Journal of Intelligent Manufacturing (2019) 30:2733–2742
https://doi.org/10.1007/s10845-018-1425-8

An effective metaheuristic algorithm for flowshop scheduling with deteriorating jobs

Hongfeng Wang[1] · **Min Huang**[1] · **Junwei Wang**[2]

Received: 23 June 2017 / Accepted: 28 May 2018 / Published online: 6 June 2018

Abstract
The deterioration effect in flowshop scheduling has gained a growing concern from the community of operational research in recent years. However, all of existing studies focus on two- or three-machine flow shops. In this paper, a m-machine ($m > 3$) flowshop scheduling problem (FSSP) with deteriorating jobs is investigated and a novel metaheuristic algorithm called multi-verse optimizer (MVO) is employed to solve it. The MVO algorithm can accomplish the optimization process via exchanging objects of universes through white/black hole and wormhole tunnels. In the novel MVO algorithm, a new elitist selection scheme is designed to construct the effective white/black hole tunnels, whereas two different local search operators are hybridized and embedded to further enhance the exploitation capability. Experimental results indicate that the proposed algorithm can achieve the satisfactory performance in solving the investigated FSSP with deteriorating jobs.

Keywords Scheduling · Flowshop scheduling · Deterioration · Metaheuristic algorithm · Multi-verse optimizer

Introduction

Over the past decades, scheduling problems in manufacturing systems have gained much attention from academic and industrial fields (Zhong et al. 2015; Wang et al. 2016a, b; Costa et al. 2017; Fu et al. 2017; Wang et al. 2017a, b; He et al. 2018). As one of the most well-known scheduling problems, flowshop scheduling problem (FSSP) is defined as finding a processing sequence of n jobs that are to be processed on m machines with the same machine route so that a given scheduling criterion is optimized (Ventura and Yoon 2013; Ta et al. 2018). The processing time of a job on a machine is assumed to be known and fixed in the classical scheduling. However, this assumption might not be true in many real-world manufacturing systems (Browne and Yechiali 1990; Alidaee and Womer 1999; Cheng et al. 2004; Wang et al. 2016a, b). For example, processing a job after long-time waiting may require an extra reheat time in the ingot industrial process. Similarly, the process efficiency of jobs may decrease over time due to the fatigue effect in the clothing industrial process. In manufacturing systems, it means that the deterioration occurs.

The deterioration effect is that a job's processing time becomes longer if it is delayed to be processed. Gupta and Gupta (1988) initially examined the deterioration effect in a single machine scheduling problem, where the actual processing time of a job was a linear function of the start time. Ng et al. (2002) indicated that a single machine scheduling problem with objective of minimizing the total weighted completion time was NP-hard if the actual processing time of a job was a linear function. In most of the research on deterioration effect (Browne and Yechiali 1990; Mosheiov 1994; Bachman and Janiak 2000; Gawiejnowicz et al. 2011), the actual processing time of a job was defined as a linear function of the start time, that is, $p_i = a_i + b_i \cdot s_i$, where p_i and s_i denoted the actual processing time and start time of job i, respectively. Ji and Zun (2005) considered the deteriorating jobs with $p_i = b_i \cdot s_i$. Cheng and Ding (2001) studied the single machine scheduling with $b_i = b$ or $a_i = a$. If a critical time of deteriorating d_i was used, that is, deterioration would take effect only on the jobs that were processed after d_i. Kunnathu (1990) defined the actual processing time of a job as a piecewise function, i.e., $p_i = \max\{a_i, a_i + b_i \cdot (s_i - d_i)\}$. In the work of Sundararaghavan and Kunnathur (1994) and

✉ Hongfeng Wang
hfwang@mail.neu.edu.cn

[1] College of Information Science and Engineering, Northeastern University, Shenyang, China

[2] Department of Industrial and Manufacturing System Engineering, The University of Hong Kong, Hong Kong, China

Jeng and Lin (2005), the actual processing time was defined as $p_i = \begin{cases} a_i, s_i \le d_i \\ a_i + b_i, otherwise \end{cases}$. Cheng et al. (2003) used a combination mechanism and the actual processing time was defined as $p_i = a_i - b_i \cdot \min\{s_i, d_i\}$.

Most of the earlier studies on scheduling with deterioration consideration focused on the single machine problems. In recent years, there is a growing interest on deterioration effect in FSSPs (Wu and Lee 2006; Shiau et al. 2007; Fu et al. 2018a). Lee et al. (2010) studied a two-machine FSSP with deteriorating jobs and blocking, where the objective function is to minimize the makespan. Wang and Wang (2013) investigated the makespan problem with a linear deterioration on a three-machine flow shop and proposed a branch and bound algorithm and a heuristic algorithm for solving it. Cheng et al. (2014) considered a two-machine FSSP with time-dependent deteriorating jobs. Lee et al. (2014) designed a branch and bound algorithm for a FSSP with deterioration effect in order to minimize the maximal tardiness. It is noted that all of existing studies focus on two- or three-machine flow shop, whereas the m-machine ($m > 3$) flowshop scheduling has been widely applied in real-world manufacturing systems.

