Scientia Iranica E (2020) 27(2), 862–879

Sharif University of Technology
Scientia Iranica
Transactions E: Industrial Engineering
http://scientiairanica.sharif.edu

The stage shop scheduling problem: Lower bound and metaheuristic

M.M. Nasiri* **and M. Hamid**

School of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran.

Received 2 October 2017; received in revised form 10 June 2018; accepted 3 November 2018

KEYWORDS
Scheduling;
Stage shop;
Mixed shop;
Water wave optimization;
Lower bound.

Abstract. Remarkable efforts have been made to develop the job shop scheduling problem up to now. As a novel generalization, the stage shop can be defined as an environment in which each job is composed of some stages and each stage may include one operation or more. A stage can be defined as a subset of operations of a job such that these operations can be done in any arbitrary relative order while the stages should be processed in a predetermined order. In other words, the operations of a stage cannot be initiated until all operations of the prior stage are completed. In this paper, an innovative lower bound by solving the preemptive open shop (using a linear programming model in polynomial time) is devised for the makespan in a stage shop problem. In addition, three metaheuristics, namely firefly (FF), Harmony Search (HS), and Water Wave Optimization (WWO) algorithms, are applied to the problem. The results of the algorithms are compared with each other with the proposed lower bound and a commercial solver.

1. Introduction

One of the important factors that affect the performance of manufacturing systems is their scheduling approach [1]. In particular, the job shop scheduling problem has various applications, even out of the manufacturing environments (e.g., see [2] for train scheduling). However, in practice, job shop assumptions are restrictive and cannot be applied to many cases. Therefore, researchers have presented several extensions of the job shop scheduling problem.

For instance, a mixture of job shop and open shop is defined as mixed shop, which is known as an NP-hard problem in its general form. In this problem, the set of jobs is partitioned into two separate subsets: one subset of the job shop type and one subset of the open shop type. Shakhlevich et al. [3] discussed the complexity of the mixed shop problems with several performance measures and illuminated the border between the NP-hard and polynomially solvable problems.

Moreover, there exist several other extensions of the job shop problem. Ramudhin and Marier [4] defined a problem called "shops with a partial ordering of operations pertaining to each job or machine," which was solved by an extension of the shifting bottleneck method. Besides, Kis [5] proposed a generalization of the job shop scheduling problem in which the routing of each job was a collection of alternative subgraphs constituting a directed acyclic graph. In this configuration, some operations were included in a subgraph, which determined their order of processing. Özgüven et al. [6] developed a mathematical programming model for flexible job shop scheduling problems with process plan flexibility and presented its computational results for hypothetically generated test problems. Nasiri

*. *Corresponding author. Fax: +98 21 8208 44 85*
E-mail address: mmnasiri@ut.ac.ir (M.M. Nasiri)

doi: 10.24200/sci.2018.5199.1146

and Kianfar [7] solved the partial job shop scheduling problem using hybrid scatter search. In another research, Nasiri and Kianfar [8] combined the genetic algorithm and tabu search to solve a new generalization of the mixed shop problem. In [9], Dugarzhapov and Kononov studied a mixed shop problem with preemptions and at most two unit operations per job. They presented a new exact polynomial-time algorithm for solving this problem. Doh et al. [10] suggested a practical priority scheduling approach. In this approach, a combination of operation/machine selection and job sequencing rules is used to concurrently decide about the routing and scheduling of jobs. Nasiri [11] proposed a modified Artificial Bee Colony (ABC) algorithm to solve the stage shop scheduling problem. He compared the results with the basic ABC and a new algorithm called Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES). Inspired by a real industrial case, another extension of the mixed shop problem called the hybrid stage shop was proposed by Rossi et al. [12]. Jin et al. [13] developed several novel mixed integer linear programming models for the scheduling of a job shop flexible manufacturing system. They considered the integration of process planning and scheduling to efficiently utilize the manufacturing resources. In another research, Gao et al. [14] worked on a flexible job shop scheduling problem with the aim of minimizing the weighted combination of two minimization criteria, namely the maximum completion time (makespan) and the mean of earliness and tardiness. In order to solve the problem, they developed a discrete Harmony Search (HS) algorithm. Bożek and Werner [15] suggested models and optimization approaches to a flexible job shop scheduling problem with lot streaming and lot sizing of the variable sublots. Zubaran and Ritt [16] presented a heuristic for the partial shop scheduling problem. They solved various problem sets and showed that their method was effective.

For an optimization problem with min (max) objective function, a lower bound (an upper bound) may be required in order to compare the results with an approximation algorithm. Obtaining this bound helps us estimate the gap between an arbitrary solution and the optimal solution to a problem. In this paper, a new lower bound for the stage shop scheduling problem is computed using a novel method, in which several open shop problems are solved. In addition, the Water Wave Optimization (WWO) algorithm is applied to a scheduling problem for the first time and compared with firefly (FF) and HS algorithms using a new statistical approach.

The remainder of the paper is organized as follows. In Section 2, the stage shop scheduling problem is introduced and its formulation and notation are presented. Section 3 describes the new method for the computation of the lower bound. In Section 4, three proposed stochastic search algorithms are discussed. The computational results are given in Section 5. Finally, Section 6 concludes the paper.

2. Problem definition and notation

In a stage shop, each job consists of several stages comprising operations the relative order of which is under control of the decision maker. If all operations of a job constitute one stage, the job can be considered as an open shop one. The execution order of the stages of a job is predetermined the same as the operations of a job in a job shop. Consequently, if each stage of a job consists of merely one operation, it resembles a job shop job.

Consider a set of jobs $J = \{1, 2, ..., n\}$ and a set of machines $M = \{1, 2, ..., m\}$. Job j includes s_j stages of operations such that these stages should be completed successively pursuant to the stage numbers: $1, 2, ..., s_j$. Stage k of job j includes a set of operations denoted by H_{jk}. Indeed, it is permitted to initiate the execution of the operations of the stage $k + 1$ of job j only when the execution of all operations of stage k (i.e., operations in H_{jk}) is terminated. The processing of each operation a should be done on a predetermined machine without any interruption for p_a time units. It is worth noting that there is no precedence relation between the operations of a particular stage. The other assumptions are as follows:

- From time zero on, all machines and all jobs are accessible;
- None of the jobs visits a particular machine more than once; in other words, recirculation is not allowed;
- There is no precedence relation between the operations of different jobs;
- Transportation times between machines are negligible;
- Only one job can be in execution on a machine at any instant of time;
- There is an infinite buffer capacity between any two machines.

The criterion which should be optimized in this paper is makespan ($C_{\max}$). For the mathematical model and more information about the stage shop problem, the reader is referred to Nasiri and Kianfar [8].

3. The new lower bound

Inspired by Carlier's lower bound [17] for the JSSP, a simple lower bound for the makespan of the stage shop problem has been presented by Nasiri and Kianfar [8].

Their lower bound and the required notation for comprehending the calculation of the new lower bound are presented in Appendix A.

Understanding similarities between the stage shop and open shop scheduling problems can be helpful in introducing a new lower bound for the makespan criterion of the stage shop problem. In the next subsection, the open shop problem that is required for finding the new lower bound is defined first and then, the relation between the objective function of the open shop problem and the new lower bound will be revealed.

3.1. Definition of the open shop problem

Consider the operations that should be processed on machine i in a stage shop problem, which are denoted by I_i. As can be seen in the assumptions of the stage shop (recirculation prohibition), at most one operation from each job is supposed to be processed on a particular machine. If the operation that should be processed by machine i is at stage k of job j, then ordered pair (j,k) is added to K_i. In other words:

$$K_i = \{(j,k) \,|\exists a \in I_i : a \in H_{jk}\}. \tag{1}$$

Now, there is at most one stage in each job and all operations in the selected stages form an open shop problem ($OpenShop_i$). In the new problem, the operations of each job can be performed in any arbitrary order without violating the stage shop constraints, because they are from the same stage of the main stage shop problem. Therefore, any solution to the defined open shop problem is a partial solution to the main stage shop problem.

Example 3.1. See Appendix B.

3.2. Release dates and due dates

After defining the open shop problem, release dates and due dates should be defined for $OpenShop_i$ (which can be represented as $O_{m'} \,|r_j|\, L_{max}$) according to the well-known idea of heads and tails [18], respectively. According to the notation given in Appendix A, for each job j, the release date is equal to r_j and the due date can be calculated by $d_j = LB_1 - q_j$ ($r_j = r_a$ and $q_j = q_a$, which respectively denote the head and tail of job j, are the same for every $a \in H_{jk}$).

3.3. Calculating the new lower bound

Using the provided definitions for $OpenShop_i$, release dates, and due dates, it is sufficient to find the optimal value (minimum) of the maximum lateness (L^*_{max}) in $OpenShop_i$.

Theorem 1. $LB_1 + \max\{0, L^*_{max}(OpenShop_i)\}$ is a lower bound for the stage shop problem.

Proof. It is assumed that there is no recirculation. Therefore, all the operations of job j in $OpenShop_i$ are from the same stage of the stage shop problem. The processing of each operation a at stage k of job j can be started only after the processing of the prior stages of job j is finished. Thus, r_j can be considered as the release date of job j. In addition, the time that is required after the completion of job j is at least equal to q_j. Therefore, for each feasible sequence of operations in $OpenShop_i$ like S and the corresponding schedule in the stage shop problem, the completion time of job $j(C_j)$ plus the tail of job $j(C_j+q_j)$ is a lower bound for the stage shop makespan of any solution with a sequence of operations identical to S (for common operations). Now, if the maximum value of this bound is obtained for all the jobs of $OpenShop_i$, a lower bound is achieved for the makespan of the mentioned order of the operations in the stage shop problem. Consequently, the lower bound can be calculated as the minimum value of this bound for all feasible sequences of $OpenShop_i$. Furthermore, because LB_1 is a constant, we have:

$$\begin{aligned}
&\min_S \left\{\max_j \{C_j + q_j\}\right\} = LB_1 \\
&\quad + \min_S \left\{\max_j \{C_j + q_j\} - LB_1\right\} = LB_1 \\
&\quad + \min_S \left\{\max_j \{C_j - (LB_1 - q_j)\}\right\} = LB_1 \\
&\quad + \min_S \left\{\max_j \{C_j - d_j\}\right\} = LB_1 \\
&\quad + \min_S \{L_{max}(OpenShop_i)\} = LB_1 \\
&\quad + L^*_{max}(OpenShop_i),
\end{aligned}$$

where d_j denotes the due date of job j, as defined above. Since $L^*_{max}(OpenShop_i)$ can be negative and we want to improve the previous lower bound, the new lower bound can be $LB_1 + \max\{0, L^*_{max}(OpenShop_i)\}$.□

In order to convert the problem to standard form, one can imagine that we want to add the maximum possible quantity to the previous lower bound (LB_1 in Eq. (A.3)).

