Scientia Iranica E (2018) 25(3), 1688–1700

SCIENTIA
IRANICA

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

On the n-job, m-machine permutation flow shop scheduling problems with makespan criterion and rework

B. Bootaki[a] **and M.M. Paydar**[b,*]

a. *Department of Industrial Engineering, Mazandaran University of Science and Technology, Babol, Iran.*
b. *Department of Industrial Engineering, Babol Noshirvani University of Technology, Babol, Iran.*

Received 9 August 2015; received in revised form 3 December 2016; accepted 6 March 2017

KEYWORDS
Permutation flow shop;
Rework;
Metaheurictics;
Simulation;
One-way ANOVA.

Abstract. This paper addresses n-job, m-machine Permutation Flow Shop Scheduling Problem (PFSSP) with unlimited intermediate buffers and rework activities. The concept of rework means that processing of a job on a machine may not meet a predefined quality level through its first process. Thus, we have a probabilistic cycle of operations for jobs on different machines based on two concepts: (1) A failure probability of a job on a machine; and (2) A descent rate that reduces processing times for the rework phase. In this case, the processing times of jobs on machines become random variables with a known probability distribution. The aim of this paper is to examine possible solution approaches to generate efficient job sequences with the least potential makespan. A wide range of simulation-based approaches are applied to address the proposed problem. These methods contain mathematical formulation, heuristic algorithms, and metaheuristics. The mechanism of the solution approaches is based on, firstly, using expected processing times to find a job sequence and, secondly, on evaluating the obtained job sequences by several simulated trials. Using the one-way ANOVA test, these methods have been compared together, and the results show the superiority of metaheuristics, especially simulated annealing, over the other methods.

1. Introduction

Flow shop systems and rework activities have recently become fully operational and critical in industries. Due to the novelty of the rework assumption in academic papers, there are many omitting fields and models to take into account. This paper deals with the application of the rework concept in a flow-shop system with a simulation and statistic viewpoint. The proposed configuration considers cyclic rework activities for a specific job on different machines to improve the quality of the job to the predefined standard. This framework holds on two different parameters:

1. The probability of a job failure to reach the standard (this type of activity can be seen at online inspection, such as using go/no-go gauges);
2. A descending rate reducing the processing times of jobs for the prospective rework activities.

Amending imperfect parts is sometimes an unavoidable task in manufacturing systems. The rework activities are important for saving the valuable raw materials. Rework activities can be seen at semiconductor, glass, steel, pharmaceutical, and food [1]. The

*. *Corresponding author. Fax: +98 1132310973*
E-mail addresses: behrang.bootaki@gmail.com (B. Bootaki); paydar@nit.ac.ir (M.M. Paydar)

doi: 10.24200/sci.2017.4443

main reasons for producing imperfect parts/jobs may be counted as process variations, imperfect technology, unfit machines, incompetent maintenance, unsteady manufacturing systems, and human errors [2]. Therefore, due to the importance of rework activities in the management and modeling of manufacturing processes, many articles in this field have emerged. According to the literature, there are totally three different types of rework configurations in manufacturing systems:

1. Imperfect parts remanufactured at the end of production cycle on the same machines;
2. A dedicated machine reworking imperfect parts;
3. Parallel machines doing both work and rework activities.

Inderfurth et al. [3] studied the problem of scheduling the production of main and defective items of a product for the same facility. The objective is to find batch sizes and positions of items to be reworked so that a given number of good-quality items can be produced and the total setup, rework, inventory holding, shortage, and disposal costs can be minimized. Chiu et al. [4] presented a model to determine the optimal lot-size for a production system with rework, i.e. a random scrap rate and a service level constraint. To prevent the loss of future sales due to customer's dissatisfaction, they considered a minimal service level per production cycle. Sarker et al. [5] developed models for the optimum batch quantity in a multi-stage system with rework process for two different operational policies: rework within the same cycle with no shortage versus rework done after N cycles, incurring shortage in each cycle. Liu et al. [6] considered a production inventory system with rework where a stationary demand is satisfied either by production setup with new raw materials or by rework setup with defective items coming from the production process. They presented mathematical models to find the optimal number of production and rework setups in a cycle, their sequence, and the economic production quantity of each setup simultaneously. Gribkovskaia et al. [7] proposed dedicated machines to implement the work and rework processes. A main facility is dedicated to the original production and a secondary one to re-manufacturing defective units coming from the main facility. After inspection, defective units of the inspected batch are transported to the re-manufacturing facility. The problem is to find a sequence of batch sizes, such that the makespan is minimized. A linear programming model was suggested to handle this problem. Kang et al. [8] studied a dispatching algorithm to solve a parallel machine scheduling problem with rework possibilities. The performance of the proposed algorithm was evaluated in terms of six performance indicators: total tardiness, maximum lateness, mean flow time, mean lateness, the number of reworks, and the number of tardy jobs, respectively. Ramezanian and Saidi-Mehrabad [9] addressed the problem of minimizing the makespan in a multi-product unrelated parallel machine scheduling problem with the possibility of producing imperfect jobs. They developed some heuristic methods based on dispatching rules in order to find the best sequence of work and rework processes on the machines. Taleizadeh et al. [10] presented an imperfect production system with multi-products, partial backordering shortage, service level, and budget constraints, in which the imperfectly produced items are supposed to get reworked. Sarkar et al. [11] introduced an EPQ model with rework process in a single-stage manufacturing system with planned backorders and random defective rates. They developed three different inventory models for three different distribution density functions at defective rates, such as uniform, triangular, and beta. Closed-form solutions for determining the economic production quantity of each inventory model were provided. Hossain and Sarker [12] considered the N-stage serial production line with an inspection station at the end of it to make decisions concerning imperfect products. They assumed that after inspecting items, each product may have several defect types requiring that it be sent back to the proper workstations or be repaired off-line on a dedicated rework station. The aim of their paper is to decide which products are better to rework on-line or off-line in order to minimize the unit cost of production. Beynaghi et al. [13] developed a hybrid solution approach based on Genetic Algorithm (GA), Variable Naighbourhood Search (VNS), and Simulated Annealing (SA) for batching work and rework processes on a single machine with the aim of minimizing makespan. Aging effect is a new assumption in their paper that makes a decision-maker determine the number and size of batches. Moussawi-Haidar et al. [14] considered an imperfect manufacturing process with two different scenarios to deal with the defective items produced, either selling them at a discount rate, or reworking them. An expected profit function with some closed-form expressions for the optimal production lot size has been derived for their proposed problem.