Therefore, this paper studies the FSSP with deteriorating jobs in a general m-machine ($m > 3$) flowshop manufacturing system. Since an FSSP with deteriorating jobs is NP-hard even for two machines, multi-verse optimizer (MVO), a novel metaheuristic algorithm inspired from cosmology, is applied to solve the investigated problem. Two key features are introduced to the proposed MVO algorithm. The first feature is a new elitist selection scheme that constructs the effective white/black hole tunnels. The other feature is a local search method that hybridizes two different local search operators to enhance the exploitation capacity.

The reminder of this paper is organized as follows. Section 2 gives a definition of the investigated problem with the relevant assumptions and notations, and presents a mixed integer programming model. Section 3 introduces the framework of the proposed MVO algorithm in detail. Section 4 carries out a series of simulation experiments and analyzes the results. The final section concludes this paper with some discussions on the future work.

Problem formulation

The investigated flowshop scheduling problem (FSSP) with deterioration jobs can be described as follows. There are n jobs that require to be processed on m machines following the same machine route. For each job that is processed on each machine, the normal process time is fixed, while the actual process time varies according to the start time. The following assumptions are invoked in modelling the investigated problem.

1. Each machine can process only one job at any time, and each job can be processed on only one machine at any time;
2. No preemption is allowed, i.e., processing of job on one machine cannot be interrupted;
3. All jobs are ready for processing at the beginning, and have no precedence constraints;
4. All machines are persistently available.

For convenience of description, the notations are listed as follows.

Indices

I	machine index set, $I = \{1, 2, \ldots, m\}$, where m is the number of machines;
J	job index set, $J = \{1, 2, \ldots, n\}$, where n is the number of jobs.

Symbol variables

s_{ij}	the start time of job j on machine i;
p_{ij}	the normal process time of job j on machine i;
a_{ij}	the deterioration rate of job j on machine i;
p'_{ij}	the actual process time of job j on machine i, where $p'_{ij} = p_{ij} + a_{ij} \cdot s_{ij}$;
c_{ij}	the completion time of job j on machine i, where $c_{ij} = s_{ij} + p'_{ij}$;
c_j	the completion time of job j, where $c_j = c_{mj}$;
d_j	the due time of job j;
G	an infinite number.

Decision variables

x_{kj}	0–1 integer decision variable. If job j is followed by job k immediately, $x_{kj} = 1$; otherwise, $x_{kj} = 0$.

Based on the above-mentioned assumptions and notations, a mixed integer programming model can be formulated as follows.

$$\min \sum_{j=1}^{n} \max\{c_j - d_j, 0\} \tag{1}$$

s.t.

$$s_{ij} + p'_{ij} \le s_{i+1,j}, \quad i = 1, \ldots, m-1, \quad j = 1, \ldots, n \tag{2}$$

$$s_{ik} + p'_{ik} \le s_{ij} + G \times \left(1 - x_{kj}\right), \quad i = 1, \ldots, m, \quad j, k = 1, \ldots, n, j \ne k \tag{3}$$

$$x_{kj} + x_{jk} \le 1, \quad i = 1, \ldots, m, \quad j, k = 1, \ldots, n, \quad j \ne k \tag{4}$$

$$c_{ij} \geq 0, s_{ij} \geq 0, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n \tag{5}$$

$$x_{kj} \in \{0, 1\}, \quad k, j = 1, \ldots, n \tag{6}$$

where formula (1) denotes the objective function is to minimize the total tardy time. Constraint (2) and constraint (3) ensure that each job can be processed on only one machine at any time and each machine can process only one job at any time, respectively. Constraint (4) states the order relation of different jobs. Constraint (5) represents the range of variables and the final constraint denotes that the decision variables can only take values of 0 or 1.

Given that the actual process time of a job on a machine is defined as a linear function of the normal process time and the start time, the investigated problem can be denoted as $F\left|prmu, p'_{ij} = p_{ij} + a_{ij} \times s_{ij}\right| \sum_{j=1}^{n} \max\{c_j - d_j, 0\}$ based on the three-field notation schemas.

Solutions method

It is obvious that the investigated problem in this paper can be translated into a traditional flowshop scheduling problem (FSSP) if the deterioration rate of any job on any machine is zero. Since the traditional FSSP is NP-hard, the investigated FSSP with deteriorating jobs is also NP-hard as its extension. Since the traditional exact methods and approximation or heuristic approaches cannot solve these large-scale NP-hard problems very well, a metaheuristic algorithm is employed to solve the model.

Algorithm framework

Over past decades, metaheuristic algorithms have been applied widely for a lot of complex combination optimization problems (Gao et al. 2016; Wang et al. 2017a, b; Dao et al. 2018; Fu et al. 2018b). Based on the mechanism of population -based stochastic optimization, metaheuristic algorithm usually starts its optimization process by generating a population of random candidate solutions called individuals, and then recombines these initial individuals over a predefined number of iterations. The main distinction among different metaheuristic algorithms lies in the mechanism of recombining individuals, which is often drawn the inspiration from nature, biology or physics.

Recently, a new metaheuristic algorithm called multi-verse optimizer (MVO) was proposed based on the theory of multi-verse in physics (Mirjalili et al. 2016). In MVO, each candidate solution in the search space is assumed to a universe and each variable in the solution is analogous to an object in that universe. Each universe is assigned to an inflation rate that is proportional to the corresponding function value of solution. MVO can accomplish the optimization process through exchanging objects among different universes inspired from three concepts, that is, white holes, black holes and wormholes, in cosmology. More specially, different universes are able to exchange the objects through white/black hole or wormhole tunnels. The basic framework of MVO can be shown by the pseudo-code in Fig. 1.

