13.25	2	3.2%	890.9	13.2	3	8.2%	823.6*	15.95	1	0.00%	904.0	16.06	4	9.8%
JG51	–	–	–	–	521.2	13.44	2	3.2%	505.0*	12.96	1	0.0%	559	15.95	3	10.70%	566.3	15.73	4	12.1%
JG52	–	–	–	–	1010.9*	14.45	1	0.0%	1129.1	14.3	3	11.7%	1134.10	15.51	4	12.20%	1123.2	15.73	2	11.1%
JG53	–	–	–	–	946.5*	13.92	1	0.0%	965.8	13.49	2	2.0%	1022.10	15.84	4	8.00%	970.8	16.06	3	2.6%
JG54	–	–	–	–	420.2*	13.52	1	0.0%	507.5	12.58	2	20.8%	624.1	16.39	4	48.50%	597.2	15.51	3	42.1%
JG55	–	–	–	–	1100.6*	21.45	1	0.0%	1123.1	24.72	2	2.0%	1128.60	28.2	3	2.50%	1287.7	25.84	4	17.0%
JG56	–	–	–	–	892.4	25.56	2	16.3%	896.0	21.14	3	16.8%	1039.00	24.84	4	35.40%	767.2*	29.21	1	0.0%
JG57	–	–	–	–	759.2	21.22	3	4.2%	883.6	19.15	4	21.3%	728.3*	23.54	1	0.00%	740.2	24.79	2	1.6%
JG58	–	–	–	–	1252.6	22.59	2	4.5%	1269.7	21.74	3	5.9%	1289.80	26.06	4	7.60%	1198.7*	29.23	1	0.0%
JG59	–	–	–	–	917.9*	23.19	1	0.0%	1096.5	22.05	2	19.4%	1220.70	29.3	3	33.00%	1344.2	25.08	4	46.4%
JG60	–	–	–	–	516.1*	21.25	1	0.0%	573.6	19.58	2	11.1%	648.4	29.59	4	25.60%	581.5	30.81	3	12.6%

Table 3 (*continued*)

	GAMS				VNS I				VNS II				VNS III				VNS IV			
	A	*T*	*R*	%E	*A*	*T*	*R*	*%E*	*A*	*T*	*R*	*%E*	*A*	*T*	*R*	*%E*	*A*	*T*	*R*	*%E*
JG61	–	–	–	–	1065.7*	26.11	1	0.0%	1158.5	23.27	2	8.7%	1634.90	25.36	4	53.40%	1237.6	25.77	3	16.1%
JG62	–	–	–	–	887.6*	24.67	1	0.0%	964.8	21.71	2	8.7%	1124.80	28.38	3	26.70%	1199.8	26.99	4	35.2%
JG63	–	–	–	–	530.6*	22.93	1	0.0%	675.0	18.61	4	27.2%	634	30.44	2	19.50%	669.9	27.7	3	26.2%
JG64	–	–	–	–	1340.1	35.53	2	2.8%	1367.9	45.02	3	5.0%	1399.40	46.8	4	7.40%	1303.1*	43.22	1	0.0%
JG65	–	–	–	–	986.2*	45.81	1	0.0%	1310.0	37.31	3	32.8%	1378.80	41.02	4	39.80%	1070.3	47.76	2	8.5%
JG66	–	–	–	–	921.1	39.07	2	3.2%	892.4*	36.73	1	0.0%	938.1	37.73	3	5.10%	939.4	43.57	4	5.3%
JG67	–	–	–	–	1312.3*	42.04	1	0.0%	1316.7	33.84	2	0.3%	1798.00	45.02	4	37.00%	1431.3	44.28	3	9.1%
JG68	–	–	–	–	1587.3	38.56	2	0.3%	1582.2*	31.72	1	0.0%	1702.80	49.29	3	7.60%	1844.3	37.79	4	16.6%
JG69	–	–	–	–	562.1*	35.92	1	0.0%	617.7	34.37	2	9.9%	889	53.2	4	58.20%	711.7	48.68	3	26.6%
JG70	–	–	–	–	1510.8	38.02	2	6.0%	1424.9*	39.65	1	0.0%	1888.30	47.85	4	32.50%	1832.9	44.64	3	28.6%
JG71	–	–	–	–	1432.8	42.99	3	11.7%	1350.8	33.2	2	5.3%	1282.2*	45.18	1	0.00%	1672.5	46.42	4	30.4%
JG72	–	–	–	–	671.5*	36.08	1	0.0%	831.6	34.92	4	23.8%	699.3	53.09	2	4.10%	746.9	46.13	3	11.2%