Based on the extensive literature on the imperfect production systems, there is a growing interest in the issue, which is obviously a real-world manufacturing sector concern. In this paper, a new type of rework configuration is presented which is widely used in manufacturing environments. A configuration with a flow line of machines in presented wherein operators at each stage check online if the quality of the part at hand reaches the standards (e.g., using go/no-go gauges) to pass to the next machine buffer or not. In this situation, the operator may succeed in producing a perfect job through their first run. However, it may need more than one single run for the part to pass certain desirable standards. It has been considered that the rework processing times of jobs decrease with a descending rate. The aim of this paper is to find which sequence of jobs/parts may lead less makespan to the manufacturing system.

The solution approach in this paper contains three different steps:

1. Processing times are presented as random variables with a specific probability distribution (herein, geometric variables);
2. Some solution approaches toward deterministic n-job, m-machine permutation flow shop scheduling (mathematical modeling, heuristics and meta-heuristics) are applied with the expected processing times to find job sequences;
3. Job sequences of each method are compared together by running several simulated trials and using one-way ANOVA test to find the best ones.

The rest of the paper is organized as follows. Section 2 describes the rework configuration discussed in this paper. Section 3 describes the solution approaches. Section 4 presents an illustrative example to better understand both the effect of rework issue on sequencing jobs on different machines and how to compare different solution approaches together using simulated trials and ANOVA test. Section 5 shows the computational results and compares the performances of the different solution methods. Finally, Section 6 concludes the paper and presents possible future interests.

2. Problem formulation

Before describing the proposed problem, some notations are necessary to be mentioned here.

m	Number of machines;
n	Number of jobs;
i	Index of machines ($i = 1, 2, \cdots, m$);
j	Index of jobs ($j = 1, 2, \cdots, n$);
l	Index of operations ($l = 1, 2, \cdots, L$);
t_{ji}	Main processing time of job j on machine i;
$\hat{P}_{ji}$	Processing time of job j on machine i, which is a discrete random variable;
P_{lji}	Processing time of the lth operation of job j on machine i (if $l = 1$, it means the main process, otherwise it means rework processes);
p_i	Defect probability of jobs on machine i;
λ	Descent rate;
S_{ki}	Start time of the kth job in the sequence on the ith machine;
F_{ki}	Finish time of the kth job in the sequence on the ith machine;
X_{jk}	Is 1, if job j is the kth job in the sequence; otherwise 0;
$\bar{C}_{\max}$	Minimum expected makespan.

Consider the n-job, m-machine PFSSP with unlimited intermediate buffers. Each job on a machine may reach the predefined qualification level by more than one processing operations. After processing, a job will be checked if it is qualified. If so, the job will proceed toward the next machine buffer; otherwise, it will be back to the current machine for processing again immediately. Therefore, operations of a job on a machine can vary between 1, 2, 3, ..., stochastically. The defect probability of jobs on machine i is a Bernoulli trial with parameter p_i. Figure 1 shows the schematic of the proposed problem.

It is clear that the number of operations required so that a job completes its process on machine i is equivalent to that of random Bernoulli trials to achieve the first success, where the probability of success equals $1 - p_i$. In other words, the number of operations is a random variable with geometric distribution with parameter $1 - p_i$ and the following formula:

$$\Pr(X = x) = p_i^{x-1}(1 - p_i); \qquad \text{for } x = 1, 2, \cdots. \quad (1)$$

Also, we consider that if a job needs rework, its processing time declines by the multiplication of a constant rate of λ. Thus, the lth operation of job j on machine i is equal to $P_{lji} = \lambda^{l-1} \times t_{ji}$. According to Eq. (1), the processing times are discrete random variables with the following function:

$$\hat{P}_{ji} : \Pr\left(P_{lji} = \sum_{k=1}^{l} \lambda^{k-1} \times t_{ji}\right) = p_i^{l-1}(1 - p_i);$$

$$\text{for } l = 1, 2, \cdots. \quad (2)$$

Cumulative distribution function in Eq. (3) is used to

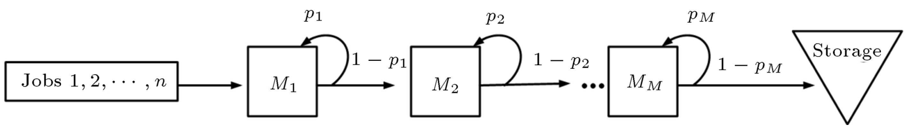

Figure 1. Schematic of the proposed flow shop with rework loops.

generate random processing times through producing random numbers between (0, 1) and matching them to the corresponding category. Mathematical expectation of processing times can be calculated as in Eq. (4):

$$F_{\hat{P}_{ji}}(x):\Pr\left(\sum_{l=1}^{x}\sum_{k=1}^{l}\lambda^{k-1}\times t_{ji}\right)=\sum_{l=1}^{x}p_i^{l-1}(1-p_i)$$