From Fig. 1, it can be seen that the search process of MVO can be divided into two stages, i.e., exploration and exploitation. In the exploration stage, the source universe can transfer the objects to the destination universe through white/black hole tunnels. The universe with the higher inflation rate or fitness would own a white hole with a higher probability, whereas the universe with the lower inflation rate is assumed to have a black hole with a higher probability. In the basic MVO algorithm, a roulette wheel selection scheme is used to maintain the exploration capacity. In the exploitation stage, only the objects in the best obtained universe can be allowed to move to any universe if there exists a wormhole between them, which is determined by a coefficient, i.e., wormhole existence probability. Another coefficient, which is termed as travelling distance rate, is used to control the exploitation degree around the best universe found so far.

In the following sections, some detailed algorithmic designs are given in order to further improve the performance of the MVO algorithm in term of exploration and exploitation when solving the FSSP with deteriorating jobs.

Solution representation

It is a basic work to express a candidate solution using a proper representation scheme is a basic work in the design of a metaheuristic algorithm. A good representation can help the algorithm obtain high quality solutions easily, while an improper scheme may make it hard for an algorithm to find even feasible solutions.

In the FSSP, a candidate solution can be regarded as a processing sequence or permutation of n jobs. However, a universe in the original MVO algorithm is represented as an n-dimensional real number vector, which cannot be directly adopted in solving the investigated problem. To overcome this obstacle, the following solution representation approach is applied.

For a given universe in the MVO, the largest value of objects is picked out and assigned a smallest value 1 of processing sequence, and then the second largest value of objects is assigned a sequence value 2. With the same way, all the object values will be handled to convert a universe to a job permutation. A simple example is illustrated in Table 1, where $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$ and $\boldsymbol{\pi} = \{\pi_1, \pi_2, \ldots, \pi_n\}$ denote a universe in the MVO and a job permutation in the FSSP, respectively.

Algorithm of Basic MVO

initialize a set of universes as P;
while (a given termination condition is not met) **do**
  evaluate the fitness of universes in P;
  sort all universes based on their fitness as Π;
  normalize the inflation rates of universes as K;
  for ($i := 1$ **to** pop_size) **do**
    $black\_hole\_index := i$;
    for ($j := 1$ **to** n) **do**
// Exchanging objects through white/black hole tunnels
      $r1 := random()$;
      if ($r1 < K(i)$) **then**
        $white\_hole\_index := roulette\_wheel\_selection(K)$;
        $P(black\_hole\_index, j) := \Pi(white\_hole\_index, j)$;
      end if
// Exchanging objects through wormhole tunnels
      $r2 := random()$;
      if ($r2 < wormhole\_existance\_probability$) **then**
        $r3 := random()$; $r4 := random()$;
        if ($r3 < 0.5$) **then**
          $P(black\_hole\_index, j) := best\_universe(j) + travelling\_distance\_rate *$
          $((ub(j) - lb(j)) * r4 + lb(j))$;
        else
          $P(black\_hole\_index, j) := best\_universe(j) - travelling\_distance\_rate *$
          $((ub(j) - lb(j)) * r4 + lb(j))$;
        end if
      end if
    end for
  end for
end while
Denotation:
  pop_size: size of population, i.e., number of universes;
  n: number of variables in a candidate solution;
  $best\_universe$: the best universe found so far;
  $ub(j)$: upper bound of the j-th variable;
  $lb(j)$: lower bound of the j-th variable;
  $random()$: a random number in [0,1].

Fig. 1 Pseudo-code for the basic MVO algorithm

Table 1 An example for representation approach

Dimension	1	2	3	4	5	6	7	8
$\boldsymbol{x}$	0.14	0.41	0.21	0.54	1.02	1.22	0.92	0.75
$\boldsymbol{\pi}$	8	6	7	5	2	1	3	4

Elitist selection

As described in the preceding section, the MVO can start its search process by exchanging the objects of universes through white/black hole tunnels. When a universe becomes the destination universe with a black hole, its corresponding source universes that have white holes can be chosen by the roulette wheel scheme in the basic MVO algorithm. The purpose of this selection is to guarantee the quality of the source universes and then to further maintain the exploration capacity of the algorithm.

Here, a new selection scheme is proposed based on the mechanism of elitism that is often used in the genetic algorithm. More specially, some better universes would be marked as the elite universes, while the left would be marked as the common universes. When a common universe has a black hole, one universe with a white hole can be chosen from the elite universes randomly or all the universes by the roulette wheel scheme. The pseudo-code of exchanging the objects of universes based on this selection scheme can be described in Fig. 2.

Local search

After exchanging objects of universes via white/black holes, each newly generated universe will undergo a random teleportation in its objects through wormholes towards the best universe found so far. Obviously, the purpose of this operation is to execute local refinement for a universe so as to improve its inflation rate or fitness using wormholes.

Here, two local search operators are hybridized and embedded into the MVO algorithm in order to enhance the exploitation capacity.