A more efficient lower bound can be obtained by solving all the open shop problems $OpenShop_i$ for $i = 1, 2, ..., m$.

Corollary 1. Eq. (2) defines a lower bound for the stage shop problem.

$$\begin{aligned}
LB_2 &= LB_1 \\
&\quad + \max\left\{0, \max_{i=\{1,\ldots,m\}} \{L_{max}(OpenShop_i)\}\right\}.
\end{aligned} \tag{2}$$

Proof. LB_2 represents the maximum lower bound obtained by solving each $OpenShop_i$ and obviously, it is also a lower bound for the stage shop problem.□

Example 3.2. See Appendix B.

Because the problem $O_m\,|r_j|\,L_{max}$ is NP-hard [19], considering its preemptive counterpart which can be solved polynomially is useful. In the next subsection, the application of the preemptive open shop is discussed.

3.4. Application of preemptive open shop

The optimal objective function of problem $O_m\,|r_j, prmp|\,L_{\max}$ is a lower bound for the fitness value of problem $O_m\,|r_j|\,L_{\max}$, because the optimal solution to the latter is a feasible solution to the former. The preemptive open shop problem is modeled efficiently by Cho and Sahni [20]. This model is stated here as explained by Pinedo [19].

First, all due dates d_j are regarded as deadlines $\bar{d}_j$ and then, a linear programming model is used to realize whether or not we can find a feasible schedule in which each job is completed on time. Assume that $a_1 < a_2 < \cdots < a_{\nu+1}$ is the ordered assortment of the diverse release times r_j and deadlines $\bar{d}_j$. Therefore, ν intervals $[a_k, a_{k+1}]$ exist. Let $\delta_k = a_{k+1} - a_k$ be the length of interval k, J_j denote the operations of job j, and x_{ijk} represent a decision variable showing the time that the operation of job j performed by machine i is processed in interval k. Now, consider the following linear model:

$$\max \sum_{k=1}^{\nu}\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ijk}, \tag{3}$$

s.t.:

$$\sum_{i=1}^{m} x_{ijk} \le \delta_k \qquad j = 1, 2, ..., n \quad k = 1, 2, ..., \nu, \tag{4}$$

$$\sum_{j=1}^{n} x_{ijk} \le \delta_k \quad i = 1, 2, ..., m \quad k = 1, 2, ..., \nu, \tag{5}$$

$$\sum_{k=1}^{\nu} x_{ijk} = p_o \qquad o \in (I_i \cap J_j), \tag{6}$$

$$x_{ijk} \ge 0 \ \text{ if } \ r_j \le a_k \ \text{ and } \ \bar{d}_j \ge a_{k+1}, \tag{7}$$

$$x_{ijk} = 0 \ \text{ if } \ r_j \ge a_{k+1} \ \text{ and } \ \bar{d}_j \le a_k. \tag{8}$$

If a feasible solution exists to the LP problem, then there should be a schedule in which all jobs are completed before or at their deadlines. Otherwise, if the LP problem does not have any feasible solution, then each $\bar{d}_j$ can be replaced by $\bar{d}_j + L$, where L is a free parameter. It is worth noting that the smallest value of L for which there is a feasible solution is the minimum value of $L_{\max}$ that can be obtained by the original due dates. The solution to the preemptive problem cannot be achieved by LP. However, a solution is not required for the calculation of the new lower bound and obtaining $L_{\max}$ is sufficient.

Example 3.3. See Appendix B.

The gap between LB_1 and LB_2 may be much greater than one. Therefore, a procedure based on Newton method is developed to find the new lower bound more quickly. The pseudo code of this procedure is explained in Algorithm 1.

4. Three stochastic search methods

4.1. Encoding and decoding

Representation of the schedule is the depiction of a relative priority rule determining the job or activity that is selected next during the scheduling process [21]. Solution encoding in the stage shop problem should determine the sequence of operations for each job at each stage and machine. Furthermore, a random key representation is utilized to encode the solution. To extract the solution by the proposed solution encoding, the Largest Position Value (LPV) rule is employed. To better clarify the encoding scheme and the decoding technique (LPV rule), we designed an example of the stage shop problem with 2 jobs, 5 machines, and 10 operations. The detailed information about the instance is illustrated in Figure 1, in which the stages are identified by thick lines and the dummy operations are not shown. In addition, the data for the problem are shown in the cells with bold and colored text and the solution data are shown in the other cells. Figures 2 and 3 show the order of operations obtained by the LPV rule for each job at each stage and machine (based on the encoded solution in Figure 1), respectively.

Job 1	Operation	**1**	**2**	**3**	**4**	**5**
	Machine	**3**	**5**	**2**	**1**	**4**
	Random key	1.45	0.78	2.03	2.20	0.70
Job 2	Operation	**6**	**7**	**8**	**9**	**10**
	Machine	**5**	**4**	**3**	**2**	**1**
	Random key	2.73	0.12	1.64	0.42	0.96

Figure 1. Solution encoding of the stage shop problem.

Initialization
  Read $StageShopPr$
  Set $step$
/* $StageShopPr$ is the problem for which the lower bound should be computed */
$prvLB \leftarrow$ PrevLowerBound($StageShopPr$)
$newLB \leftarrow prvLB$
For $m = 1$ **to** $nmach$
  $flag \leftarrow False$
  Do
    LPSolve($StageShopPr, newLB, m$)

/* Solves LP for preemptive $OpenShop_m$ with $newLB$ and returns infeasible=true if LP is infeasible */
    If ($infeasible$)
      $flag \leftarrow True$
      $prvLB \leftarrow newLB$
      $newLB \leftarrow (newLB + step)$
    End if
  While ($infeasible$)
  If ($flag$)
    Do
      LPSolve($StageShopPr, \frac{newLB+prvLB}{2}, m$)
      If (! $infeasible$)
        $newLB \leftarrow \frac{newLB+prvLB}{2}$}
      Else
        $prvLB \leftarrow \frac{newLB+prvLB}{2}$
      End if
    While ($newLB$! = $prvLB + 1$)
  End if
End For
Return $newLB$

Algorithm 1. The procedure for finding the new lower bouns.

Stage	Alternative process plans	Selected by LPV
1	1→2	✓
	2→1	
2	3→4→5	
	3→5→4	
	4→3→5	✓
	4→5→3	
	5→3→4	
	5→4→3	

(a)

Stage	Alternative process plans	Selected by LPV
1	6→7→8	
	6→8→7	✓
	7→6→8	
	7→8→6	
	8→6→7	
	8→7→6	
2	9→10	
	10→9	✓

(b)

Figure 2. Alternative process plans for (a) job 1 and (b) job 2.

For instance, Figure 2(a) shows that at the first stage of job 1, operation 1 is processed before operation 2 because the random key of operation 1 is greater than that of operation 2 (1.45 versus 0.78). Therefore, $1 \rightarrow 2$ is selected as the preferred process plan for job 1 at the first stage. Moreover, since operation 8 has a greater random key than operation 1 (1.64 versus 1.45), it will precede operation 1 in processing on machine 3 (see Figure 3).

Machine	Sequence of the operations on each machine
1	4→10
2	3→9
3	8→1
4	5→7
5	6→2

Figure 3. Permutation representation on machines.

4.2. The Harmony Search (HS) algorithm

HS is a metaheuristic algorithm which imitates the music improvisation process to develop a strong optimization strategy [22,23]. In the music improvisation process, musicians improvise the pitches on the instruments to find a perfect state of harmony. In the HS algorithm, each solution is called a "harmony," which is represented by a D-dimensional real vector. The general steps of the HS algorithm are as follows:

- **Initialize the problem and HS parameters:** The aim of this step is to specify the parameters of the HS algorithm, including the Harmony Memory

Size (HMS), Harmony Memory Considering Rate (HMCR), Pitch Adjusting Rate (PAR), and the Number of Improvisations (NI).

- **Initialize the Harmony Memory (HM):** In this step, the initial HM is generated using a uniform distribution as Eq. (9):

$$x_{id} = x_d^{\min} + rand(0,1) \times \left(x_d^{\max} - x_d^{\min}\right)$$
$$d = 1, ..., D; \quad i = 1, ..., HMS. \tag{9}$$

- **Improvise a new harmony:** Three main rules are used to improvise a new harmony vector $x' = (x'_1, x'_2, \ldots, x'_D)$ in the HS algorithm, including memory consideration, pitch adjustment, and random selection.

  For each $d \in \{1, ..., D\}$, if $\text{rand}(0,1) < HMCR$ (memory consideration), the new harmony vector is $x'_d = x_{id}$ where $i = rand(1, 2, \ldots, HMS)$; otherwise, if $rand(0,1) < PAR$ (pitch adjustment), the new harmony vector is $x'_d = x'_d + bw \times (rand(0,2) - 1)$ where bw is an arbitrary distance bandwidth which is often selected as $bw_d = 0.01 \times \left(x_d^{\max} - x_d^{\min}\right)$. Otherwise (with random selection), the new harmony vector is $x'_d = x_d^{\min} + rand(0,1) \times \left(x_d^{\max} - x_d^{\min}\right)$.