A=Average of obtained solutions
T=CPU time (second)
R=rank among other obtained solutions
%E=%error

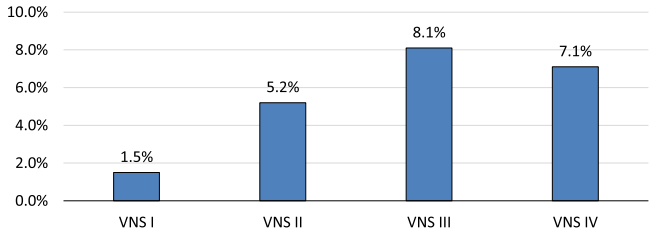

Fig. 18. Average %error for each VNS.

unavailability periods. The aim is to schedule jobs to minimize the sum of maximum earliness and tardiness. A mathematical formulation is developed to solve small size samples of the considered problem exactly. In this research, a knowledge-based variable neighborhood search algorithm is developed to solve large size samples, too. In the proposed algorithm, two knowledge module based local search is used to guide VNS. Knowledge module extracts the knowledge from the best solutions and feed it back to the local search throughout searching process. It is shown through computational experimentation that the proposed algorithm is able to obtain optimal or near-optimal solutions within a reasonable amount of time. The following suggestions are offered for future works:

I. Considering additional constrains, such as learning effects or breakdowns, in scheduling problems.
II. Investigating scheduling problem with unavailability constraints in the case of having parallel machines.
III. Developing other meta-heuristic algorithms to optimize the considered problem, and making a comprehensive comparison with the proposed solution method.
IV. Proposing other objective functions such as total completion time, total tardiness and total machine workloads to conduct multi-objective scheduling problems with unavailability constraints.

References

[1] Azadeh A, Sheikhalishahi M, Firoozi M, Khalili S. An integrated multi-criteria Taguchi computer simulation-DEA approach for optimum maintenance policy and planning by incorporating learning effects. *Int J Prod Res* 2013;51:5374–85.
[2] Yazdani M, Khalili SM, Jolai F. A parallel machine scheduling problem with two-agent and tool change activities: an efficient hybrid metaheuristic algorithm. *Int J Comput Integr Manuf* 2016:1–14.
[3] Azadeh A, Sheikhalishahi M, Khalili SM, Firoozi M. An integrated fuzzy simulation–fuzzy data envelopment analysis approach for optimum maintenance planning. *Int J Comput Integr Manuf* 2014;27:181–99.
[4] Laalaoui Y, M'Hallah R. A binary multiple knapsack model for single machine scheduling with machine unavailability. *Comput Oper Res* 2016;72:71–82.
[5] Low C, Ji M, Hsu C-J, Su C-T. Minimizing the makespan in a single machine scheduling problems with flexible and periodic maintenance. *Appl Math Model* 2010;34:334–42.
[6] Rapine C, Brauner N, Finke G, Lebacque V. Single machine scheduling with small operator-non-availability periods. *J Sched* 2012;15:127–39.
[7] N. Mladenovic. A variable neighborhood algorithm-a new metaheuristic for combinatorial optimization. in: *Papers presented at Optimization Days*; 1995. p. 112.
[8] Avanthay C, Hertz A, Zufferey N. A variable neighborhood search for graph coloring. *Eur J Oper Res* 2003;151:379–88.
[9] Ribeiro CC, Souza M c C. Variable neighborhood search for the degree-constrained minimum spanning tree problem. *Discret Appl Math* 2002;118:43–54.
[10] Sevkli M, Aydin ME. Parallel variable neighbourhood search algorithms for job shop scheduling problems. *IMA J Manag Math* 2007;18:117–33.
[11] Wang L. A hybrid genetic algorithm–neural network strategy for simulation optimization. *Appl Math Comput* 2005;170:1329–43.
[12] R. G. ReynoldsAn introduction to cultural algorithms, in: *Proceedings of the third annual conference on evolutionary programming*; 1994. p. 131–139.
[13] Chung C-J, Reynolds RG. A testbed for solving optimization problems using cultural algorithms. *Evolut Program* 1996:225–36.
[14] Ho NB, Tay JC, Lai EM-K. An effective architecture for learning and evolving flexible job-shop schedules. *Eur J Oper Res* 2007;179:316–33.
[15] Louis SJ, McDonnell J. Learning with case-injected genetic algorithms. *Evolut Comput, IEEE Trans* 2004;8:316–28.