Fig. 2 Pseudo-code of exchanging objects of universes through white/black hole tunnels

Procedure of exchanging the objects of universes through white/black hole tunnels

sort all universes based on their fitness as Π;
normalize the inflation rates of universes as K;
for ($i := 1$ **to** pop_size) **do**
  $black\_hole\_index := i$;
  if ($black\_hole\_index < elite\_size$) **then**
    for ($j := 1$ **to** n) **do**
      $r1 := random()$;
      if ($r1 < K(i)$) **then**
        $white\_hole\_index := roulette\_wheel\_selection(K)$;
        $P(black\_hole\_index, j) := \Pi(white\_hole\_index, j)$;
      end if
    end for
  else
    for ($j := 1$ **to** n) **do**
      $r1 := random()$;
      if ($r1 < 0.5$) **then**
        $white\_hole\_index := random(1, elite\_size)$;
        $P(black\_hole\_index, j) := \Pi(white\_hole\_index, j)$;
      else
        $white\_hole\_index := roulette\_wheel\_selection(K)$;
        $P(black\_hole\_index, j) := \Pi(white\_hole\_index, j)$;
      end if
    end for
  end if
end for
Denotation:
  $elite\_size$: number of elite universes;
  $random(1, m)$: a random integer value in $[1, m]$.

Fig. 3 Pseudo-code for CBLS operator

Procedure of executing CBLS operation upon a real number vector $\boldsymbol{x}$

$\Phi := \emptyset$;
for ($i := 1$ **to** ls_size) **do**
  $\boldsymbol{x}' := \boldsymbol{x}$;
  $ls\_index := random(1, n)$;
  $\boldsymbol{x}'_{ls\_index} := \left(ub(ls\_index) - lb(ls\_index)\right) * random() + lb(ls\_index)$;
  $\Phi := \Phi \cup \{\boldsymbol{x}'\}$;
end for
evaluate all universes in Φ;
choose the best universe $\boldsymbol{x}^*$ in Φ;
if ($\boldsymbol{x}^*$ is better than $\boldsymbol{x}$) **then**
  $\boldsymbol{x} := \boldsymbol{x}^*$;
end if
Denotation:
  ls_size: execution number of local search.

1. Coding based local search: the first local search method can be termed as coding based local search (CBLS). In the MVO algorithm, each universe can be coded by a real number vector. Therefore, CBLS can be designed as executing local changes upon the real number vector in a universe, which can be shown in Fig. 3.
2. Solution based local search: the second local search method can be called as solution based local search (SBLS), which is designed as swapping two different jobs in a job permutation. Figure 4 can illustrate how to execute SBLS operation upon a job permutation π.

Obviously, it is very important to select two jobs from $\boldsymbol{\pi}$ in this SBLS operator. There are n jobs to be processed on m machines in the investigated FSSP with deteriorating jobs. Let $[k]$ and $[l]$ denote the jobs in the kth and lth processing sequence on a machine, respectively. Without loss of generality, set $k > l$ here. Let $s_{i[k]}$, $p_{i[k]}$ and $a_{i[k]}$ denote the start

Procedure of executing SBLS operation upon a job permutation $\boldsymbol{\pi}$

$\Phi := \emptyset$;
for ($i := 1$ **to** ls_size) **do**
  $\boldsymbol{\pi}' := swap(\boldsymbol{\pi}, k, l)$;
  $\Phi := \Phi \cup \{\boldsymbol{\pi}'\}$;
end for
evaluate all solutions in Φ;
choose the best universe $\boldsymbol{\pi}^*$ in Φ;
if ($\boldsymbol{\pi}^*$ is better than $\boldsymbol{\pi}$) **then**
  $\boldsymbol{\pi} := \boldsymbol{\pi}^*$;
end if

Fig. 4 Pseudo-code for SBLS operator

time, the normal process time and the deterioration rate of the job in the kth process sequence on machine i. Therefore, four different heuristic rules will be considered when designing SBLS.

Heuristic rule I Here, we only consider the scheduling of n jobs on the first machine. Two jobs, that is, k and l, are selected randomly to execute the swap operation. The corresponding jobs are termed as $[k]$ and $[l]$. Let P_k denote the initial actual process time of the job in the kth process sequence and P'_k denote the actual process time of the job in the kth process sequence after being swapped.

If job $[k]$ is initially processed in the kth process sequence on machine 1, we have $P_k = p_{1[k]} + a_{1[k]} \times s_{1[k]}$. If job $[l]$ is swapped with job $[k]$, we have $P'_k = p_{1[l]} + a_{1[l]} \times s_{1[k]}$. It is obvious that $P_k > P'_k$ means job l has a less completion time than job k since they have the same start time on machine 1. Once job l is swapped with job k in the process sequence, the completion times of all the jobs between job k and job l would decrease due to the ahead of their start times.

Therefore, a heuristic rule can be adopted that the swap operation will be executed only if $P_k > P'_k$ when two jobs k and l are selected randomly from a job permutation.

Heuristic rule II This rule is very similar to the previous one except that the last machine in flow shop will be considered. Here, we have $P_k = p_{m[k]} + a_{m[k]} \times s_{m[k]}$ and $P'_k = p_{m[l]} + a_{m[l]} \times s_{m[k]}$. If $P_k > P'_k$, the operation of swapping two jobs k and l can be allowed.