- **Update HM:** If the fitness of the generated harmony vector is better than that of the worst harmony, then the worst harmony in the HM should be replaced by the generated harmony $x' = (x'_1, x'_2, \ldots, x'_D)$.

- Check the stopping criterion: The maximum NI is the criterion to stop the HS algorithm [24–26].

The pseudo code of the HS algorithm is described in Algorithm 2.

Initialization
  Set *parameters HMS, HMCR, PAR, and NI;*
  t=0;
  Generate *an initial population with HMS size*
For *i*=1 **to** *HMS*
  Evaluate *the fitness function of each harmony vector*
End for
For *t*=1 **to** *NI*
  Generate *new solution x*
    For *d* =1 **to** *D*
      If (rand(0,1) < *HMCR*) (*memory consideration*)
        $x'_d = x_{id}$ *where* $i = \text{rand}(1,2,\ldots,HMS)$
      Else if (rand(0,1) < *PAR*) (*pitch adjustment*)
        $x'_d = x'_d + bw \times (rand(0,2) - 1)$
      Else (*random selection*)
        $x'_d = x_d^{min} + \text{rand}(0,1) \times (x_d^{max} - x_d^{min})$
      End if
    End for
  If $(f(x') < f(x^{worst}))$
    $x^{worst} = x'$
  End if
End for
Return *the best schedule*

Algorithm 2. Pseudo code for harmony search algorithm.

4.3. The firefly (FF) algorithm

The FF algorithm is a novel algorithm inspired by the social behavior of fireflies. This algorithm is based on the flashing characteristics of fireflies and introduced by Yang [27]. Attracting other fireflies is the main role of the flash of a firefly.

Three main rules of the FF algorithm are explained as follows:

1. All fireflies are unisex: The flashing behavior is usually used to send signals to the opposite sex. However, any firefly can attract the other fireflies in this algorithm.
2. The brightness of a firefly determines its attractiveness. The less bright firefly will be attracted by the brighter one as the apparent brightness of a firefly is proportional to its distance from the other firefly. If both have the same brightness, they will move randomly.
3. The brightness of a firefly is equivalent to the value of the objective function. In minimization problems, the brightness of a firefly is reversely proportional to the value of the objective function. On the other hand, in a maximization problem, the brightness of a firefly is directly proportional to the value of the objective function. Important issues about the FF algorithm are given as follows:

   - **Initialization:** The initial positions of agents in the search space are determined randomly as:

$$x_{id}^0 = x_d^{\min} + rand(0,1) \times \left(x_d^{\max} - x_d^{\min}\right),$$
$$i = 1, ..., N; \quad d = 1, ..., D, \tag{10}$$

     where x_{id}^0 is the d'th dimension of firefly i in its initial position.

   - **Attractiveness:** In order to generalize the main form of the attractiveness function $\beta(r)$, any monotonically decreasing functions such as Eq. (11) can be considered:

$$\beta(r) = \beta_0 \exp(-\gamma r^m); \quad m \geq 1, \tag{11}$$

     where r is the distance between two fireflies. Also, β_0 and γ are the initial attractiveness of a firefly and the absorption coefficient, respectively.

   - **Distance between fireflies:** The distance between any two fireflies i and j (at positions x_i and x_j) can be calculated as follows:

$$r_{ij} = x_i - x_j = \sqrt{\sum_{d=1}^{D} (x_{id} - x_{jd})^2};$$
$$i \neq j; \ i = 1, ..., N; \ j = 1, ..., N. \tag{12}$$

Initialization
  Create *an initial population* (N *fireflies*)
  Define *light intensity* I_i *as the value of the objective function of solution* x_i*,namely* $f(x_i)$
  Define *absorption coefficient* γ
While (*iteration* < *Max iteration*)
For $i = 1$ **to** N (*all* N *fireflies*)
  For $j = 1$ **to** N (*all* N *fireflies*)
    If $(I_i < I_j)$
      Move *firefly i towards firefly j*
    End if
    Change *attractiveness with distance r using* $\exp(-\gamma r^2)$
    Calculate *the objective function of the new solutions*
    Update *light intensity*
  End for
End for
  Find *the current best*
  iteration = iteration + 1
End while
Return *the best solution*

Algorithm 3. Pseudo code for firefly algorithm.

- **Movement of firefly:** The movement of firefly i, attracted to the brighter firefly j, can be calculated as:

$$x_i = x_i + \beta(r)(x_j - x_i) + \alpha(rand(0,1) - 0.5);$$
$$i \neq j;\ i = 1, ..., N;\ j = 1, ..., N. \quad (13)$$

The second term of the equation shows how a firefly moves toward the other fireflies and the third term represents the random movement of a firefly. The coefficient $\alpha \in [0, 1]$ is a random parameter determined by the problem of interest, while $rand$ is a random number obtained by a uniform distribution in the space [0,1] [28–30].

The main steps of the FF algorithm are similar to those of the Particle Swarm Optimization (PSO). The pseudo-code of FF can be summarized as given in Algorithm 3.

4.4. Water Wave Optimization (WWO) algorithm

WWO is an innovative meta-heuristic algorithm inspired by the shallow water wave theory to solve global optimization problems [31]. In WWO algorithm, each solution can be considered as a "wave" with a height h and a length λ. Furthermore, the search space is similar to the seabed area and the solution fitness can be calculated according to the seabed depth (i.e., the shorter the distance to the still water level, the higher the fitness value). In order to solve a maximization problem with objective function f, this algorithm firstly initializes a set of waves ($h = h_{\max}$, $\lambda = 0.5$). Afterwards, three operators, namely propagation, refraction, and breaking, are used during the problem-solving process.

- **Propagation:** In this operator, in each generation, wave X propagates only one time to generate a new wave X' by changing the dimension of the original wave X as:

$$X'(d) = X(d) + rand(-1, 1).\lambda.L(d), \quad (14)$$

where rand $(-1, 1)$ is a random number generated uniformly in the range of $[-1, 1]$ and $L(d)$ is considered as the length of the dth dimension in the search area ($1 \leq d \leq n$). If the new determined position is not feasible, its position will be changed to a random value in the range of $[-1, 1]$.

After the propagation and generating the new wave X', its fitness will be calculated and compared with the original wave X. If the fitness of the offspring wave X' is higher than that of the original wave X, i.e., $f(X') > f(X)$, wave X' will be the substitute for wave X and its height is reset to $h_{\max}$; otherwise, the original wave X will be maintained while its height will be reduced to one.

As seen in Figure 4, the waves in deep water have wavelengths longer than the ones in the shallow water. However, the waves in shallow water have longer heights than the ones in the deep water. Moving of a wave from the deep water (with low fitness value) toward the shallow water (with high fitness value) will decrease its wavelength. WWO algorithm mimics such phenomena to determine the wavelength of each wave X after each generation as:

$$\lambda = \lambda.\alpha^{-(f(x)-f_{\min}+\varepsilon)/(f_{\max}-f_{\min}+\varepsilon)}, \quad (15)$$

```
Set the parameters of WWO (n-Pop, h_max, α, β and k_max)
Generate an initial population P of n waves randomly
While stop criterion is not verified do
    for each wave X ∈ P do
        Propagate wave X to a new wave X' using Eq. (4.6);
        Calculate the fitness value wave X', i.e. f(X');
        if f(X') > f(X) then
            if f(X') > f(X_best) then
                Break wave X' into a train of solitary waves by using Eq. (4.8);
                Update wave X_best with wave X';
            End if
            Replace wave X with wave X';
        Else
            X.h = X.h−1;
            if X.h = 0 then
                Refract wave X to a new wave X' using Eqs. (4.9) and (4.10);
            End if
        End if
        Update the wavelengths using Eq. (4.7);
    End for
End while
Return the best solution;
```

Algorithm 4. Pseudo code for Water Wave Optimization (WWO).

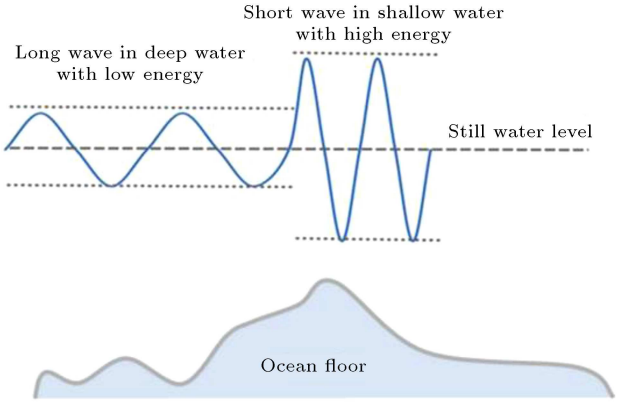

Figure 4. Illustration of the various wave shapes in deep and shallow water.

where the maximum and minimum fitness values in the current population are represented by $f_{\max}$ and $f_{\min}$, respectively. Also, α denotes the length reduction coefficient of the wave and ε is a small positive constant to prevent zero-division-error.

- **Breaking:** This operator is used to conduct a local search around the current best wave. If the fitness value of a new wave X is higher than that of the current best wave, it will be broken into a set of solitary waves through breaking operator. In detail, this operator first selects k dimensions randomly (where k is a random number chosen between 1 and an algorithm parameter $k_{\max}$) and then, generates a solitary wave X' by shifting from its original position at each selected dimension d as:

$$X'_d = X_d + norm\,(0,1)\,.\beta L\,(d)\,, \tag{16}$$

where β is the breaking coefficient. Experimentally, it is recommended by Zheng [31] to set k to $\min(12, n/2)$ in which n is the dimension of the problem. If there is not any wave among solitary waves whose fitness value is higher than that of wave X, we maintain wave X.