$$\text{for } x=1,2,\cdots, \tag{3}$$

$$\begin{aligned}
E\left(\hat{P}_{ji}\right) &= \sum_{l=1}^{\infty}P_{lji}\times\Pr(P_{lji})\\
&= \sum_{l=1}^{\infty}\left[\left(\sum_{k=1}^{l}\lambda^{k-1}\times t_{ji}\right)\times p_i^{l-1}(1-p_i)\right]\\
&= t_{ji}(1-p_i)\sum_{l=1}^{\infty}\left[\left(\sum_{k=1}^{l}\lambda^{k-1}\right)\times p_i^{l-1}\right]\\
&= t_{ji}(1-p_i)\sum_{l=1}^{\infty}\frac{1-\lambda^l}{1-\lambda}\times p_i^{l-1}\\
&= t_{ji}\left(\frac{1-p_i}{1-\lambda}\right)\sum_{l=1}^{\infty}(1-\lambda^l)\times p_i^{l-1}\\
&= \frac{t_{ji}}{p_i}\left(\frac{1-p_i}{1-\lambda}\right)\sum_{l=1}^{\infty}(1-\lambda^l)\times p_i^{l}\\
&= \frac{t_{ji}}{p_i}\left(\frac{1-p_i}{1-\lambda}\right)\sum_{l=1}^{\infty}\left[p_i^l-(\lambda p_i)^l\right]\\
&= \frac{t_{ji}}{p_i}\left(\frac{1-p_i}{1-\lambda}\right)\left[\frac{p_i}{1-p_i}-\frac{\lambda p_i}{1-\lambda p_i}\right]\\
&= \frac{t_{ji}}{1-\lambda}-\frac{t_{ji}}{1-\lambda}\left(\frac{1-p_i}{1/\lambda-p_i}\right)\\
&= \frac{t_{ji}}{1-\lambda}\left[1-\frac{1-p_i}{1/\lambda-p_i}\right]\\
&= \frac{t_{ji}}{1-\lambda}\left[\frac{1/\lambda-p_i-1+p_i}{1/\lambda-p_i}\right]\\
&= \frac{t_{ji}}{1-\lambda}\left(\frac{1/\lambda-1}{1/\lambda-p_i}\right).
\end{aligned} \tag{4}$$

Some other assumptions are as follows:

- No preemption of jobs is allowed;
- Set-up, inspection, and handling times have been included in the main processing times;
- All jobs are available for processing at time zero;
- Machines are available at all times.

3. Solution methods

Totally, the solution approaches to PFSSPs can be classified into three categories: exact methods, heuristic algorithms, and metaheuristics. Among the exact methods, we can refer to complete enumeration, mathematical modeling, dynamic programming, branch-and-bound approaches, elimination rules, and row generation algorithms [15].

Because of the NP-hardness of flow shop scheduling class [16], computational difficulties of the exact methods are severe for large problems. Consequently, abundant heuristics and metaheuristics have been developed during the past six decades. Heuristic algorithms are mostly based on creative lemma and basic rules. Some heuristics for PFSSPs with makespan criterion can be found in [17-20].

Metaheuristics are algorithms inspired by basic rules followed in nature. All of them start with an initial solution or a set of solutions, and try to make them better and better through several iterations with the aim of escaping from local optimums. There are many metaheuristic algorithms that have been implemented on flow shop scheduling problems with makespan criterion including: Simulated Annealing (SA), Genetic Algorithm (GA), Tabu Search (TS), greedy approaches, Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC) algorithm, and Variable Neighborhood Search (VNS). Additionally, hybrid algorithms, combining some of these algorithms, have been developed in the literature [21].

In this paper, we choose some early constructive heuristic algorithms that have been clues for other researchers, including Palmer's [17], CDS [18], Gupta's [19], and NEH [20] heuristics. Also, some strong and basic metaheuristic approaches, such as GA, SA, and VNS, were developed for the proposed problem. Actually, another factor in selecting the proposed methods is their simplicity of coding and intelligibility.

3.1. Mathematical modeling approach

$$\min \bar{C}_{\max}.$$

Subject to:

$$\bar{C}_{\max}\geq F_{nm}, \tag{5}$$

$$S_{11}=0, \tag{6}$$

$$S_{k1}=F_{(k-1)1}, \qquad k\geq 2, \tag{7}$$

$$F_{ki}\approx S_{ki}+\sum_{j=1}^{n}X_{jk}\hat{P}_{ji}, \qquad \forall\, k,i, \tag{8}$$

$$S_{1i}=F_{1(i-1)}, \qquad i\geq 2, \tag{9}$$

$$S_{ki} = \max\left\{F_{k(i-1)}, F_{(k-1)i}\right\}, \qquad k, i \geq 2, \tag{10}$$

$$\sum_{k=1}^{n} X_{jk} = 1, \qquad \forall\, j, \tag{11}$$

$$\sum_{j=1}^{n} X_{jk} = 1, \qquad \forall k, \tag{12}$$

$$S_{ki} \quad \text{and} \quad F_{ki} \geq 0; \qquad X_{jk} \in \{0,1\}. \tag{13}$$

The objective function and Inequation (5) together ensure that the minimum expected makespan equals to the finish time for the last job on the last machine. Eq. (6) determines that the first job in sequence on the first machine starts at time zero. Eq. (7) shows that the start time for the kth job in sequence (for $k \geq 2$) on the first machine is equal to the finish time for the preceding job on the first machine. Eq. (8) computes the finish time for jobs on different machines according to their start time and processing time. Eq. (9) indicates that the start time for the first job in sequence on a machine (for $i \geq 2$) equals its finish time on the previous machine. Eq. (10) shows that the start time for the kth job in sequence (for $k \geq 2$) on a machine (for $i \geq 2$) equals the bigger value between the finish time for the kth job on the previous machine and that for the preceding job on the current machine. Eqs. (11) and (12) are used to make unique assignments for jobs and their positions in sequence to calculate the processing times in Eq. (8). Inequation (13) determines the nature of decision variables of the model.

3.2. Heuristic algorithms

In this subsection, the implementation of the heuristics Palmer, Gupta, CDS, and NEH is discussed in detail. Generally, these heuristics are developed to solve deterministic PFSSPs. However, these heuristics can be applied to our stochastic problem by replacing the processing times with their expected values obtained by Eq. (4).