Heuristic rule III Let T_{ij} denote the equivalent actual process time of job j on machine i and T_i denote the total load of machine i. Here, T_i is defined as the sum of the equivalent actual process times of all jobs in machine i, which can be calculated by the following formula.

$$T_i = \sum_{j=1}^{n} T_{ij} = \sum_{j=1}^{n} \left(p_{ij} + a_{ij} \times s_{ij}\right) \tag{7}$$

Therefore, we can firstly achieve the machine with the maximal load value and then the same swap operation with the preceding two heuristic rules will be applied.

Heuristic rule IV Let $\bar{P}_k$ and $\bar{P}'_k$ denote the average actual process time of the job in the kth process sequence on all machines before and after swapping job k and job l, respectively. Here, $\bar{P}_k$ and $\bar{P}'_k$ can be calculated by the following two formulas.

$$\bar{P}_k = \frac{1}{m} \sum_{i=1}^{m} \left(p_{i[k]} + a_{i[k]} \times s_{i[k]}\right) \tag{8}$$

$$\bar{P}'_k = \frac{1}{m} \sum_{i=1}^{m} \left(p_{i[l]} + a_{i[l]} \times s_{i[k]}\right) \tag{9}$$

Similarly, the swap operation will be executed if the value of $\bar{P}_k$ is larger than $\bar{P}'_k$, otherwise the jobs k and l are not swapped.

Based on the above mentioned discussion, the proposed solution method for the investigated FSSP with deteriorating jobs can be expressed by the pseudo-code in Fig. 5.

Simulation experiments

Test instances

In this section, a set of random test instances are generated using the method of Chu (1992), which can be described as follows.

Here, the test instance that there are n jobs are to be processed in an m-machine flow shop can be termed as $m \times n$. The normal process time of a job on a machine is uniformly distributed in the interval $[1, 100]$. The due time of a job is generated using a discrete uniform distribution in the range $[T \times (1 - \tau - R/2), T \times (1 - \tau + R/2)]$, where T denotes the sum of the normal process time of all jobs,τ and R denote two predefined parameters ($\tau = 0.5$ and $R = 0.5$ in the experiments). The deterioration rate of a job on a machine is generated randomly in the interval $(0.1, 0.3)$.

Experimental settings

The experiments are carried out in order to examine the performance of our proposed algorithm on the test instances. All the algorithms are coded in C and run on Thinkpad T420 with a 2.3 GHz. The following parameters are used in the proposed MVO algorithm. The values of pop_size, $elite\_size$ and ls_size are set to 15, 5 and 3, respectively. The value of WEB and TDR are both set to 0.5, and the upper bound and lower bound of each variable, i.e., ub and lb, are set to 4.0 and 0.0, respectively.

To make a fair comparison, the maximal number of allowable fitness evaluations is set to $250 * n$. For each experiment of an algorithm on a test instance, 30 inde-

Algorithm of the proposed MVO

```
initialize a set of universes as P;
while (a given termination condition is not met) do
    evaluate the fitness of universes in P;
    sort all universes based on their fitness as Π;
// execute local search upon all elite universes
    for (i:= 1 to elite_size) do
        r0:= random();
        if (r0 < 0.5) then
            execute CBLS upon x(i) in Π;
        else
            execute SBLS upon π(i) in Π;
        end if
    end for
    normalize the inflation rates of universes as K;
    for (i:= 1 to pop_size) do
        black_hole_index:= i;
        for (j:= 1 to n) do
            if (black_hole_index < elite_size) then
                r1:= random();
                if (r1 < K(i)) then
                    white_hole_index:= roulette_wheel_selection(K);
                    P(black_hole_index, j):= Π(white_hole_index, j);
                end if
            else
                if (r1 < 0.5) then
                    white_hole_index:= random(1, elite_size);
                    P(black_hole_index, j):= Π(white_hole_index, j);
                else
                    white_hole_index:= roulette_wheel_selection(K);
                    P(black_hole_index, j):= Π(white_hole_index, j);
                end if
            end if
            r2:= random();
            if (r2 < wormhole_existance_probability) then
                r3:= random(); r4:= random();
                if (r3 < 0.5) then
                    P(black_hole_index, j):= best_universe(j) + travelling_distance_rate *
                    ((ub(j) − lb(j)) * r4 + lb(j));
                else
                    P(black_hole_index, j):= best_universe(j) − travelling_distance_rate *
                    ((ub(j) − lb(j)) * r4 + lb(j));
                end if
            end if
        end for
    end for
```

Fig. 5 Pseudo-code for the proposed MVO algorithm

pendent runs are executed with the same initial seeds. For each run, the percentage relative error (PRE) is calculated as follows:

$$PRE\,(A) = \frac{1}{R} \times \sum_{i=1}^{R} \left(\frac{C_i^A - C_i^B}{C_i^B} \right) \tag{10}$$

where C_i^B is the best fitness value found by any of peer algorithms and C_i^A is the best fitness value obtained by algorithm A in ith run. Obviously, the smaller the value of $PRE\,(A)$ is, the better performance algorithm A has.