- **Refraction:** The refraction operator moves only the motionless waves (whose heights diminish to zero) to new positions. The position after the refraction can be calculated by Eq. (17). That is, a new wave X' is generated according to the position of both the original wave X and the best wave X_{best} at each dimension d.

$$X'_d = norm\left(\frac{X_{best,d} + X_d}{2}, \frac{X_{best,d} - X_d}{2}\right), \tag{17}$$

where $\text{norm}(\mu, \sigma)$ denotes a normal random number with a mean of μ and a standard deviation of σ. After each refraction operation, the wave height of X' is reinitialized to $h_{\max}$ and its wavelength can be calculated by:

$$\lambda' = \lambda \frac{f\,(X)}{f\,(X')}. \tag{18}$$

As in [31], the pseudo-code of the WWO algorithm is described by Algorithm 4.

5. Computational Results

All the experiments of this research were implemented in Visual C++ 2010 and performed on a laptop with 2.5 GHz CPU (Core i5-3210M).

5.1. The problem instances

The problem instances (which are called ATA01-50) used in this paper are exactly the 50 instances of Nasiri and Kianfar [8]. These problems are based on the well-known Taillard's benchmarks (which are available on his website [32]) for the job shop scheduling [33] and all processing times and their respective machines are the same. The only data generated by Nasiri and Kianfar [8] are the structures of the stages, which are available in Supplementary Material, Appendix C. To know more about the way of building the structure, please refer to Nasiri and Kianfar [8].

5.2. Lower bounds for the stage shop problem instances

The only parameter to be set is *step*, which is set to 30. Table 1 displays the new lower bound for the stage shop problem instances. The results show that the lower bounds are improved in 26 out of 50 problems. It seems that the improvements are more significant when the problem sizes are square (number of jobs and number of machines are the same).

5.3. Setting parameters

The values of parameters have a significant effect on the efficiency of the algorithm. If they are not correctly set, getting the appropriate results will become challenging. Therefore, in this section, to improve the behavior of the suggested algorithms, the values for the parameters of each algorithm are tuned using Taguchi design method. Taguchi method is one of best approaches to the calibration of input parameters for meta-heuristic solution methods (see [11,34]). This method exploits orthogonal arrays to manage and adjust experiences in the presence of a group of decision variables or factors. The aim of this method is to minimize the effect of noise and to obtain the optimal level of signal factors. To get more information about Taguchi method, the interested readers can refer to [35].

As already mentioned, there are three parameters in the HS algorithm, including HMS, HMCR, and PAR. The FF algorithm has four parameters, including β_0, $n - Pop$, γ, and α. In addition, the WWO algorithm has five parameters, including $n - Pop$, $h_{\max}$, α, β, and $k_{\max}$. The considered levels for the parameters of

Table 1. The new lower bound for ATA problem instances.

Problem	Size	LB_1	LB_2	Problem	Size	LB_1	LB_2
ATA01	15 × 15	977	1048	ATA26	20 × 20	1207	1363
ATA02	15 × 15	942	1097	ATA27	20 × 20	1331	1331
ATA03	15 × 15	921	1030	ATA28	20 × 20	1269	1269
ATA04	15 × 15	911	1035	ATA29	20 × 20	1267	1363
ATA05	15 × 15	940	949	ATA30	20 × 20	1212	1246
ATA06	15 × 15	1030	1074	ATA31	30 × 15	1764	1764
ATA07	15 × 15	951	1032	ATA32	30 × 15	1774	1774
ATA08	15 × 15	963	1096	ATA33	30 × 15	1729	1729
ATA09	15 × 15	982	1054	ATA34	30 × 15	1828	1828
ATA10	15 × 15	933	1065	ATA35	30 × 15	1729	1731
ATA11	20 × 15	1139	1164	ATA36	30 × 15	1777	1777
ATA12	20 × 15	1251	1251	ATA37	30 × 15	1771	1771
ATA13	20 × 15	1178	1178	ATA38	30 × 15	1673	1673
ATA14	20 × 15	1130	1132	ATA39	30 × 15	1641	1654
ATA15	20 × 15	1148	1148	ATA40	30 × 15	1602	1602
ATA16	20 × 15	1181	1181	ATA41	30 × 20	1830	1830
ATA17	20 × 15	1257	1287	ATA42	30 × 20	1761	1761
ATA18	20 × 15	1153	1246	ATA43	30 × 20	1694	1694
ATA19	20 × 15	1202	1202	ATA44	30 × 20	1787	1787
ATA20	20 × 15	1216	1216	ATA45	30 × 20	1731	1738
ATA21	20 × 20	1217	1322	ATA46	30 × 20	1856	1856
ATA22	20 × 20	1240	1371	ATA47	30 × 20	1690	1690
ATA23	20 × 20	1185	1236	ATA48	30 × 20	1744	1818
ATA24	20 × 20	1271	1402	ATA49	30 × 20	1758	1758
ATA25	20 × 20	1256	1385	ATA50	30 × 20	1674	1674

Table 2. Tuned values for the parameters of the proposed algorithms.

Harmony search	**HMS**	**HMCR**	**PAR**		
	30	0.3	0.95		
Firefly	***n*-Pop**	α	β	Γ	
	50	0.9	0.1	5	
Water wave optimization	***n*-Pop**	α	$k_{\max}$	$h_{\max}$	**B**
	150	1.026	12	6	Linearly decreases from 0.25 to 0.0002

each algorithm (based on similar researches as well as trial and error method) and their analysis are shown in Supplementary Data, Appendix C. The tuned values of the parameters for each algorithm are shown in Table 2.

5.4. Comparison of the algorithms

In this section, the performance of the developed metaheuristic algorithms, namely HS, FF, and WWO, is evaluated in comparison. Each of the 50 problem instances mentioned in Subsection 5.1 is solved 30 times by each algorithm. To make a fair judgment, the same termination criterion (10^5 function evaluations) is used for all the three algorithms. For each run of a problem instance, the quality of the metaheuristic methods is evaluated by comparing their best fitness values with the lower bound using the Relative Gap (RG) measure. The RG is calculated by $RG = (OFV_{Meta} - LB_2)/(LB_2)$, where OFV_{Meta} and LB_2 are the objective function value of the best solution obtained by the metaheuristic algorithm and the lower bound obtained by the presented approach, respectively.

The computational results are summarized in Table 3, in which "UB" indicates the Upper Bounds derived using a pure solver (CPLEX in GAMS software) in 3600s. Also, "Best," "M_{av}," "ARG," and "CPU_{av} (s)" represent the best makespan, the average makespan, the average of the RGs, and the average of CPU times achieved by each algorithm over 30 runs for each problem, respectively. As seen in Table 3, in HS, FF, and WWO algorithms, M_{av} is lower than the upper bound obtained by solving the MIP model using a pure solver in 34, 35, and 21 out of 50 instances, respectively. It should be noted that the best upper bound found for each problem (over all experiments) is distinguished by bold letters. Table 3 shows that all of the bold numbers are related to the metaheuristic algorithms.

In addition, the sensitivity analysis of CPU_{av} based on the change in problem size for three metaheuristic algorithms is illustrated in Figure 5. As far as the CPU time is concerned, the FF algorithm outperforms WWO in all the problems. FF also shows better performance than HS in all the problems except for those with the largest sizes.

Figure 6 presents a graph depicting the convergence behavior of FF, HS, and WWO algorithms for problem ATA25. As it can be seen, FF and WWO have faster convergence rates than HS.

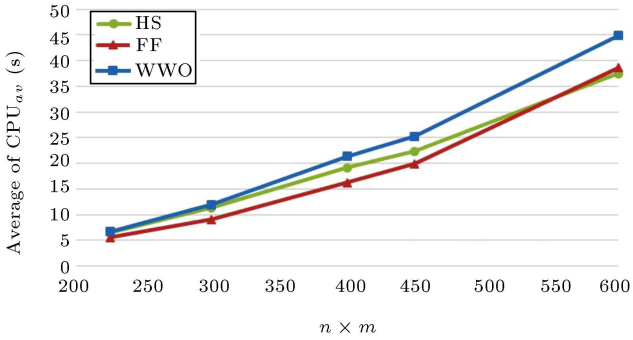

Figure 5. Average CPU time versus the size of problem instance.

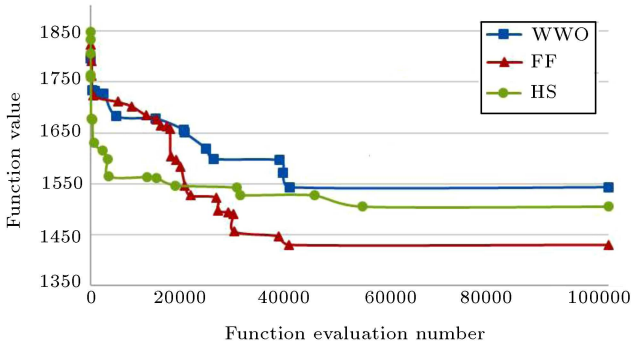

Figure 6. The convergence curves of algorithms for problem ATA25.

A statistical procedure is applied to comparing the performance of the developed algorithms in terms of the above-defined RG measure. We employ two statistical approaches, namely single-problem and multiple-problem analyses.

5.4.1. Single-problem analysis

In the single-problem analysis, the proposed algorithms are compared with each other over each problem instance independently using the parametric/non-parametric statistical tests [36]. In order to carry out a parametric test, it is required to check three conditions, namely independence, normality, heteroscedasticity [37]. In the current research, the independence condition is satisfied for the results of each algorithm as

Table 3. Summary of the results of the experiments in 10^5 evaluations.