[16] Michalski RS. Learnable evolution model: evolutionary processes guided by machine learning. *Mach Learn* 2000;38:9–40.
[17] Xing L-N, Chen Y-W, Wang P, Zhao Q-S, Xiong J. A knowledge-based ant colony optimization for flexible job shop scheduling problems. *Appl Soft Comput* 2010;10:888–96.
[18] Schmidt G. Scheduling with limited machine availability. *Eur J Oper Res* 2000;121:1–15.
[19] Ángel-Bello F, Álvarez A, Pacheco J, Martínez I. A single machine scheduling problem with availability constraints and sequence-dependent setup costs. *Appl Math Model* 2011;35:2041–50.
[20] Rustogi K, Strusevich VA. Single machine scheduling with general positional deterioration and rate-modifying maintenance. *Omega* 2012;40:791–804.
[21] Zammori F, Braglia M, Castellano D. Harmony search algorithm for single-machine scheduling problem with planned maintenance. *Comput Ind Eng* 2014;76:333–46.
[22] Wang S, Liu M. A branch and bound algorithm for single-machine production scheduling integrated with preventive maintenance planning. *Int J Prod Res* 2013;51:847–68.
[23] Yin Y, Xu J, Cheng T, Wu CC, Wang DJ. Approximation schemes for single-machine scheduling with a fixed maintenance activity to minimize the total amount of late work. *Nav Res Logist (NRL)* 2016;63:172–83.
[24] Xu D, Wan L, Liu A, Yang D-L. Single machine total completion time scheduling problem with workload-dependent maintenance duration. *Omega* 2015;52:101–6.
[25] Cui W-W, Lu Z, Pan E. Integrated production scheduling and maintenance policy for robustness in a single machine. *Comput Oper Res* 2014;47:81–91.
[26] Luo W, Cheng TE, Ji M. Single-machine scheduling with a variable maintenance activity. *Comput Ind Eng* 2015;79:168–74.
[27] Hfaiedh W, Sadfi C, Kacem I, Hadj-Alouane A. A branch-and-bound method for the single-machine scheduling problem under a non-availability constraint for maximum delivery time minimization 2/1/. *Appl Math Comput* 2015;252:496–502.
[28] Bai J, Li Z-R, Wang J-J, Huang X. Single machine common flow allowance scheduling with deteriorating jobs and a rate-modifying activity. *Appl Math Model* 2014;38:5431–8.
[29] Vahedi-Nouri B, Fattahi P, Rohaninejad M, Tavakkoli-Moghaddam R. Minimizing the total completion time on a single machine with the learning effect and multiple availability constraints 3/1/. *Appl Math Model* 2013;37:3126–37.
[30] Li W-X, Zhao C-L. Deteriorating jobs scheduling on a single machine with release dates, rejection and a fixed non-availability interval. *J Appl Math Comput* 2015;48:585–605.
[31] Kacem I, Kellerer H, Lanuel Y. Approximation algorithms for maximizing the weighted number of early jobs on a single machine with non-availability intervals. *J Comb Optim* 2015;30:403–12.
[32] M. Gu, X. Lu, J. Gu, Y. ZhangSingle-machine scheduling problems with machine aging effect and an optional maintenance activity *Applied Mathematical Modelling*; 2016.
[33] Liu M, Wang S, Chu C, Chu F. An improved exact algorithm for single-machine scheduling to minimise the number of tardy jobs with periodic maintenance. *Int J Prod Res* 2015:1–12.
[34] Wang S. Bi-objective optimisation for integrated scheduling of single machine with setup times and preventive maintenance planning. *Int J Prod Res* 2013;51:3719–33.
[35] Chen Y, Zhang A, Tan Z. Complexity and approximation of single machine scheduling with an operator non-availability period to minimize total completion time. *Inf Sci* 2013;251:150–63.
[36] Fan J, Lu X, Liu P. Integrated scheduling of production and delivery on a single machine with availability constraint 1/11/. *Theor Comput Sci* 2015;562:581–9.
[37] Mashkani O, Moslehi G. Minimising the total completion time in a single machine scheduling problem under bimodal flexible periodic availability constraints. *Int J Comput Integr Manuf* 2016;29:323–41.
[38] Nian H, Mao Z. Single-machine scheduling with job rejection, deteriorating effects, and deteriorating maintenance activities. *Math Probl Eng* 2013;2013.