Experimental results and analysis

In the first experiment, we investigate the performance of MVO algorithms with different selection schemes in order to examine the effect of the elitist selection scheme proposed in Sect. 3.3. In detailed, two variants of MVO are compared, that is, one employs the proposed elitist selection scheme, while the other uses the traditional roulette wheel selection scheme. The number of jobs (n) are set from 20, 40, 60, 80 to 100 and the numbers of machines (m) are set to 5 and 10, respectively. Thus, 10 different test instances would be generated. The experimental results can be shown in Table 2.

Table 2 Experimental results of MVO algorithms with different selection schemes

Instance	Elitist	Roulette wheel	Instance	Elitist	Roulette wheel
5 × 20	0.0256	0.0370	10 × 20	0.0763	0.1717
5 × 40	0.0312	0.0433	10 × 40	0.0982	0.2032
5 × 60	0.0454	0.0535	10 × 60	0.1110	0.2484
5 × 80	0.0570	0.0702	10 × 80	0.1661	0.2819
5 × 100	0.0613	0.0868	10 × 100	0.2053	0.3594

From Table 2, it can be seen that the elitist selection scheme proposed in Sect. 3.2 make a better effect upon the performance of MVO than the traditional roulette wheel selection scheme. The MVO algorithm with elitist selection can always achieve the better experimental results in all test instances, which validate our expectation of the proposed selection scheme upon the performance of MVO for the investigated problem.

In the second experiment, we examine the effect of local search methods proposed in Sect. 3.3 upon the performance of MVO algorithms. In detail, six different variants of MVO, where non local search, only CBLS, CBLS and SBLS with four different heuristic rules are embedded respectively, are evaluated and compared on the same test instances as in the preceding experiment. From the experimental results in Table 3, several conclusions can be observed as follows.

Firstly, the performance of five MVO algorithms with local search methods is always better than that of MVO algorithm without local search method on all test instances. These results indicate that the local search methods proposed in this paper can be helpful to improve the performance of MVO algorithms for the investigated FSSP with deteriorating jobs.

Secondly, four MVO algorithms that employ two local search methods outperform the MVO algorithm that only uses one local search method on all test instances. In fact, CBLS can be regarded as a random and problem-independent local search as a result of executing upon the real number vector in a universe, while SBLS can be regarded as a problem-dependent local search as a result of executing upon the jobs permutation in a solution. These results show that it is beneficial to hybridize two different local search methods together.

Thirdly, heuristic rules III and IV perform better than heuristic I and II in improving the performance of MVO algorithms on the most test instances. This happens because scheduling of jobs on only one (the first or the final) machine would be involved in heuristic rule I and heuristic rule II, whereas the corresponding heuristic information on jobs on all machines can be considered in heuristic rule III and heuristic IV.

In the final experiment, we compare our proposed MVO algorithm with two state-of-the-art algorithms, i.e., opposition-based differential evolution (ODDE) (Li and Yin 2013) and hybrid modified global-best harmony search (hmGHS) (Wang et al. 2011) in solving the investigated FSSP with deteriorating jobs. These two peer algorithms are also proposed for FSSP with the similar solution presentation scheme. For ODDE and hmGHS, their parameters are fixed the same as those in the original paper. The same test instances are also used as those in the preceding experiments. The experimental results can be shown in Table 4, where the statistical results in parentheses denotes comparing the proposed MVO algorithm with the corresponding peer algorithm by the one-tailed t-test with 58 degrees of freedom at a 0.05 level of significance. The t-test result is shown as '$s+$', '$s-$' or '$\sim$' when our proposed MVO algorithm is significantly better than, significantly worse than, or statistically equivalent to its peer algorithms, respectively.

From this table, it can be seen that our proposed MVO algorithm always outperforms two peer algorithms, which is further verified by the results of t-test of MVO versus ODDE or hmGHS. As we can see, MVO performs significantly better than ODDE on 10 test instances and hmGHS

Table 3 Experimental results of MVO algorithms with different local search methods

Instance	nonLS	CBLS	CBLS+ SBLS-I	CBLS+ SBLS-II	CBLS+ SBLS-III	CBLS+ SBLS-IV
5 × 20	0.0913	0.0433	0.0139	0.0150	0.0148	0.0165
5 × 40	0.1348	0.0625	0.0232	0.0204	0.0160	0.0182
5 × 60	0.1969	0.0729	0.0459	0.0277	0.0203	0.0195
5 × 80	0.2808	0.0890	0.0660	0.0304	0.0264	0.0284
5 × 100	0.3604	0.1309	0.0730	0.0450	0.0537	0.0664
10 × 20	0.1472	0.0741	0.0293	0.0185	0.0145	0.0147
10 × 40	0.1773	0.1077	0.0420	0.0267	0.0152	0.0220
10 × 60	0.2489	0.1505	0.0518	0.0331	0.0294	0.0246
10 × 80	0.3321	0.2322	0.0753	0.0525	0.0376	0.0375
10 × 100	0.5021	0.3797	0.1209	0.0826	0.0607	0.0528

Table 4 Experimental results of MVO, ODDE and hmGHS

Instance	MVO	ODDE	hmGHS
5×20	0.0197	0.0481($s+$)	0.0876 ($s+$)
5×40	0.0563	0.0376 ($\sim$)	0.1335 ($s+$)
5×60	0.0534	0.0809 ($s+$)	0.1822 ($s+$)
5×80	0.0964	0.1661 ($s+$)	0.0754 ($\sim$)
5×100	0.0555	0.0751 ($\sim$)	0.1756 ($s+$)
10×20	0.0062	0.2398 ($s+$)	0.1012 ($s+$)
10×40	0.0340	0.1050 ($s+$)	0.2351 ($s+$)
10×60	0.0245	0.0980 ($s+$)	0.3583 ($s+$)
10×80	0.0502	0.0885 ($s+$)	0.1102 ($s+$)
10×100	0.0325	0.0840 ($s+$)	0.3253 ($s+$)

on 11 test instances. These experimental results can clearly demonstrate the superiority of the proposed MVO algorithm in solving the investigated FSSP with deteriorating jobs.