Problem	UB	HS				FF				WWO			
		Best	M_{av}	ARG	CPU_{av}(s)	Best	M_{av}	ARG	CPU_{av}(s)	Best	M_{av}	ARG	CPU_{av}(s)
ATA01	1191	**1112**[a]	1153.0[b]	0.100	6.3	1114[a]	1163.1[b]	0.110	5.7	1192	1219.3	0.163	6.5
ATA02	1171	1160[a]	1194.7	0.089	6.3	**1154**[a]	1210.1	0.103	5.6	1218	1264.7	0.153	6.4
ATA03	1150	**1137**[a]	1176.7	0.142	6.4	1140[a]	1186.2	0.152	5.7	1181	1232.6	0.197	6.3
ATA04	1101	**1060**[a]	1098.0[b]	0.061	6.3	1078[a]	1113.7	0.076	5.6	1108	1168.7	0.129	6.4
ATA05	1089	**1081**[a]	1129.4	0.190	6.3	1090[a]	1142.7	0.204	5.6	1132	1192.1	0.256	6.7
ATA06	1197	**1168**[a]	1210.9	0.127	6.2	1184[a]	1215.4	0.132	5.5	1215	1265.8	0.179	6.8
ATA07	1141	**1109**[a]	1150.6	0.115	6.3	1127[a]	1166.5	0.130	5.7	1167	1227.2	0.189	6.9
ATA08	1169	**1089**[a]	1133.7[b]	0.034	6.4	1114[a]	1151.8[b]	0.051	5.3	1153[a]	1211.0	0.105	6.8
ATA09	1195	**1168**	1206.4	0.145	6.4	1183[a]	1213.8	0.152	5.3	1200	1268.3	0.203	6.9
ATA10	1140	**1123**[a]	1150.7	0.080	7.1	1131[a]	1167.0	0.096	5.3	1157	1213.3	0.139	6.9
ATA11	1357	**1300**[a]	1352.6[b]	0.162	10.8	1302[a]	1329.9[b]	0.142	9.1	1345	1399.6	0.202	11.1
ATA12	1337	1361	1396.4	0.116	11.9	**1325**[a]	1370.6	0.096	9.0	1395	1457.5	0.165	11.2
ATA13	1368	1332[a]	1367.8[b]	0.161	12.2	**1286**[a]	1331.7[b]	0.130	9.2	1365[a]	1414.4	0.201	11.3
ATA14	1298	1273[a]	1309.1	0.156	12.0	**1240**[a]	1290.5[b]	0.140	9.0	1322	1374.2	0.214	11.2
ATA15	1322	1309[a]	1353.9	0.179	12.5	**1276**[a]	1338.1	0.166	9.3	1326	1394.0	0.214	11.8
ATA16	1324	1341	1381.7	0.170	10.9	**1300**[a]	1366.5	0.157	8.8	1376	1452.2	0.230	12.5
ATA17	1434	1429[a]	1462.8	0.137	10.8	**1398**[a]	1440.6	0.119	9.1	1453	1525.8	0.186	12.5
ATA18	1429	1408[a]	1433.3	0.150	10.8	**1366**[a]	1413.6[b]	0.135	9.0	1428[a]	1485. 7	0.192	12.7
ATA19	1317	1328	1363.4	0.134	11.0	**1304**[a]	1348.9	0.122	9.1	1342	1415.1	0.177	12.6
ATA20	1322	1343	1372.0	0.128	10.8	**1294**[a]	1346.1	0.107	9.1	1358	1415.7	0.164	12.6
ATA21	1739	1438[a]	1488.0[b]	0.126	19.4	**1394**	1477.8[b]	0.118	16.1	1480[a]	1557.4[b]	0.178	21.3
ATA22	1621	1486[a]	1533.6[b]	0.119	19.4	**1462**[a]	1506.1[b]	0.099	16.3	1503[a]	1580.5[b]	0.153	21.2
ATA23	1483	1376[a]	1404.2[b]	0.136	19.0	**1330**[a]	1376.8[b]	0.114	16.2	1396[a]	1478.1[b]	0.196	21.4
ATA24	1805	1467[a]	1528.3[b]	0.090	19.4	**1451**[a]	1512.6[b]	0.079	16.0	1544[a]	1617.0[b]	0.153	21.2
ATA25	1515	1437[a]	1483.5[b]	0.071	19.0	**1411**[a]	1457.4[b]	0.052	16.2	1513[a]	1564.0	0.129	21.5
ATA26	1539	1435[a]	1483.1[b]	0.088	19.1	**1420**[a]	1465.4[b]	0.075	16.3	1495[a]	1569.4	0.151	21.5
ATA27	1692	1512[a]	1533.7[b]	0.152	18.3	**1456**[a]	1509.8[b]	0.134	16.4	1575[a]	1623.7[b]	0.220	21.2
ATA28	1584	1417[a]	1446.0[b]	0.139	19.4	**1382**[a]	1423.3[b]	0.122	16.1	1447[a]	1517.7[b]	0.196	21.3
ATA29	1721	1495[a]	1536.4[b]	0.127	19.5	**1449**[a]	1500.0[b]	0.100	16.5	1544[a]	1597.6[b]	0.172	21.5
ATA30	1507	1397[a]	1432.2[b]	0.149	18.9	**1352**[a]	1397.5[b]	0.122	16.2	1421[a]	1506.8[b]	0.209	21.1
ATA31	1916	1897[a]	1913.1[b]	0.085	23.3	**1823**[a]	1862.4[b]	0.056	20.1	1902[a]	1982.5	0.124	25.7
ATA32	2037	1907[a]	1940.3[b]	0.094	23.3	**1843**[a]	1887.8[b]	0.064	20.0	1949[a]	2017.1[b]	0.137	25.9
ATA33	1984	1939[a]	1964.3[b]	0.136	22.7	**1836**[a]	1901.0[b]	0.099	19.9	1986	2038.7	0.179	25.3
ATA34	1992	1937[a]	1984.8[b]	0.086	23.1	**1870**[a]	1923.8[b]	0.052	20.1	1964[a]	2056.8	0.125	25.3
ATA35	2026	1886[a]	1914.7[b]	0.106	23.2	**1827**[a]	1862.8[b]	0.076	19.7	1884[a]	1988.1[b]	0.149	25.4
ATA36	1966	1914[a]	1949.6[b]	0.097	23.3	**1859**[a]	1894.5[b]	0.066	19.9	1961[a]	2042.3	0.149	25.7
ATA37	1933	1888[a]	1927.4[b]	0.088	21.7	**1831**[a]	1880.3[b]	0.062	20.3	1924[a]	1999.6	0.129	25.7
ATA38	1767	1796	1824.0	0.090	21.7	**1752**[a]	1782.7	0.066	19.8	1832	1894.6	0.132	25.2
ATA39	2054	1844[a]	1881.5[b]	0.138	21.0	**1779**[a]	1818.3[b]	0.099	19.4	1939[a]	1988.2[b]	0.202	24.3
ATA40	1923	1842[a]	1883.3[b]	0.176	20.4	**1760**[a]	1833.0[b]	0.144	19.6	1909[a]	1991.3	0.243	23.9
ATA41	2318	2017[a]	2048.4[b]	0.119	37.2	**1910**[a]	1993.9[b]	0.090	36.8	2053[a]	2150.2[b]	0.175	46.0
ATA42	2147	1913[a]	1953.6[b]	0.109	37.9	**1835**[a]	1894.3[b]	0.076	36.0	1938	2033.9[b]	0.155	44.7
ATA43	2143	1872[a]	1906.1[b]	0.125	37.5	**1811**[a]	1846.9[b]	0.090	36.4	1900[a]	1993.2[b]	0.177	45.0
ATA44	2184	1958[a]	2006.3[b]	0.123	37.4	**1897**[a]	1950.3[b]	0.091	36.4	2055[a]	2128.8[b]	0.191	44.8
ATA45	2280	1981[a]	2027.6[b]	0.167	37.7	**1911**[a]	1963.2[b]	0.130	36.5	2039[a]	2108.4[b]	0.213	44.7
ATA46	2416	1932[a]	1972.4[b]	0.063	37.3	**1863**[a]	1914.4[b]	0.031	37.1	2006[a]	2076.1[b]	0.119	43.7
ATA47	2249	1908[a]	1948.8[b]	0.153	37.1	**1835**[a]	1888.8[b]	0.118	39.1	1929[a]	2044.9[b]	0.210	45.0
ATA48	2260	1943[a]	1978.5[b]	0.088	37.6	**1884**[a]	1923.4[b]	0.058	42.4	1999[a]	2083.0[b]	0.146	44.5
ATA49	2103	1893[a]	1941.6[b]	0.104	37.7	**1848**[a]	1892.3[b]	0.076	42.8	1971[a]	2045.9[b]	0.164	44.9
ATA50	2272	1965[a]	2000.5[b]	0.195	37.1	**1906**[a]	1952.0[b]	0.166	42.4	1985[a]	2068.2[b]	0.236	45.7
Average			1584.7	0.122	19.3		1556.0	0.105	17.9		1658.8	0.175	22.01

[a] The best makespan found by the algorithm is better than the upper bound obtained by solving the MIP model.

[b] The average makespan of 30 runs of the algorithm is better than the upper bound obtained by solving the MIP model.

Table 4. Comparison of Harmony Search (HS) and firefly (FF): p-values obtained by Wilcoxon test/paired t-test, difference between Average Relative Gap (ARG) and the result of statistical analysis in the single-problem analysis.