[39] Hsu C-J. Single-machine scheduling with aging effects and optional maintenance activity considerations. *Math Probl Eng* 2013;2013.
[40] Amin-Nayeri M, Moslehi G. An optimum algorithm for single machine with early/tardy cost. *Esteghlal—J Eng* 2000;19:35–48.
[41] Tavakkoli-Moghaddam R, Moslehi G, Vasei M, Azaron A. Optimal scheduling for a single machine to minimize the sum of maximum earliness and tardiness considering idle insert. *Appl Math Comput* 2005;167:1430–50.
[42] Tavakkoli-Moghaddam R, Moslehi G, Vasei M, Azaron A. A branch-and-bound algorithm for a single machine sequencing to minimize the sum of maximum earliness and tardiness with idle insert. *Appl Math Comput* 2006;174:388–408.
[43] Tavakkoli-Moghaddam R, Vasei M. A single machine sequencing problem with idle insert: simulated annealing and branch-and-bound methods. *Int J Ind Eng* 2008;19.
[44] Moslehi G, Mirzaee M, Vasei M, Modarres M, Azaron A. Two-machine flow shop scheduling to minimize the sum of maximum earliness and tardiness. *Int J Prod Econ* 2009;122:763–73.
[45] Mahnam M, Moslehi G. A branch-and-bound algorithm for minimizing the sum of maximum earliness and tardiness with unequal release times. *Eng Optim* 2009;41:521–36.
[46] Moslehi G, Mahnam M, Amin-Nayeri M, Azaron A. A branch-and-bound algorithm to minimise the sum of maximum earliness and tardiness in the single machine. *Int J Oper Res* 2010;8:458–82.
[47] Nekoiemehr N, Moslehi G. Minimizing the sum of maximum earliness and maximum tardiness in the single-machine scheduling problem with sequence-dependent setup time. *J Oper Res Soc* 2011;62:1403–12.
[48] Mahnam M, Moslehi G, Ghomi SMTF. Single machine scheduling with unequal release times and idle insert for minimizing the sum of maximum earliness and tardiness. *Math Comput Model* 2013;57:2549–63.
[49] Benmansour R, Allaoui H, Artiba A, Hanafi S. Minimizing the weighted sum of maximum earliness and maximum tardiness costs on a single machine with periodic preventive maintenance 7//. *Comput Oper Res* 2014;47:106–13.
[50] Azadeh A, Seif J, Sheikhalishahi M, Yazdani M. An integrated support vector regression–imperialist competitive algorithm for reliability estimation of a shearing machine 2016/01/02. *Int J Comput Integr Manuf* 2016;29:16–24.
[51] Yazdani M, Jolai F. Lion Optimization Algorithm (LOA): a nature-inspired metaheuristic algorithm 1//. *J Comput Des Eng* 2016;3:24–36.
[52] Hansen P, Mladenović N. *Variable neighborhood search*. *Search methodologies*. Springer; 313–37.
[53] Liao C-J, Cheng C-C. A variable neighborhood search for minimizing single machine weighted earliness and tardiness with common due date. *Comput Ind Eng* 2007;52:404–13.
[54] Kirlik G, Oguz C. A variable neighborhood search for minimizing total weighted tardiness with sequence dependent setup times on a single machine. *Comput Oper Res* 2012;39:1506–20.
[55] Liu L, Zhou H. Hybridization of harmony search with variable neighborhood search for restrictive single-machine earliness/tardiness problem. *Inf Sci* 2013;226:68–92.
[56] Arroyo JEC, dos Santos Ottoni R, de Paiva Oliveira A. Multi-objective variable neighborhood search algorithms for a single machine scheduling problem with distinct due windows. *Electron Notes Theor Comput Sci* 2011;281:5–19.
[57] Wang X, Tang L. A population-based variable neighborhood search for the single machine total weighted tardiness problem. *Comput Oper Res* 2009;36:2105–10.
[58] Hsu C-J, Low C, Su C-T. A single-machine scheduling problem with maintenance activities to minimize makespan. *Appl Math Comput* 2010;215:3929–35.
[59] Tizhoosh HR. Opposition-based learning: a new scheme for machine intelligence. *CIMCA/IAWTIC* 2005:695–701.
[60] M. Rashid, A. R. Baig Improved opposition-based PSO for feedforward neural network training. In: *Proceedings of the International Conference on Information Science and Applications (ICISA)*; 2010. p. 1–6.
[61] M. Shokri, H. R. Tizhoosh, M. S. Kamel Opposition-based Q (λ) with non-markovian update. In: *Proceedings of the IEEE International*