3.2.1. Palmer's heuristic

Palmer [17] introduced a "slope index" for each job in the sequence which is calculated through the following formula:

$$SI_j = -\sum_{i=1}^{m} \left[m - (2i-1)\right] t_{ji}/2. \tag{14}$$

Then, descending order of slope indexes finds the efficient job sequence as follows:

$$SI_{[1]} \geq SI_{[2]} \geq \cdots \geq SI_{[n]}. \tag{15}$$

3.2.2. CDS heuristic

The CDS algorithm by Campbell et al. [18] generates $m-1$ artificial two-machine sub-problems which can be solved by Johnson's rule (see [22]). Then, the sequence of the sub-problem with the minimum makespan indicates the efficient job sequence. Processing times for the kth artificial two-machine sub-problems for the jth job on the ith machine should be calculated as follows:

$$t'_{j1} = \sum_{i=1}^{k} t_{ji}, \qquad \text{and} \qquad t'_{j2} = \sum_{i=m-k+1}^{m} t_{ji}. \tag{16}$$

3.2.3. Gupta's heuristic

Gupta [19] introduced another job index for $m > 2$ below, which like the former one, the descending order of job indexes shows the efficient job sequence:

$$SI_j = e_j / \min_{1 \leq k \leq m-1} \left\{t_{jk} + t_{j(k+1)}\right\},$$

where:

$$e_j = \begin{cases} 1 & \text{if } t_{j1} < t_{jm} \\ -1 & \text{if } t_{j1} \geq t_{jm} \end{cases} \tag{17}$$

3.2.4. NEH heuristic

NEH heuristic by Nawaz et al. [20] is based on the assumption that a job with more total processing times on all machines should be given higher priority compared to those with less total processing times on all machines. Algorithm 1 shows the NEH heuristic's procedure.

3.3. Metaheuristic methods

A general step for implementing every metaheuristic is to define a solution representation. In this paper, the solutions are shown by an ordinal string of numbers between one to the number of jobs. The position of each digit in the sequence determines which job is done earlier. For example, consider the string [5 2 3 1 4]. This means that Job 5 is processed first; then, Job 2 is processed, and so on. In the following, other elements and steps of the proposed metaheuristics are briefly described.

3.3.1. Genetic algorithm

GA is a powerful population-based mataheuristic which adopts the principles of natural evolution of organisms. GA was initially developed by Holland [23]. In GA, an initial population of solutions is generated, and then by applying three different operators called crossover, mutation, and copy, the new population is reproduced. The crossover operator merges the qualifications of two selected solutions as parents to make offspring with similarities to each parent. The Roulette Wheel selection procedure, as proposed by Goldberg [24], is the selection strategy for crossover operator. The goal of Roulette Wheel strategy is to allow the "fittest" solutions to be considered more often to reproduce children for the next generation. In a minimizing problem, the

```
Step 1: For all jobs, calculate $T_j = \sum_{i=1}^{m} t_{ji}$
Step 2: Arrange the jobs in descending order of $T_j$'s.
Step 3: Pick up the first and second jobs from the list of Step 2
        and find the best sequence for these two jobs by calculating
        makespan for the two possible sequences. Do not change the
        relative positions of these two jobs with respect to each other
        to the end of algorithm. Set $j = 3$.
Step 4: Pick up the $j$th job in the list generated in Step 2 and find
        the best sequence by placing it at all possible $j$ positions in
        the partial sequence found in the previous step, without
        changing the relative positions to each other of the already
        assigned jobs.
Step 5: If $j = n$, stop; otherwise, set $j = j + 1$ and go to Step 4.
```

Algorithm 1. NEH heuristic's procedure.

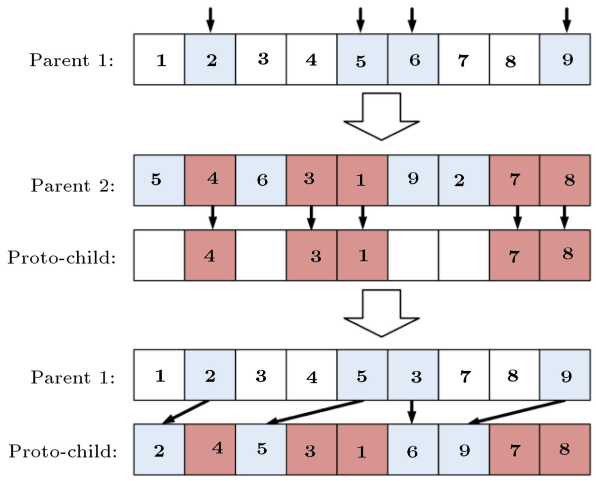

Figure 2. The schematic for order-based crossover.

opposite of the objective function is usually selected as the fitness of each individual. Order-based crossover (OBX) is chosen for the proposed crossover operator with the following steps. Also, Figure 2 depicts these steps for two random solutions.

Step 1. Select a set of positions from parent 1 at random;

Step 2. Produce proto-child by copying non-selected values from parent 2;

Step 3. produce offspring by copying selected values from parent 1.

Despite crossover, mutation operator makes aimless changes to the solutions with the hope of escaping from local optimums. Thus, the selection strategy for mutation operator is the random selection. The mutation operator used in this paper is swapping operator which replaces two randomly selected gens together. Figure 3 shows the swapping mutation.

Copy or reproduction operator duplicates some of the best solutions into the next generation without

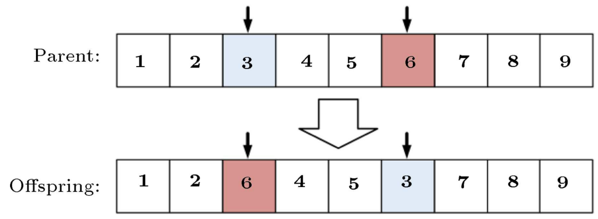

Figure 3. The schematic for the swapping mutation.

any change. This procedure guarantees that the next generation is as good as the previous one. GA has been fully applied to numerous problems in the literature; thus, for more information, the reader is referred to some of the recent works [25-27].

3.3.2. Simulated annealing

SA algorithm is a local search algorithm inspired by the annealing technique used by the metallurgists to obtain a "well-ordered" solid with minimal level of energy. This process is done by slowly cooling the high-temperature solid. The cooling process in SA makes the diversification role and prevents the embedded local search technique from being trapped in local optimum. At the start of the algorithm, when the temperature is high, the probability of selecting worse neighbors is high, and then this rate decreases by temperature abatement. SA was first introduced by Kirkpatrick et al. [28] in order to solve hard combinatorial optimization problems. Algorithm 2 shows the pseudocode for SA. The function for decreasing the temperature at each stage is $T = \beta \times T$. Parameter β is known as the rate of cooling or decrement factor. In general, convergence speed and solution quality both depend on β. As β increases the CPU time, solution quality increases. To find neighbors for a solution, the swapping operator is used.