Problem	ATA01*	ATA02	ATA03	ATA04	ATA05	ATA06	ATA07	ATA08	ATA09	ATA10
p-value	0.171	0.009	0.072	0.011	0.061	0.346	0.019	0.009	0.182	.000
Difference of ARG	–0.010	–0.014	–0.009	–0.015	–0.014	–0.004	–0.015	–0.016	–0.007	–0.015
Result	ND	HoF	ND	HoF	ND	ND	HoF	HoF	ND	HoF
Problem	ATA11	ATA12*	ATA13	ATA14*	ATA15	ATA16	ATA17	ATA18*	ATA19*	ATA20*
p-value	0.000	0.000	0.000	0.005	0.001	0.037	0.000	0.004	0.02	0.001
Difference of ARG	0.020	0.021	0.031	0.016	0.014	0.013	0.017	0.016	0.012	0.021
Result	FoH	FoH	FoH	FoH	FoH	FoH	FoH	FoH	FoH	FoH
Problem	ATA21	ATA22	ATA23*	ATA24*	ATA25	ATA26	ATA27*	ATA28*	ATA29*	ATA30
p-value	0.112	0.000	0.000	0.026	0.007	0.013	0.000	0.001	0.000	0.000
Difference of ARG	0.008	0.020	0.022	0.011	0.019	0.013	0.018	0.018	0.027	0.028
Result	ND	FoH	FoH	FoH	FoH	FoH	FoH	FoH	FoH	FoH
Problem	ATA31*	ATA32	ATA33*	ATA34	ATA35	ATA36	ATA37*	ATA38*	ATA39	ATA40*
p-value	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Difference of ARG	0.029	0.030	0.037	0.033	0.030	0.031	0.027	0.025	0.038	0.031
Result	FoH	FoH	FoH	FoH	FoH	FoH	FoH	FoH	FoH	FoH
Problem	ATA41*	ATA42	ATA43	ATA44*	ATA45	ATA46	ATA47	ATA48	ATA49	ATA50
p-value	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Difference of ARG	0.030	0.034	0.035	0.031	0.037	0.031	0.036	0.030	0.028	0.029
Result	FoH	FoH	FoH	FoH	FoH	FoH	FoH	FoH	FoH	FoH

Note: ND: There is no statistical difference between algorithms; FoH: Firefly outperforms harmony search;

HOF: harmony search outperforms firefly.

*: At least one of the conditions of parametric tests is not satisfied; therefore, Wilcoxon signed-ranks test is used.

they are obtained by independent runs with randomly generated initial populations. A Kolmogorov-Smirnov test is used to carry out the normality analysis at the significance level of $\alpha = 0.05$. It is well-known that p-values greater than the level of significance α mean the fulfillment of the normality condition. The p-values obtained by the Kolmogorov-Smirnov test are included in Supplementary Data, Appendix C. Based on the results, the normality condition is satisfied in most of the cases. More precisely, in HS, FF, and WWO algorithms, the normal distribution is accepted in 48, 47, and 45 out of 50 cases, respectively. We use the Levene's test to determine whether homoscedasticity condition is verified or not. The results of the Levene's test are given in Supplementary Data, Appendix C.

If all the required conditions utilizing the parametric tests are simultaneously met in a statistical hypothesis test, a paired t-test can be used (as in this paper) for the pairwise comparison of the algorithms with respect to the RG measure. Otherwise, if any of the three conditions is not met, a non-parametric test is more reliable than the parametric test [37]. In this case, the Wilcoxon signed-rank test as a non-parametric test is utilized. In fact, these tests (Wilcoxon or paired t-test) are conducted to check whether there is a significant difference between the two algorithms under comparison in terms of the RG measure in each problem instance ($H_o : \mu_{alg1\mathrm{RG}} = \mu_{alg2\mathrm{RG}}$).

Tables 4–6 represent the p-values obtained by Wilcoxon test/paired t-test, Average RG (ARG) difference between the two algorithms under comparison, and the results of the single-problem analysis. If the p-value is less than 0.05, the null hypothesis is rejected at the level of significance $\alpha = 0.05$. Therefore, we can express that the two algorithms are statistically different from each other based on the comparison metric of RG for the considered problem instance. Thus, every algorithm which has a lower ARG outperforms the other one in that problem instance.

The results of Tables 3–6 allow us to make the following conclusions:

- As far as the RG is concerned, the HS algorithm has a better performance than the FF algorithm in five out of the first 10 problem instances, while as the size of problem instances grows, FF outperforms HS in all the next 40 problem instances except for ATA21;
- Both FF and HS algorithms have statistically con-

Table 5. Comparison of Harmony Search (HS) and Water Wave Optimization (WWO): p-values obtained by Wilcoxon test/paired t-test, difference between Average Relative Gap (ARG) and the result of statistical analysis in the single-problem analysis.

Problem	ATA01	ATA02	ATA03*	ATA04	ATA05*	ATA06	ATA07	ATA08	ATA09	ATA10
p-value	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Difference of ARG	–0.063	–0.064	–0.054	–0.068	–0.066	–0.051	–0.074	–0.070	–0.059	–0.059
Result	HoW	HoW	HoW	HoW	HoW	HoW	HoW	HoW	HoW	HoW
Problem	ATA11*	ATA12*	ATA13*	ATA14*	ATA15	ATA16*	ATA17*	ATA18*	ATA19*	ATA20*
p-value	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Difference of ARG	–0.040	–0.049	–0.040	–0.057	–0.035	–0.060	–0.049	–0.042	–0.043	–0.036
Result	HoW	HoW	HoW	HoW	HoW	HoW	HoW	HoW	HoW	HoW
Problem	ATA21*	ATA22	ATA23*	ATA24*	ATA25	ATA26*	ATA27*	ATA28*	ATA29	ATA30*
p-value	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Difference of ARG	–0.052	–0.034	–0.060	–0.063	–0.058	–0.063	–0.068	–0.057	–0.045	–0.060
Result	HoW	HoW	HoW	HoW	HoW	HoW	HoW	HoW	HoW	HoW
Problem	ATA31*	ATA32*	ATA33*	ATA34*	ATA35*	ATA36*	ATA37*	ATA38*	ATA39*	ATA40*
p-value	0.000	0.000	0.000	0.000	0.000	0.000	.0000	0.000	0.000	0.000
Difference of ARG	–0.039	–0.043	–0.043	–0.039	–0.042	–0.052	–0.041	–0.042	–0.065	–0.067
Result	HoW	HoW	HoW	HoW	HoW	HoW	HoW	HoW	HoW	HoW
Problem	ATA41*	ATA42*	ATA43*	ATA44*	ATA45*	ATA46*	ATA47*	ATA48*	ATA49	ATA50*
p-value	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Difference of ARG	–0.056	–0.046	–0.051	–0.069	–0.046	–0.056	–0.057	–0.058	–0.059	–0.040
Result	HoW	HoW	HoW	HoW	HoW	HoW	HoW	HoW	HoW	HoW

Note: HoW: Harmony search outperforms water wave optimization algorithm.

*: At least one of the conditions of parametric tests is not satisfied; therefore, Wilcoxon signed-ranks test is used.

Table 6. Comparison of firefly (FF) and Water Wave Optimization (WWO): p-values obtained by Wilcoxon test/paired t-test, difference between Average Relative Gap (ARG) and the result of statistical analysis in the single-problem analysis.

Problem	ATA01*	ATA02	ATA03*	ATA04	ATA05*	ATA06	ATA07	ATA08	ATA09	ATA10*
p-value	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Difference of ARG	–0.054	–0.050	–0.045	–0.053	–0.052	–0.047	–0.059	–0.054	–0.052	–0.044
Result	FoW	FoW	FoW	FoW	FoW	FoW	FoW	FoW	FoW	FoW
Problem	ATA11	ATA12	ATA13*	ATA14	ATA15	ATA16	ATA17*	ATA18	ATA19	ATA20*
p-value	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Difference of ARG	–0.060	–0.069	–0.070	–0.074	–0.049	–0.073	–0.066	–0.058	–0.055	–0.057
Result	FoW	FoW	FoW	FoW	FoW	FoW	FoW	FoW	FoW	FoW
Problem	ATA21	ATA22	ATA23	ATA24	ATA25	ATA26*	ATA27	ATA28*	ATA29	ATA30
p-value	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Difference of ARG	–0.060	–0.054	–0.082	–0.074	–0.077	–0.076	–0.086	–0.074	–0.072	–0.088
Result	FoW	FoW	FoW	FoW	FoW	FoW	FoW	FoW	FoW	FoW
Problem	ATA31*	ATA32*	ATA33	ATA34*	ATA35	ATA36*	ATA37*	ATA38*	ATA39*	ATA40
p-value	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Difference of ARG	–0.068	–0.073	–0.080	–0.073	–0.072	–0.083	–0.067	–0.067	–0.103	–0.099
Result	FoW	FoW	FoW	FoW	FoW	FoW	FoW	FoW	FoW	FoW
Problem	ATA41	ATA42*	ATA43	ATA44*	ATA45	ATA46*	ATA47*	ATA48*	ATA49	ATA50*
p-value	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Difference of ARG	–0.085	–0.079	–0.086	–0.100	–0.084	–0.087	–0.092	–0.088	–0.087	–0.069
Result	FoW	FoW	FoW	FoW	FoW	FoW	FoW	FoW	FoW	FoW

Note: FoW: Firefly outperforms water wave optimization algorithm.

*: At least one of the conditions of parametric tests is not satisfied; therefore, Wilcoxon signed-ranks test is used.

Table 7. p-values obtained by the normality test of Kolmogorov-Smirnov in multiple-problem analysis.

Algorithm	Harmony search	Firefly	Water wave optimization
p-value	0.200	0.200	0.200

Table 8. p-values obtained by the Levene's heteroscedasticity test in the multiple-problem analysis (based on means).

Algorithm	Harmony search vs. firefly	Harmony search vs. water wave optimization	Firefly vs. water wave optimization
p-value	0.797	0.933	0.858

Table 9. Result of the paired t-test in the multiple-problem analysis.

	Paired differences								
				95% confidence interval of the difference					
Algorithms	**Mean**	**SD**	**SEM**	**Lower**	**Upper**	***t***	**Df**	***p*-value**	**Result**
HS vs. FF	0.01749	0.01675	0.00237	0.01273	0.02225	7.384	49	0.000	FoH
HS vs. WWO	–0.05305	0.01067	0.00151	–0.05609	–0.05002	–35.174	49	0.000	HoW
FF vs WWO	–0.07054	0.01527	0.00216	–0.07488	–0.06620	–32.672	49	0.000	FoW

Note: SD: Standard Deviation, SEM: Standard Error Mean, DF: Degree of Freedom; FoH: Firefly outperforms harmony search; HoW: Harmony search outperforms water wave optimization algorithm; FoW: Firefly outperforms water wave optimization algorithm.

siderable superiority to the WWO algorithm in all problem instances;

- FF algorithm has the best performance among the proposed metaheuristic algorithms in solving the stage shop problem, especially for large-sized problem instances.

