Applied Mathematical Modelling 38 (2014) 2490–2504

Contents lists available at ScienceDirect

Applied Mathematical Modelling

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

Hybrid flow shop scheduling with sequence dependent family setup time and uncertain due dates

M. Ebrahimi, S.M.T. Fatemi Ghomi *, B. Karimi

Department of Industrial Engineering, Amirkabir University of Technology, 424 Hafez Avenue, 15916-34311, Tehran, Iran

ARTICLE INFO

Article history:
Received 18 August 2012
Received in revised form 12 September 2013
Accepted 11 October 2013
Available online 15 November 2013

Keywords:
Hybrid flow shop scheduling
Multi-objective optimization
Sequence-dependent setup time
Stochastic modeling
Metaheuristic algorithms

ABSTRACT

This paper studies the scheduling problem in hybrid flow shop (HFS) environment. The sequence dependent family setup time (SDFST) is concerned with minimization of makespan and total tardiness. Production environments in real world include innumerable cases of uncertainty and stochasticity of events and a suitable scheduling model should consider them. Hence, in this paper, due date is assumed to be uncertain and its data follow a normal distribution. Since the proposed problem is *NP-hard*, two metaheuristic algorithms are presented based on genetic algorithm, namely: Non-dominated Sorting Genetic Algorithm (NSGAII) and Multi Objective Genetic Algorithm (MOGA). The quantitative and qualitative results of these two algorithms have been compared in different dimensions with multi phase genetic algorithm (MPGA) used in literature review. Experimental results indicate that the NSGAII performs very well when compared against MOGA and MPGA in a considerably shorter time.

1. Introduction

Scheduling is one of the important problems in production planning systems. In general form, scheduling problems are categorized as difficult problems and also flow shop scheduling problems include a wide range of production and assembly models. One of the scheduling environments that have a very good adaptability with most of the real world problems is hybrid flow shop (HFS). HFS is considered as one of the difficult problems due to its complexity [1]. According to Linn and Zhang [2] definition, scheduling problem of HFS includes a series of production stages that each individual one of these stages includes machine or parallel machines. And at least there is one stage which includes more than one machine. Parallel machines in each stage can be divided into three groups: identical parallel machines, uniform parallel machines, and unrelated parallel machines. In this paper, scheduling problem of HFS with identical parallel machines has been studied. All of machines located in one stage are quite identical and the processing time of each job in that stage does not depend on machine being used.

In most scheduling problems, setup time is considered very unimportant and insignificant or it is considered as a part of the processing time, this will result in simplification of analysis scheduling, but in most of real problems, this assumption is unreal and machine setup is unavoidable when changing job on the machinery. That is also introduced by Ulungn et al. [3]. They divided the setup time into two categories: sequence-independent: setup time is only dependent on the job that is being processed, therefore it is called sequence-independent. Sequence-dependent: setup time depends on the job to be processed and the one yet processed, therefore it is called sequence-dependent. One of the last studies that considered sequence-dependent scheduling, has been the study done by Behnamian and Fatemi Ghomi [4]. The most difficult

* Corresponding author. Tel.: +98 21 64545381; fax: +98 21 66954569.
E-mail address: Fatemi@aut.ac.ir (S.M.T. Fatemi Ghomi).

0307-904X/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.apm.2013.10.061

condition of setup times occurs when setup times are dependen sequence. In this condition, setup time is dependent on the current job that is being processed, and also the previous job that has been processed on that machine. In most cases these setup times are unsymmetrical. It means that the setup time from job i to j is different compared to the setup time from job j to i. Especially when the job processing is based on group technology (GT), there are sequence-dependent setup times between groups of jobs. In this paper the above mentioned condition is considered, which means sequence dependent setup times with respect to processing jobs is considered on a GT basis. GT is a management theory that its purpose is to group parts according to similarity of the production process or production characteristics (exterior shape of parts) or both of them. A lot of researchers have used GT in cell production system for scheduling jobs. In a cell production system, parts are grouped according to the similarity of production process and also the kind of machines that exist in a cell [5]. For all of parts included in one group, one setup time is considered. It means that setup time for the jobs inside one group is not considered, but it is calculated for each group. By doing this work, the total setup time decreases and system performance enhances [6].

In industrial world, nothing is certain and parameters that affect the production of a product are stochastic. In traditional methods, uncertain variables and constraints are not considered [7]. This paper studies due date as stochastic and hence stochastic modeling is applied in formulation of our problem.