Conclusions

In this paper, a novel multi-verse optimizer (MVO) is experimentally investigated for solving the scheduling problem with deteriorating jobs in a m-machine ($m > 3$) flowshop manufacturing system. To the best of our knowledge, this is the first reported application of MVO for the flowshop scheduling problem with deteriorating jobs. The novelty of the proposed algorithm lies in a new elitist selection scheme that constructs the effective white/black hole tunnels among universes, and two different local search methods that enhance the exploitation capacity of the algorithm. The results of a series of experiments indicate that the two key features are helpful to improve the performance of the novel MVO. We can conclude that the novel MVO algorithm is a good optimizer in solving the FSSP with deteriorating jobs.

There are some future work to be done. First, it is important to analyze the sensitivity of parameters, such as pop_size, $elite\_size$ and ls_size, upon the performance of the proposed MVO algorithm. It is noted that these parameters were determined based on some primary experiments. Second, it is interesting to develop an adaptive method to set the values of these parameters. Third, it is desirable to employ the MVO algorithm to solve other scheduling and combinatorial optimization problems.

Acknowledgements We would like to thank the anonymous editor and reviewers for their thoughtful suggestions and constructive comments. This paper was partially supported by National Science Foundation of China under Grants 71671032, 71571156, 61703290 and 71620107003, and Fundamental Research Funds for the Central Universities Grant N160402002.

References

Alidaee, B., & Womer, N. K. (1999). Scheduling with time dependent processing times: Review and extensions. *Journal of the Operational Research Society*, *50*, 711–720.

Bachman, A., & Janiak, A. (2000). Minimizing maximum lateness under linear deterioration. *European Journal of Operational Research*, *126*(3), 557–566.

Browne, S., & Yechiali, U. (1990). Scheduling deteriorating jobs on a single processor. *Operations Research*, *38*(1), 495–498.

Costa, A., Cappadonna, F. A., & Fichera, S. (2017). A hybrid genetic algorithm for minimizing makespan in a flow-shop sequence-dependent group scheduling problem. *Journal of Intelligent Manufacturing*, *28*(6), 1269–1283.

Cheng, T. C. E., & Ding, Q. (2001). Single machine scheduling with step deteriorating processing times. *European Journal of Operational Research*, *134*(1), 623–630.

Cheng, T. C. E., Ding, Q., Kovalyov, M. Y., Bachman, A., & Janiak, A. (2003). Scheduling jobs with piecewise linear decreasing processing times. *Naval Research Logistics*, *50*(6), 531–554.

Cheng, T. C. E., Ding, Q., & Lin, B. M. T. (2004). A concise survey of scheduling with time-dependent processing times. *European Journal of Operational Research*, *152*, 1–13.

Cheng, M. B., Tadikamalla, P. R., Shang, J. F., & Zhang, S. Q. (2014). Bicriteria hierarchical optimization of two-machine flow shop scheduling problem with time-dependent deteriorating jobs. *European Journal of operational research*, *234*(3), 650–657.

Chu, C. B. (1992). A branch-and-bound algorithm to minimize total flow time with unequal release dates. *Naval Research Logistics*, *39*(6), 859–875.

Dao, T. K., Pan, T. S., & Pan, J. S. (2018). Parallel bat algorithm for optimizing makespan in job shop scheduling problems. *Journal of Intelligent Manufacturing*, *29*(2), 451–462.

Fu, Y. P., Wang, H. F., Huang, M., & Wang, J. W. (2017). A decomposition based multiobjective genetic algorithm with adaptive multipopulation strategy for flowshop scheduling problem. *Natural Computing*,. https://doi.org/10.1007/s11047-016-9602-1.

Fu, Y. P., Ding, J. L., Wang, H. F., & Wang, J. W. (2018). Two-objective stochastic flow-shop scheduling with deteriorating and learning effect in Industry 4.0-based manufacturing system. *Applied Soft Computing*. https://doi.org/10.1016/j.asoc.2017.12.009.

Fu, Y. P., Wang, H. F., Tian, G. D., Li, Z. W., & Hu, H. S. (2018). Two-agent stochastic flow shop deteriorating scheduling via a hybrid multi-objective evolutionary algorithm. *Journal of Intelligent Manufacturing*,. https://doi.org/10.1007/s10845-017-1385-4.

Gao, K. Z., Suganthan, P. N., Pan, Q. K., Chua, T. J., Cai, T. X., & Chong, C. S. (2016). Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives. *Journal of Intelligent Manufacturing*, *27*(2), 363–374.

Gawiejnowicz, S., Lee, W. C., Lin, C. L., & Wu, C. C. (2011). Single-machine scheduling of proportionally deteriorating jobs by two agents. *Journal of the Operational Research Society*, *62*, 1983–1991.