5.4.2. Multiple-problem analysis

The conclusions reached in the previous subsection can be tested by the multiple-problem analysis by which the behavior of the algorithms is analyzed considering all problem instances simultaneously [36]. In fact, the multiple-problem analysis is used to check whether there is a global difference between the algorithms over all 50 problem instances or not. To perform the multiple-problem analysis, the ARG obtained for each problem instance is used.

In order to perform the multiple-problem analysis, three required conditions for the parametric tests must be checked at first. Tables 7 and 8 represent the p-values achieved by the normality and heteroscedasticity tests in the multiple-problem analysis, respectively. As Tables 7 and 8 show, the required conditions for the parametric tests are simultaneously met in all cases. Therefore, a paired t-test is conducted to perform pairwise comparisons among the metaheuristics for all the problem instances.

The results of the conducted paired t-tests are reported in Table 9. Given that the p-value for all statistical hypothesis tests is less than the confidence level ($\alpha = 0.05$), we can claim that the pairwise performances of the algorithms are statistically different. Hence, each algorithm with the lower ARG outperforms the other one in all problem instances. According to Table 3 (ARG) and Table 9 (p-value), we can observe that the FF algorithm outperforms the competitor algorithms and the WWO algorithm has the worst performance in solving the stage shop scheduling problem, which is consistent with the conclusion of the previous subsection. It should be noted that all the statistical tests are conducted at the level of significance $\propto = 0.05$ by the SPSS 21 software package.

Moreover, the plots of 95% confidence interval for makespan mean are depicted in Figure 7 to validate the obtained results.

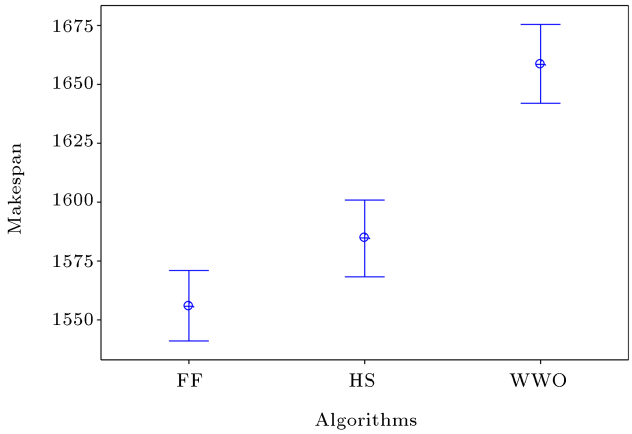

Figure 7. Means and interval plots for the makespan over all problem instances.

6. Conclusion

To evaluate the performance of approximation algorithms for large-size problems, it is not possible to find the optimal solution using the integer programming models. Therefore, a good lower bound can be a strong basis for these circumstances. In this paper, a new lower bound for the stage shop problem was proposed using the open shop problem with maximum lateness criterion. The computational results showed remarkable improvements in lower bounds for the problem instances. Three metaheuristic algorithms, namely Harmony Search (HS), firefly (FF), and Water Wave Optimization (WWO), were applied to the stage shop scheduling problem. Furthermore, the Relative Gap (RG) between the best fitness values obtained by metaheuristic algorithms and the corresponding new lower bound was taken into account to evaluate the performance of the developed metaheuristic algorithms. Then, we compared the effectiveness of the algorithms using a relatively new statistical approach, which was conducted in two ways, namely single-problem and multiple-problem analyses. Experimental results indicated that the FF algorithm outperformed the competitor algorithms in both single-problem and multiple-problem analyses. As a future direction, the proposed lower bound can be used in a branch and bound framework to find the optimal solution to the stage shop. In addition, considering other challenges (e.g., preemption, blocking, flexible environment, and sequence dependent setup times) in the stage shop scheduling problem is recommended. Furthermore, the uncertainty in processing time or machine breakdown can be regarded as research suggestions.

Supplementary data

The Supplementary data are available at:
http://scientiairanica.sharif.edu/jufile?ar_sfile=111821

Acknowledgements

The authors would like to thank the editor and reviewers for their valuable comments and suggestions, which helped to improve the paper. In addition, they would like to acknowledge the financial support of this research by University of Tehran under grant number 29922/1/02.

References

1. Nasiri, M.M., Yazdanparast, R., and Jolai, F. "A simulation optimisation approach for real-time scheduling in an open shop environment using a composite dispatching rule", *International Journal of Computer Integrated Manufacturing.*, **30**(12), pp. 1239–1252 (2017).
2. Lange, J. and Werner, F. "Approaches to modeling train scheduling problems as job-shop problems with blocking constraints", *Journal of Scheduling.*, **21**(2), pp. 191–207 (2018).
3. Shakhlevich, N.V., Sotskov, Y.N., and Werner, F. "Complexity of mixed shop scheduling problems: A survey", *Eur. J. Oper. Res.*, **120**(2), pp. 343–351 (2000).
4. Ramudhin, A. and Marier, P. "The generalized shifting bottleneck procedure", *Eur. J. Oper. Res.*, **93**(1), pp. 34–48 (1996).
5. Kis, T. "Job-shop scheduling with processing alternatives", *Eur. J. Oper. Res.*, **151**(2), pp. 307–332 (2003).
6. Özgüven, C., Özbakır, L., and Yavuz, Y. "Mathematical models for job-shop scheduling problems with routing and process plan flexibility", *Appl. Math. Modell.*, **34**(6), pp. 1539–1548 (2010).
7. Nasiri, M.M. and Kianfar, F. "A hybrid scatter search for the partial job shop scheduling problem", *The International Journal of Advanced Manufacturing Technology*, **52**(9–12), pp. 1031–1038 (2011).
8. Nasiri, M.M. and Kianfar, F. "A GA/TS algorithm for the stage shop scheduling problem", *Comput. Ind. Eng.*, **61**(1), pp. 161–170 (2011).
9. Dugarzhapov, A. and Kononov, A. "A polynomial-time algorithm for the preemptive mixed-shop problem with two unit operations per job", *Journal of Scheduling*, **19**(1), pp. 61–72 (2016).
10. Doh, H.-H., Yu, J.-M., Kim, J.-S., et al. "A priority scheduling approach for flexible job shops with multiple process plans", *International Journal of Production Research*, **51**(12), pp. 3748–3764 (2013).
11. Nasiri, M.M. "A modified ABC algorithm for the stage shop scheduling problem", *Applied Soft Computing*, **28**, pp. 81–89 (2015).
12. Rossi, A., Soldani, S., and Lanzetta, M. "Hybrid stage shop scheduling", *Expert Systems with Applications*, **42**(8), pp. 4105–4119 (2015).
13. Jin, L., Tang, Q., Zhang, C., et al. "More MILP models for integrated process planning and scheduling", *International Journal of Production Research*, **54**(14) pp. 1–16 (2016).
14. Gao, K.-Z., Suganthan, P.N., Pan, Q.-K., et al. "Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives", *Journal of Intelligent Manufacturing*, **27**(2), pp. 363–374 (2016).
15. Bożek, A. and Werner, F. "Flexible job shop scheduling with lot streaming and sublot size optimisation", *International Journal of Production Research*, **56**(19), pp. 1–21 (2017).
16. Zubaran, T.K. and Ritt, M. "An effective heuristic algorithm for the partial shop scheduling problem", *Comput. Oper. Res.*, **93**, pp. 51–65 (2018).
17. Carlier, J. "The one-machine sequencing problem", *Eur. J. Oper. Res.*, **11**(1), pp. 42–47 (1982).

18. Nowicki, E. and Smutnicki, C. "An advanced tabu search algorithm for the job shop problem", *Journal of Scheduling*, **8**(2), pp. 145–159 (2005).

19. Pinedo, M., *Scheduling: Theory, Algorithms, and Systems*, Springer (2012).

20. Cho, Y. and Sahni, S. "Preemptive scheduling of independent jobs with release and due times on open, flow and job shops", *Oper. Res.*, **29**(3), pp. 511–522 (1981).

21. Nasiri, M.M. "A pseudo particle swarm optimization for the RCPSP", *The International Journal of Advanced Manufacturing Technology*, **65**, pp. 909–918 (2013).

22. Saka, M.P., Hasançebi, O., and Geem, Z.W. "Metaheuristics in structural optimization and discussions on harmony search algorithm", *Swarm and Evolutionary Computation*, **28**, pp. 88–97 (2016).

23. Geem, Z.W., Kim, J.H., and Loganathan, G. "A new heuristic optimization algorithm: harmony search", *Simulation*, **76**(2), pp. 60–68 (2001).

24. Omran, M.G. and Mahdavi, M. "Global-best harmony search", *Applied Mathematics and Computation*, **198**(2), pp. 643–656 (2008).

25. Wang, L., Pan, Q.-K., and Tasgetiren, M.F. "A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem", *Comput. Ind. Eng.*, **61**(1), pp. 76–83 (2011).

26. Das, S., Mukhopadhyay, A., Roy, A., et al. "Exploratory power of the harmony search algorithm: analysis and improvements for global numerical optimization", *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, **41**(1), pp. 89–106 (2011).

27. Yang, X.-S., *Nature-Inspired Metaheuristic Algorithms*, Luniver Press (2010).

28. Khadwilard, A., Chansombat, S., Thepphakorn, T., et al. "Application of firefly algorithm and its parameter setting for job shop scheduling", in *First Symposium on Hands-On Research and Development* (2011).