3.3.3. Variable neighborhood search

Variable Neighborhood Search (VNS) is a simple and effective trajectory meta-heuristic firstly proposed by

```
Choose, at random, an initial solution $s$ for the system to be optimized;
Initialize temperature $T$;
while the stopping criterion is not satisfied do
    repeat
        Randomly select $s' \in N(s)$;
        if $f(s') \leq f(s)$ then
            $s \leftarrow s'$;
        else
            $s \leftarrow s'$ with a probability of $e^{-\frac{f(s')-f(s)}{T}}$;
        end
    until the inner cycle stopping criterion is met;
    Decrease $T$;
end
return the best solution met.
```

Algorithm 2. Pseudocode for SA algorithm.

Table 1. Main processing times.

	Machine 1	Machine 2	Machine 3	Machine 4	Machine 5
Job 1	10	8	9	10	1
Job 2	7	10	4	1	2
Job 3	2	5	3	5	10
Job 4	8	1	1	7	3
Job 5	10	4	5	1	8
Job 6	2	2	5	8	10
Job 7	7	10	9	8	3
Job 8	7	1	1	6	8
Job 9	9	5	2	7	9
Job 10	2	4	3	6	4
Defect probabilities	0.13	0.09	0.08	0.05	0.14

Mladenovic and Hansen [29]. The basis of VNS is using more than one neighborhood structure, and systematically changing the neighborhood within a local search process. Unlike many other meta-heuristics, the basic scheme of VNS and its extensions requires few and sometimes no parameters. After an initial solution is generated, the main cycle of VNS begins. This cycle has three steps: shaking, local search, and move. In the shaking step, a solution, s', is randomly selected in the nth neighborhood of current solution, s. Then, s' is used as the initial solution of a local search procedure, till solution s'' is generated. The local search can use any neighborhood structure regardless of the main neighborhood structure. At the end of the local search process, if s'' is better than s, then s'' replaces s and the cycle starts again with $n = 1$. Otherwise, the algorithm moves to the next neighborhood $n + 1$ and a new shaking phase starts using this neighborhood.

4. An illustrative example

In this section, an illustrative example is produced to have a better comprehension of the application of the proposed solution approaches. Consider a permutation flow shop with 10 jobs and 5 machines. The main processing times are presented in Table 1. The defect probabilities are also generated randomly and shown in Table 1. The descent rate for processing times is arbitrarily assumed 0.6. By means of this information and using Eq. (4), the expected processing times are calculated and recorded in Table 2. These values are used in all solution approaches to find a job sequence. Table 3 shows the job sequence obtained by each method and their elapsed times.

To evaluate the average makespan for each sequence, Algorithm 3 should be done.

Let us run this example for a random trial.

Table 2. Expected processing times.

	Machine 1	**Machine 2**	**Machine 3**	**Machine 4**	**Machine 5**
Job 1	10.84	8.45	9.45	10.30	1.09
Job 2	7.59	10.57	4.20	1.03	2.18
Job 3	2.16	5.28	3.15	5.15	10.91
Job 4	8.67	1.05	1.05	7.21	3.27
Job 5	10.84	4.22	5.25	1.03	8.73
Job 6	2.16	2.11	5.25	8.24	10.91
Job 7	7.59	10.57	9.45	8.24	3.27
Job 8	7.59	1.057	1.05	6.18	8.73
Job 9	9.76	5.28	2.10	7.21	9.82
Job 10	2.16	4.22	3.15	6.18	4.36

Table 3. Efficient sequences by the solution methods.

	Job sequence										**Elapsed time (sec)**
Lingo 9	3	10	9	6	8	7	1	5	4	2	75
Palmer's	6	3	8	10	9	4	5	7	1	2	0.0016
CDS	10	3	8	6	9	7	1	5	2	4	0.0024
Gupta's	8	6	10	9	3	7	1	5	2	4	0.0011
NEH	10	8	3	6	7	9	1	2	5	4	0.0033
GA	6	3	10	7	9	8	1	2	5	4	20
SA	3	8	10	6	7	9	1	2	5	4	20
VNS	3	10	6	8	7	9	1	2	5	4	20

```
Step 1: Define the number of trials.
Step 2: Make random simulated processing times
        using cumulative distribution functions
        Eq. (3).
Step 3: Run job sequence for the table of processing
        times obtained from previous step; then,
        record the makespan.
Step 4: If the maximum number of trials is reached,
        go to Step 5; otherwise, go to Step 2.
Step 5: Report the average value for makespan.
```

Algorithm 3. Procedure for calculating the average makespan for a job sequence.

Regard Table 4 as random numbers for generating the processing times. With running the job sequences with these processing times, the makespans of different approaches are presented in Table 5. As shown in Table 5, NEH heuristic has the minimum makespan of 85.28 time units over the other methods.

It is clear that the result obtained from just a single performance is not dependable. Now, we are interested in finding the better solution approaches through a statistical manner using the one-way ANOVA technique in MINITAB® 14.1 software package. The number of simulated trials is considered to be 1000. The whole 1000 makespans generated for each method are inserted as a different column in the software. Then, the obtained data are analyzed to detect whether there is a significant difference between the means of makespans for different approaches. Algorithm 4 shows the procedure for finding the best set of methods at a confidence level of 95%.

```
Step 1: Do the one-way ANOVA for data
        set of all solution approaches.
Step 2: If $P$-Value is less than 0.05, go to
        Step 3; otherwise, report the
        remaining approaches as the best.
Step 3: Remove the worst approaches and
        do the one-way ANOVA test again
        for the others. Then, go to Step 2.
```

Algorithm 4. Procedure for finding the best set of solution methods.

Consecutively doing the one-way ANOVA test for the methods at hand, checking the P-Value, and removing the worst method in case of $P\text{-Value} < 0.05$ resulted in SA and VNS finally, as the best methods for this example. Due to the huge volume of the computations, only the final ANOVA test results are shown in Table 6. Because P-Value is bigger than 0.05

Table 4. Random numbers to produce processing times for the first trial.