Symposium on Approximate Dynamic Programming and Reinforcement Learning, ADPRL ; 2007. p. 288–295.

[62] M. Ventresca, H. R. Tizhoosh. Improving the convergence of back-propagation by opposite transfer functions. In: *Proceedings of the International Joint Conference on Neural Networks, IJCNN'06*; 2006. p. 4777–4784.

[63] M. Ventresca, H. R. Tizhoosh. Opposite transfer functions and back-propagation through time. In: *Proceedings of the IEEE Symposium on Foundations of Computational Intelligence, FOCI* ; 2007. p. 570–577.

[64] M. Ventresca, H. R. Tizhoosh. Numerical condition of feedforward networks with opposite transfer functions. In: *Proceedings of the IEEE International Joint Conference on Neural Networks, IJCNN (IEEE World Congress on Computational Intelligence)* ; 2008. p. 3233–3240.

[65] H. R. Tizhoosh, F. Sahba. Quasi-global oppositional fuzzy thresholding. In: *Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE* ; 2009. p. 1346–1351.

[66] Tizhoosh HR. Opposite fuzzy sets with applications in image processing. *IFSA/EUSFLAT Conf* 2009:36–41.

[67] Malisia AR, Tizhoosh HR. Applying opposition-based ideas to the ant colony system. *Swarm Intell Symp, 2007 SIS 2007 IEEE* 2007:182–9.

[68] A. R. MalisiaInvestigating the application of opposition-based ideas to ant algorithms; 2007.

[69] M. Ventresca, H. R. Tizhoosh. Simulated annealing with opposite neighbors. In: *Proceedings of the IEEE Symposium on Foundations of Computational Intelligence, FOCI* ; 2007. p. 186–192.

[70] Gao X, Wang X, Ovaska S, Zenger K. A hybrid optimization method of harmony search and opposition-based learning. *Eng Optim* 2012;44:895–914.

[71] M. Ergezer, D. Simon, D. Du. Oppositional biogeography-based optimization. In: *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, SMC*; 2009. p. 1009–1014.

[72] Rahnamayan S, Tizhoosh HR, Salama MM. Opposition-based differential evolution. *Evolut Comput, IEEE Trans* 2008;**12**:64–79.

[73] S. Rahnamayan, H. R. Tizhoosh, M. M. Salama. Opposition-based differential evolution algorithms. In: *Proceedings of the IEEE Congress on Evolutionary Computation, CEC* ; 2006. p. 2010–2017.

[74] S. Rahnamayan, H. R. Tizhoosh, M. M. Salama. Opposition-based differential evolution for optimization of noisy problems. In: *Proceedings of the IEEE Congress on Evolutionary Computation, CEC*; 2006. p. 1865–1872.

[75] Wang H, Wu Z, Rahnamayan S, Liu Y, Ventresca M. Enhancing particle swarm optimization using generalized opposition-based learning. *Inf Sci* 2011;181:4699–714.

[76] L. Han, X. He. A novel opposition-based particle swarm optimization for noisy problems. In: *Proceedings of the Third International Conference on Natural Computation, ICNC* ; 2007. p. 624–629.

[77] J. Tang, X. Zhao. An enhanced opposition-based particle swarm optimization. In: *Proceedings of the WRI Global Congress on Intelligent Systems, GCIS'09*; 2009. p. 149–153.

[78] Shaw B, Mukherjee V, Ghoshal S. A novel opposition-based gravitational search algorithm for combined economic and emission dispatch problems of power systems. *Int J Electr Power Energy Syst* 2012;35:21–33.

[79] Xu Q, Wang L, Wang N, Hei X, Zhao L. A review of opposition-based learning from 2005 to 2012. *Eng Appl Artif Intell* 2014;29:1–12.

[80] Tizhoosh HR. Opposition-based learning: a new scheme for machine intelligence. In: *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation, and International Conference on Intelligent Agents, Web Technologies and Internet Commerce* ; 2005. p. 695–701.

[81] Low C, Hsu C-J, Su C-T. A modified particle swarm optimization algorithm for a single-machine scheduling problem with periodic maintenance 9//. *Expert Syst Appl* 2010;37:6429–34.