29. Chandrasekaran, K. and Simon, S.P. "Network and reliability constrained unit commitment problem using binary real coded firefly algorithm", *International Journal of Electrical Power & Energy Systems*, **43**(1), pp. 921–932 (2012).

30. Gandomi, A., Yang, X.-S., Talatahari, S., et al. "Firefly algorithm with chaos", *Communications in Nonlinear Science and Numerical Simulation*, **18**(1), pp. 89–98 (2013).

31. Zheng, Y.-J. "Water wave optimization: a new nature-inspired metaheuristic", *Comput. Oper. Res.*, **55**, pp. 1–11 (2015).

32. Taillard, E.D. Available from: <http://mistic.heig-vd.ch/taillard> (2017).

33. Taillard, E.D. "Benchmarks for basic scheduling problems", *Eur. J. Oper. Res.*, **64**(2), pp. 278–285 (1993).

34. Mousavi, S.M., Hajipour, V., Niaki, S.T.A., et al. "Optimizing multi-item multi-period inventory control system with discounted cash flow and inflation: Two calibrated meta-heuristic algorithms", *Appl. Math. Modell.*, **37**(4), pp. 2241–2256 (2013).

35. Peace, G.S., *Taguchi Methods: A Hands-on Approach*, Addison Wesley Publishing Company (1993).

36. García, S., Molina, D., Lozano, M., et al. "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization", *Journal of Heuristics*, **15**(6), pp. 617–644 (2009).

37. Zar, J., Biostatistical analysis, 4th Edition, Prentice Hall, Englewood Cliffs (1999).

Appendix A

The previous lower bound

Here, we give the simple lower bound for the makespan of the stage shop problem, which was presented by Nasiri and Kianfar [8]. First, let us define some parameters.

a, b	Indices for operations
J_j	Set of operations of job j
I_i	Set of operations that should be processed on machine i
H_{jk}	Set of operations of stage k of job j
P_a	Processing time of operation a

Now, it is needed to define heads (r) and tails (q). In the stage shop problem, r_a parameters for all operations of a stage ($a \in H_{jk}$) are equal and the same is true for q_a. The parameters can be defined as follows:

$$r_a = \sum_{t=1}^{k-1} \sum_{b \in H_{jt}} p_b, \tag{A.1}$$

$$q_a = \sum_{t=k+1}^{s_j} \sum_{b \in H_{jt}} p_b. \tag{A.2}$$

These definitions are consistent with the conventional definitions of heads and tails in job shop and open shop.

$$LB_1 = \max\left\{ \max_{i \in \{1,\ldots,m\}} \left\{ \min_{a \in I_i} r_a + \sum_{a \in I_i} p_a + \min_{a \in I_i} q_a \right\}, \right.$$
$$\left. \max_{j \in \{1,\ldots,n\}} \left\{ \sum_{a \in J_j} p_a \right\} \right\}. \tag{A.3}$$

The operations of job $j(J_j)$ cannot be performed concurrently. Hence, the makespan would be greater

than or equal to the sum of the processing times of the operations in J_j even though there is no idle time between them. On the other hand, the operations that should be processed on machine $i(I_i)$ cannot be performed in parallel, either. In addition, the first of these operations cannot be started before $\min_{a\in I_i} r_a$. Then, at least the sum of the processing times $(\sum_{a\in I_i} p_a)$ should be passed before the last operation of I_i is completed. Finally, makespan would be at least by $\min_{a\in I_i} q_a$ greater than the completion of the last operation.

Appendix B

Examples

Example 3.1. In the stage shop problem of Figure B.1, we want to form an open shop.

Suppose that we want to derive $OpenShop_1$ from the mentioned stage shop problem. As can be seen in Figure B.1, machine 1 is used at stage 1 of every three jobs and therefore, $K_1 = \{(1,1),(2,1),(3,1)\}$. Other machines required in forming are machines 3 and 5 that are related to operations of the stage 1 of each job. In $OpenShop_1$, the machines 1, 3, and 5 are renumbered to 1, 2, and 3, respectively. Indeed, $OpenShop_1$ includes three jobs, three machines, and five operations. The data of $OpenShop_1$ is shown in Table B.1. For better illustration, we also describe the formation of $OpenShop_2$, which is different from $OpenShop_1$ in some way, as the operations of $OpenShop_2$ are from stages with different indices. Three operations in l_2 are from stage 2 of job 1, stage 2 of job 2, and stage 3 of job 3. Therefore, $K_2 = \{(1,2),(2,2),(3,3)\}$.

		Operations						
	Jobs	1	2	3	4	5	6	$\sum_{a\in J_j} p_a$
Processing Time	1	2	1	3	3	3	6	18
	2	5	4	2	2	3	4	20
	3	6	2	4	2	4	2	20
Machine	1	3	1	4	2	6	5	
	2	1	2	4	6	5	3	
	3	5	1	3	6	2	4	

Figure B.1. An instance of the stage shop problem.

Table B.2. Calculations for obtaining the lower bound.

	Machines					
	1	**2**	**3**	**4**	**5**	**6**
$\min_{a\in I_i} r_a$	0	3	0	3	0	3
$\sum_{a\in I_i} p_a$	8	11	10	7	15	7
$\min_{a\in I_i} q_a$	12	2	0	0	0	0
Total	20	16	10	10	15	10

Example 3.2. Consider the stage shop problem in Figure B.1, including 3 jobs and 6 machines.

The stages of each job are identified with different colors, e.g., job 3 comprises four stages and job 2 includes three stages. The previous lower bound can be calculated using Eq. (A.3). Therefore, we have $LB_1 = 20$ (the calculations are shown in Figure B.1 and Table B.2).

Solving $OpenShop_1$ with complete enumeration of feasible solutions results in $L_{\max} = 1$, which is equivalent to the lower bound $LB_2 = 20 + 1 = 21$. The optimal solution is represented in Figure B.2.

Example 3.3. Consider the problem $OpenShop_1$ (in Example 3.1) with preemption, namely a problem with three jobs and three machines $(O_3\,|r_j, prmp|\,L_{max})$.

In this problem, there are two intervals that can be specified with $a_1 = r_1 = r_2 = r_3 = 0$, $a_2 = d_1 = d_2 = 5$, and $a_3 = d_3 = 8$. The lengths of the intervals are $\delta_1 = 5$ and $\delta_2 = 3$, respectively. Therefore, there would be 18 $(3 \times 3 \times 2)$ decision variables x_{ijk}. Now,

Table B.1. Open shop problem for machine 1 ($OpenShop_1$).

		Machines					
	Jobs	**1**	**2**	**3**	r_j	q_j	$d_j = LB_1 - q_j$
Processing time	1	1	2	—	0	15	5
	2	5	—	—	0	15	5
	3	2	—	6	0	12	8

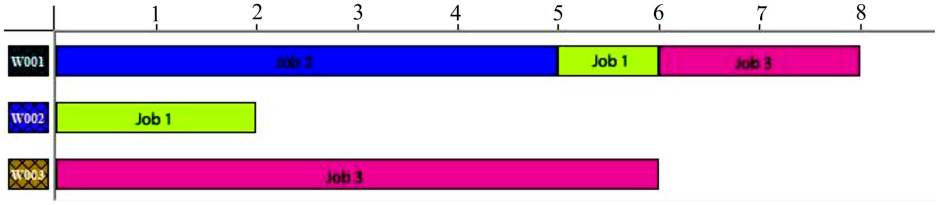

Figure B.2. The solution to the open shop problem for machine 1.

the variables with the value of zero should be specified for each i: $X_{i11} = 0$ because $r_1 \geq a_2$; $x_{i24} = 0$ because $d_2 \leq a_4$; $x_{i32} = 0$ because $r_3 \geq a_3$; and $x_{i31} = 0$ because $r_3 \geq a_2$.

In addition, machine 3 is not required by job 1 and job 2. Also, machine 2 is not required by job 2 and job 3. Thus, for $k = 1, 2$: $x_{31k} = 0$, $x_{22k} = 0$, $x_{32k} = 0$, and $x_{23k} = 0$.

The first set of constraints of the LP includes six constraints. The first constraint of this set (for $j = 1$, $k = 1$) is:

$$x_{111} + x_{211} + x_{311} \leq 5.$$

The second set of constraints of the LP has six constraints. The first constraint of this set (for $i = 1$, $k = 1$) is:

$$x_{111} + x_{121} + x_{131} \leq 5.$$

The third set of constraints of the LP has nine constraints. The first constraint of this set (for $i = 1$, $j = 1$) is:

$$x_{111} + x_{112} = 1.$$

The linear programming model obtained in this phase has no feasible solution. Therefore, each of due dates d_j is replaced by $d_j + 1$. Consequently, we have $a_1 = 0$, $a_2 = 6$, and $a_3 = 9$. The lengths of the new intervals are equal to $\delta_1 = 6$ and $\delta_2 = 3$, respectively. In addition, the variables that should take the value of zero do not change. Solving the new problem leads to a feasible solution. As a result, the minimum $L_{\max}$ in the preemptive (like the non-preemptive) problem is equal to one and the new lower bound is equal to $LB_2 = 20 + 1 = 21$.

Biographies

Mohammad Mahdi Nasiri is an Associate Professor at University of Tehran, Tehran, Iran. He received his BSc, MSc, and PhD degree in Industrial Engineering from Sharif University of Technology, Tehran, Iran, in 2004, 2006, and 2011, respectively. His research interests are in scheduling, cross-dock optimization, supply chain management, transportation, and integer programming. His papers have appeared in International Journal of Production Research, Computers and Industrial Engineering, Applied Soft Computing, Transportation, and other reputed journals.

Mahdi Hamid is a PhD candidate in the field of Industrial Engineering in the School of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran. He earned BSc degree from K. N. Toosi University of Technology and MSc degree from University of Tehran both in Industrial Engineering. He has published several journal and conference papers in various fields. His research interests include safety, optimization, scheduling, simulation, healthcare systems, and data analysis.