	Random numbers						**Processing times**				
	M1	**M2**	**M3**	**M4**	**M5**		**M1**	**M2**	**M3**	**M4**	**M5**
Job 1	0.2539	0.5245	0.0298	0.1629	0.7777		10	8	9	10	1
Job 2	0.0882	0.5036	0.9640	0.7845	0.9776		7	10	6.40	1	3.20
Job 3	0.4951	0.1380	0.5501	0.9891	0.6019		2	5	3	8	10
Job 4	0.9241	0.6163	0.6331	0.4137	0.9882		12.80	1	1	7	5.88
Job 5	0.1847	0.7600	0.0713	0.8611	0.6344	$\Rightarrow$	10	4	5	1	8
Job 6	0.9618	0.0572	0.4742	0.5126	0.9422		3.20	2	5	8	16
Job 7	0.1485	0.2840	0.1797	0.2160	0.2119		7	10	9	8	3
Job 8	0.2900	0.8940	0.1992	0.2768	0.8104		7	1	1	6	8
Job 9	0.1864	0.9744	0.6400	0.2879	0.6136		9	8	2	7	9
Job 10	0.4286	0.2888	0.6209	0.9312	0.6618		2	4	3	6	4

Table 5. Comparison between the solution approaches for the first trial.

	Lingo 9	**Palmer's**	**CDS**	**Gupta's**	**NEH**	**GA**	**SA**	**VNS**
$C_{\max}$	90.6000	95.2000	87.4800	90.2800	85.2800	86.2800	86.0800	86.0800

Table 6. The one-way ANOVA test results of SA and VNS.

Source	**DF**	**SS**	**MS**	***F***	***P*-value**
Factor	1	40.5	40.5	1.49	0.222
Error	1998	54260.2	27.2		
Total	1999	54300.7			
Method	***N***	**Mean**	**StDev**		
SA	1000	88.125	5.069		
VNS	1000	88.41	5.35		

(0.222), the difference between these two approaches is not significant; therefore, they are known as the best set of solution approaches to this example.

5. Computational results

In this section, the proposed solution approaches are solved for several random test problems in three different categories. For small category, the number of jobs and machines ranges from 5 to 10 and 3 to 10, respectively. For medium category, the number of jobs and machines ranges from 15 to 30 and 5 to 25, respectively. In addition, for large category, the number of jobs and machines ranges from 50 to 100 and 10 to 50, respectively. Further, for all categories, descent rate λ is arbitrarily assumed to be 0.6, and the main processing times and defect rates are randomly generated from [1, 10] and [0.3, 0.5], respectively. The maximum number of simulated trials is considered to be 1000. The mathematical model is run on Lingo 9 software, and the heuristics and metaheuristics are coded in MATLAB R2010 (a). All test problems are solved by a notebook with an Intel® Core (TM) i5 2.4 GHz and 2 GB RAM. Table 7 shows the elapsed times and set of the best solution approaches to each test problem. The procedure for finding the best solutions is the same as that discussed in Algorithm 4. The elapsed times for the solution methods of each type were so close that we avoided presenting them distinctly.

Because the n-job m-machine flow shop sequencing problems belong to the class of NP-hard problems (see [16]), the computational requirements for obtaining an optimal solution increase exponentially as problem size increases. Therefore, as seen in Table 7, Lingo 9 cannot reach even a local optimum ever after 3600 seconds for most of the medium and large test problems. The heuristics are polynomial time algorithms, and their elapsed times barely exceed one second. However, the metaheuristics have a different structure and their convergence process needs to be completed during the algorithm performance. Thus, the comparison between the heuristics and metaheuristics in terms of computational time does not make sense. However, in real case, a maximum elapsed time of three or four minutes for metaheuristics may not affect the decision-maker's satisfaction level.

Table 8 summarizes the information in Table 7 and shows the percentage at which a method lies on the best approaches for each category and also for the

Table 7. Computational results of random test problems.

				CPU time (Seconds)			**Best approaches**							
No.	**Category**	n	m	**Lingo 9**	**Heuristics**	**Metaheuristics**	**Lingo 9**	**Palmer**	**CDS**	**Gupta**	**NEH**	**GA**	**SA**	**VNS**
1		5	3	$< 1^*$		10	✓	✓	✓	✓	✓			
2		5	5	12^*		10	✓		✓		✓	✓	✓	
3		7	3	1^*		14			✓		✓			
4		7	5	84^*		14				✓	✓			
5	Small	7	7	53^*	< 1	14	✓				✓	✓		
6		10	3	1^*		20	✓	✓	✓			✓		✓
7		10	5	246^*		20	✓					✓	✓	✓
8		10	7	254^*		20	✓					✓	✓	✓
9		10	9	1649^*		20	✓				✓	✓	✓	✓
10		10	10	1813^*		20	✓						✓	✓
11		15	5	181^*		30						✓		✓
12		15	10	3600^{**}		30								✓
13		15	15	—		30							✓	✓
14		20	10	—		40						✓	✓	✓
15	Medium	20	15	—	< 1	40								✓
16		20	20	—		40						✓		
17		30	10	—		60								✓
18		30	15	—		60						✓	✓	
19		30	20	—		60							✓	
20		30	25	—		60							✓	✓
21		50	10	—		100						✓	✓	
22		50	15	—		100							✓	
23		50	25	—		100							✓	
24		70	15	—		140								✓
25	Large	70	30	—	< 1	140							✓	
26		70	35	—		140						✓	✓	
27		90	20	—		180							✓	
28		90	25	—		180							✓	
29		90	30	—		180							✓	
30		100	50	—		200								✓

*: The global optimum was reached;

**: A local optimum was found after 3600 sec (not global optimum);

—: No feasible solution was found ever after 3600 sec.

Table 8. Percentage of being among the best methods for different categories of test problems.