Gupta, J. N. D., & Gupta, S. K. (1988). Single facility scheduling with nonlinear processing times. *Computers & Industrial Engineering*, *14*(4), 387–393.

He, Z. G., Guo, Z. X., & Wang, J. W. (2018). Integrated scheduling of production and distribution operations in a global MTO supply chain. *Enterprise Information Systems*,. https://doi.org/10.1080/17517575.2018.1428770.

Jeng, A. A. K., & Lin, B. M. T. (2005). Minimizing the total completion time in single-machine scheduling with step-deteriorating jobs. *Computers & Operations Research*, *32*(3), 521–536.

Ji, W. B., & Zun, Q. X. (2005). Scheduling jobs under decreasing linear deterioration. *Information Processing Letters*, *94*(2), 63–69.

Kunnathu, A. S. (1990). Minimizing the makespan with late start penalties added to processing times in a single facility scheduling problem. *European Journal of Operational Research*, *47*(1), 56–64.

Lee, W. C., Shiau, Y. R., Chen, S. K., & Wu, C. C. (2010). A two-machine flowshop scheduling problem with deteriorating jobs and blocking. *International Journal of Production Economics*, *124*, 188–197.

Lee, W. C., Yeh, W. C., & Chung, Y. H. (2014). Total tardiness minimizationin permutation flowshop with deterioration consideration. *Applied Mathematical Modelling*, *38*(13), 3081–3092.

Li, X. T., & Yin, M. H. (2013). An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure. *Advanced Engineering Software*, *55*, 10–31.

Mirjalili, S., Mohammad, S., & Hatamlou, A. (2016). Multi-verse optimizer: A nature-inspired algorithm for global optimization. *Neural Computing and Applications*, *27*, 495–513.

Mosheiov, G. (1994). Scheduling jobs under simple linear deterioration. *Computers & Operations Research*, *21*, 653–659.

Ng, C. T., Cheng, T. C. E., Bachman, A., & Janiak, A. (2002). Three scheduling problems with deteriorating jobs to minimize the total completion time. *Information Processing Letters*, *81*(6), 327–333.

Shiau, Y. R., Lee, W. C., Wu, C. C., & Chang, C. M. (2007). Two-machine flowshop scheduling to minimize mean flow time under simple linear deterioration. *International Journal of Advanced Manufacturing Technology*, *34*, 774–782.

Sundararaghavan, P. S., & Kunnathur, A. S. (1994). Single machine scheduling with start time dependent processing times: Some solvable cases. *European Journal of Operational Research*, *78*(3), 394–403.

Ta, Q. C., Billaut, J. C., & Bouquard, J. L. (2018). Matheuristic algorithms for minimizing total tardiness in the m-machine flow-shop scheduling problem. *Journal of Intelligent Manufacturing*, *29*(3), 617–628.

Ventura, J. A., & Yoon, S. H. (2013). A new genetic algorithm for lot-streaming flow shop scheduling with limited capacity buffers. *Journal of Intelligent Manufacturing*, *24*(6), 1185–1196.

Wang, H. F., Fu, Y. P., Huang, M., & Wang, J. W. (2016). Multiobjective optimisation design for enterprise system operation in the case of scheduling problem with deteriorating jobs. *Enterprise Information Systems*, *10*(3), 268–285.

Wang, H. F., Fu, Y. P., Huang, M., Huang, G. Q., & Wang, J. W. (2017). A NSGA-II based memetic algorithm for multiobjective parallel nonidentical flowshop scheduling problem. *Computers & Industrial Engineering*, *113*, 185–194.

Wang, J. B., & Wang, M. Z. (2013). Minimizing makespan in three-machine flow shops with deteriorating jobs. *Computers & OperationsResearch*, *40*(2), 547–557.

Wang, J. W., Dou, R. L., Muddada, R. R., & Zhang, W. J. (2017). Management of a holistic supply chain network for proactive resilience: Theory and case study. *Computers & Industrial Engineering*,. https://doi.org/10.1016/j.cie.2017.12.021.

Wang, J. W., Muddada, R. R., Wang, H. F., Ding, J. L., Lin, Y., & Zhang, W. J. (2016). Towards a resilient holistic supply chain network system: Concept, review and future direction. *IEEE Systems Journal*, *10*(2), 410–421.

Wang, L., Pan, Q. K., & Tasgetiren, M. F. (2011). A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem. *Computer Industrial Engineering*, *61*(1), 76–83.

Wu, C. C., & Lee, W. C. (2006). Two-machine flow shop scheduling to minimize mean flow time under linear deterioration. *International Journal of Production Economics*, *103*, 572–584.

Wu, W. H., Wu, W. H., Chen, J. C., Lin, W. C., Wu, J. P., & Wu, C. C. (2015). A heuristic-based genetic algorithm for the two-machine flowshop scheduling with learning consideration. *Journal of Manufacturing Systems*, *35*, 223–233.

Zhong, R. Y., Huang, G. Q., Lan, S., Dai, Q. Y., Zhang, T., & Xu, C. (2015). A two-level advanced production planning and scheduling model for RFID-enabled ubiquitous manufacturing. *Advanced Engineering Informatics*, *29*(4), 799–812.