	Lingo 9	Heuristics				Metaheuristics		
		Palmer	CDS	Gupta	NEH	GA	SA	VNS
Small-sized test problems	8/10	2/10	4/10	2/10	6/10	6/10	5/10	5/10
Medium-sized test problems	0/10	0/10	0/10	0/10	0/10	4/10	5/10	7/10
Large-sized test problems	0/10	0/10	0/10	0/10	0/10	2/10	8/10	2/10
Total	**8/30**	**2/30**	**4/30**	**2/30**	**6/30**	**12/30**	**18/30**	**14/30**

whole test problems. Accordingly, the following results can be concluded:

- Totally, SA is the best solution method, which is the best approach in 60% of all test problems. The next best approach is VNS with 46.67% of success. At the next grade, GA is another best solution in 40% of all test problems, and so on;
- Totally, the metaheuristic methods perform better than heuristics;
- Totally, NEH heuristic is the best method among other heuristics. The next better heuristic is CDS;
- In small-sized problems, Lingo 9 software acts the best among the other approaches with 80% of success. However, it fails to find any job sequences in a reasonable time for larger cases. In the next grade, GA and NEH perform better than the others with 60% of success;
- In the medium-sized problems, Heuristics act poorly, since they have never been placed among the best solutions in any test problems. Among the metaheuristics, VNS acts the best with 70% of success;
- In the large-sized problems, the heuristics still perform poorly. Among the metaheuristics, SA significantly dominates the other methods with 80% of success.

What makes a metaheuristic carry out its best performance is tuning the controlling parameters of the algorithm. There are some sorts of controlling parameters with the characteristic of "the larger, the better." Thus, as the size of the problem enlarges, the need for selecting a bigger value for these parameters increases. Outer/inner cycle performance duration of metaheuristics accounts for these types of parameters. To make an identical condition for the proposed metaheuristics, the termination rule for all is set to a maximum elapsed time of $2 \times n$ seconds. The constant n (number of jobs) is interpreted as the size of test problems. Also, the inner cycle duration is assumed to be as big as possible according to the full consideration of test problem configurations. Based on the computational difficulty viewpoint, the population size in GA behaves like the inner cycle iterations in SA and VNS algorithms. Correspondingly, the population

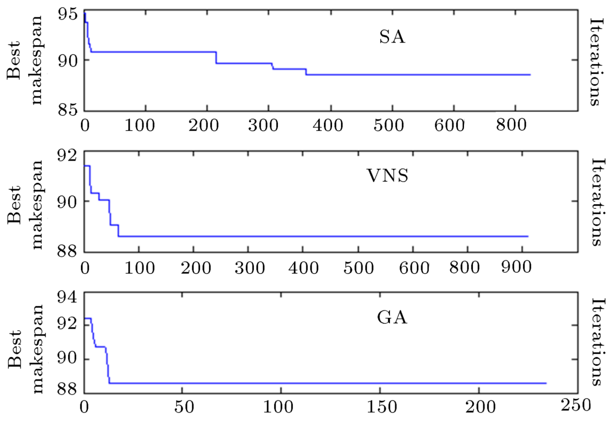

Figure 4. Convergence diagram for different metaheuristics.

Table 9. Other controlling parameters for GA and SA.

Algorithm	Controlling parameters
GA	Crossover rate = 0.7; Mutation rate = 0.01
SA	$T_0 = 400$; $\beta = 0.99$

size of GA and the inner cycle size for SA and VNS are assumed to be 1000. To make sure that this big inner cycle may not prevent the algorithms from completing their convergence, some random test problems within the predefined configurations were performed. Figure 4 depicts the convergence diagram of the proposed metaheuristics for a random test problem. This figure demonstrates that there is a good balance between the termination condition of $2 \times n$ seconds and the inner iteration size of 1000. Other controlling parameters for the metaheuristics are shown in Table 9.

The main concern that may question the computational section is how the change of descending parameter λ would affect the performance of solution approaches. To tackle this challenge, the authors have run the solution approaches for two other random generated instance packages of time with a value of $\lambda = 0.1$ (as a low rate) and $\lambda = 0.9$ (as a high rate). The results are almost similar to when $\lambda = 0.6$. Because of the enormous computations results, no more investigation is presented in this section.

6. Conclusions and remarks for future works

In this paper, a type of a stochastic flow shop system was introduced, in which the number of operations for a job on a machine is a geometric random variable. In other words, a job may need one or two or even more operations to be completed on a machine. Therefore, the processing times of jobs on different machines can be presented as random variables with known probability distributions considering the following two elements:

1. Defect rates of jobs on machines;
2. A descent rate for deriving the rework processing times from the main processing times.

Several solution approaches, including mathematical modeling, heuristics, and metaheuristics, were applied to minimize makespan. In all of the approaches, the mathematical expectations of processing times were applied. Then, in order to evaluate the obtained job sequences and compare the solution approaches together, 1000 simulated trials have been generated and analyzed using the one-way ANOVA test.

According to the results, the proposed metaheuristics, especially SA and VNS, have a significant superiority over the other methods. The mathematical modeling is just suitable for small-sized problems because it is unable to find any job sequences for larger problems in a reasonable time (less than one hour). Among the heuristics, NEH heuristic performs better for small-sized test problems. Also, the results show that if the number of jobs and machines is less than 10, it is better to use NEH heuristic due to its low elapsed time compared with the mathematical modeling and metaheuristics. For larger test problems (n and $m > 10$), only the metaheuristics can find efficient job sequences.

In fact, this paper is a pioneering work in applying rework assumption to flow shop systems. As a result, there may be many assumptions and models to consider for future researches. For example, the problem can be developed by additional constraints, including setup times, release times, due dates, limited buffers, etc. Also, the concept of rework can include configurations with backward movements.

References

1. Flapper, S.D.P. and Teunter, R.H. "Logistic planning of rework with deteriorating work-in-process", *International Journal of Production Economics*, **88**, pp. 51-59 (2004).
2. Cao, Y., Subramaniam, V., and Chen, R. "Performance evaluation and enhancement of multistage manufacturing systems with rework loops", *Computers & Industrial Engineering*, **62**, pp. 161-176 (2012).
3. Inderfurth, K., Janiak, A., Kovalyov, M.Y., and Werner, F. "Batching work and rework processes with limited deterioration of reworkables", *Computers & Operations Research*, **33**, pp. 1595-1605 (2006).
4. Chiu, S.W., Ting, C.K., and Chiu, Y.S.P. "Optimal production lot sizing with rework, scrap rate, and service level constraint", *Mathematical and Computer Modelling*, **46**, pp. 535-549 (2007).
5. Sarker, B.R., Jamal, A.M.M., and Mondal, S. "Optimal batch sizing in a multi-stage production system with rework consideration", *European Journal of Operational Research*, **184**, pp. 915-929 (2008).
6. Liu, N., Kim, Y., and Hwang, H. "An optimal operating policy for the production system with rework", *Computers & Industrial Engineering*, **56**, pp. 874-887 (2009).
7. Gribkovskaia, I.V., Kovalev, S., and Werner, F. "Batching for work and rework processes on dedicated facilities to minimize the makespan", *Omega*, **38**, pp. 522-527 (2010).
8. Kang, Y.H., Kim, S.S., and Shin, H.J. "A dispatching algorithm for parallel machines with rework processes", *Journal of the Operational Research Society*, **61**, pp. 144-155 (2010).
9. Ramezanian, R. and Saidi-Mehrabad, M. "Multi-product unrelated parallel machines scheduling problem with rework processes", *Scientia Iranica E*, **19**(6), pp. 1887-1893 (2012).
10. Taleizadeh, A.A., Jalali-Naini, S.G., Wee, H.M., and Kuo, T.C. "An imperfect multi-product production system with rework", *Scientia Iranica E*, **20**(3), pp. 811-823 (2013).
11. Sarkar, B., Cárdenas-Barrón, L.E., Sarkar, M., and Singgih, M.L. "An economic production quantity model with random defective rate, rework process and backorders for a single stage production system", *Journal of Manufacturing Systems*, **33**(3), pp. 423-435 (2014).
12. Hossain, M.S.J. and Sarker, B.R. "Optimal locations of on-line and off-line rework stations in a serial production system", *International Journal of Production Research*, **54**(12), pp. 3603-3621 (2016).
13. Beynaghi, A., Moztarzadeh, F., Shahmardan, A., Alizadeh, R., Salimi, J., and Mozafari, M. "Makespan minimization for batching work and rework process on a single facility with an aging effect: a hybrid meta-heuristic algorithm for sustainable production management", *Journal of Intelligent Manufacturing*, (2016). DOI: 10.1007/s10845-016-1223-0.
14. Moussawi-Haidar, L., Salameh, M., and Nasr, W. "Production lot sizing with quality screening and rework", *Applied Mathematical Modelling*, **40**, pp. 3242-3256 (2016).
15. Baker, K.R. and Trietsch, D., *Principles of Sequencing and Scheduling*, New York, Wiley (2009).

16. Rinnooy Kan A.H.G., *Machine Scheduling Problems: Classification, Complexity and Computations*, The Hague, Netherlands, Martinus Nijhoff Publishers (1976).
17. Palmer, D.S. "Sequencing jobs through a multistage process in the minimum total time: a quick method of obtaining a near optimum", *Journal of the Operational Research Society*, **16**, pp. 101-107 (1965).
18. Campbell, H.G., Dudek, R.A., and Smith, M.L. "A heuristic algorithm of the n-job, m-machine sequencing problem", *Management Science*, **16**(10), pp. 630-637 (1970).
19. Gupta, J.N.D. "A functional heuristic algorithm for the flow-shop scheduling problem", *Operational Research Quarterly*, **22**(1), pp. 39-47 (1971).
20. Nawaz, M., Enscore, J.E., and Ham, I. "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem", *Omega*, **11**(1), pp. 91-95 (1983).
21. Li, X. and Yin, M. "A discrete artificial bee colony algorithm with composite mutation strategies for permutation flow shop scheduling problem", *Scientia Iranica E*, **19**(6), pp. 1921-1935 (2012).
22. Johnson, S.M. "Optimal two and three-stage production schedules with set-up times included", *Naval Research Logistics Quarterly*, **1**, pp. 61-68 (1954).
23. Holland, J.H., *Adaptation in Natural and Artificial Systems*, Cambridge, MIT Press (1975).
24. Goldberg, D.E., *Genetic Algorithms: Search, Optimization and Machine Learning*, Boston, Addison-Wesley Press (1989).
25. Mahdavi, I., Paydar, M.M., Solimanpur, M., and Heidarzade, A. "Genetic algorithm approach for solving a cell formation problem in cellular manufacturing", *Expert Systems with Applications*, **36**(3), pp. 6598-6604 (2009).
26. Paydar, M.M. and Saidi-Mehrabad, M. "A hybrid genetic-variable neighborhood search algorithm for the cell formation problem based on grouping efficacy", *Computers & Operations Research*, **40**(4), pp. 980-990 (2013).
27. Bootaki, B., Mahdavi, I., and Paydar, M.M. "A hybrid GA-AUGMECON method to solve a cubic cell formation problem considering different worker skills", *Computers & Industrial Engineering*, **75**, pp. 31-40 (2014).
28. Kirkpatrick, S., Gelatt, C., and Vecchi, M. "Optimization by simulated annealing", *Science*, **220**(4598), pp. 671-680 (1983).
29. Mladenovic, N. and Hansen, P. "Variable neighborhood search", *Computers & Operations Research*, **24**, pp. 1097-1100 (1997).

Biographies

Behrang Bootaki received his BSc degree in Industrial Engineering from University of Bojnord in 2011. Also, he received his MSc degree in Industrial Engineering from Mazandaran University of Science & Technology in 2013. He has published some papers in well-known national and international journals. His research interests include combinatorial optimization problems, mathematical modeling, multi-objective optimization techniques, and metaheuristics.

Mohammad Mahdi Paydar is an Assistant Professor in Industrial Engineering at Babol Noshirvani University of Technology. He received his PhD degree in Industrial Engineering from Iran University of Science and Technology. His research interests are cellular manufacturing systems and modelling of manufacturing applications. He has published articles in journals, such as Computers and Industrial Engineering, Computers and Operations Research, Expert Systems with Applications, International Journal of Advanced Manufacturing Technology, Journal of Manufacturing Systems, International Journal of Production Research, and 20 papers in international conferences.