Most of the real world problems include simultaneous optimization of different objectives that have conflict with each others. The final solutions show a compromise between different objectives according to the decision maker opinion. In this paper, two objectives have been considered as follows:

$$f_1 = \text{makespan}: \quad C_{max} = \max\{C_i\} \quad i = 1, 2, \ldots, n, \tag{1}$$

$$f_2 = \text{total weighted tardiness} = TWT = \sum_{i=1}^{n} W_i T_i = \sum_{i=1}^{n} W_i \max(0, C_i - d_i) \quad i = 1, 2, \ldots, n, \tag{2}$$

where n is the number of jobs.

Gupta and Darrow [8] expressed that even the (HFS) problem with sequence dependent setup time (SDST) and two machines is *NP-hard*. So, in this paper, two metaheuristic algorithms based on genetic algorithm (GA) have been used to solve the problem. These algorithms are called: Non-dominated Sorting Genetic Alghorithm (NSGAII) and Multi Objective Genetic Algorithm (MOGA). To evaluate performance algorithms, their results are compared with those of Multi Phase Genetic Algorithm (MPGA) used by Karimi et al. [9].

Our goal in this paper is to apply stochastic modeling to solve HFS problem with uncertain due date and sequence dependent family setup time (SDFST). The makespan and total weighted tardiness are to be optimized simultaneously. In order to obtain pareto optimal solutions, we apply two metaheuristic algorithms, NSGAII and MOGA. The paper is organized as follows: Section 2 gives the brief literature review of HFS problems. Section 3 is the problem description, mathematical model, and stochastic modeling. Section 4 gives a description of two metaheuristic algorithms based on genetic algorithm. Section 5 presents numerical results. Section 6 is devoted to conclusions and further research directions.

2. Literature review

2.1. Uncertain due date

Production environments in the real world include innumerable cases of uncertainty and stochasticity of events. Some cases are as follows: machinery breakdown, change in due date, uncertainty of process time of a job on a machine. A suitable scheduling model should consider all of the uncertainty conditions, in order to approach problems of the real world. Linn and Zhang [2] expressed that solving HFS problem that is an *NP-hard* problem, is usually difficult. The first study in the field of stochastic scheduling problems was done by Cheng [10] in single machine environment. He developed his article in 1991 and in this research, a common due date was defined for all jobs and each job has a stochastic process time with specific mean and variance. Elyasi and Salmasi [11] studied flow shop scheduling problem minimizing the expected number of tardy jobs. The authors assumed that the jobs with deterministic processing times and stochastic due dates arrive randomly to the flow shop cell. The due date of each job was assumed to be normally distributed with known mean and variance. A dynamic method was proposed for this problem by which the m machine stochastic flow shop problem was decomposed into m stochastic single machine sub-problems. Then, each sub-problem was solved as an independent stochastic single machine scheduling problem by a mathematical programming model. Kamburowski [12] examined the scheduling problem for flow shop with two machines. He considered the processing time as stochastic and assigned exponential distribution to the processing time. The objective function for this problem was minimization of the expected makespan. He extended problem with three machines [13]. The number of the researches studied the HFS together with stochastic assumptions is very limited. One of the researches in this field was studied by Rajendran and Holtheaus [14]. They studied dispatching rule in the flow shop and job shop environments while considering stochastic processing time and arrival time of jobs. Tang et al. [15] also proposed a similar problem. They considered flexible flow shop assuming stochastic arrival time of jobs. In the continuation of their work they used neural network method to select dispatching rule. Gourgand et al. [16] proposed stochastic scheduling problems in m machine flow shop with unlimited buffers. The processing time of each job on each machine is a random variable following exponentially distributed with a known rate. In order to solve the performance evaluation

problem, they proposed a recursive algorithm based on a Markov chain to compute the expected makespan and a discrete event simulation model to evaluate the expected makespan. Baker and Altheimer [17] investigated the stochastic flow shop problem with m machines and general distributions for processing times. They used heuristic methods to solve this problem. Almeder and Hartl [18] studied flexible flow shop stochastic scheduling problem with limited buffers. In the first step, they analyze a simplified model and develop a variable neighborhood search (VNS). In the second step, the solution approach was adapted to a real-world case using a detailed discrete-event simulation to evaluate the production plans. Wang et al. [19] presented a class of hypothesis test based on the genetic algorithm for the problem of flow shop scheduling with stochastic process time. Aydilek and Allahverdi [20] studied flow shop with two machines and stochastic process time. The authors considered the stochastic process time bounded and in a certain interval. Ahmadizar et al. [21] studied the problem of group shop stochastic scheduling. Group shop scheduling problem is a combination of scheduling in job shop, open shop and flow shop environments. They assumed the release time and process time of jobs as stochastic and with a specific distribution function. They used a metaheuristic algorithm and a heuristic algorithm to find a job schedule which minimizes the expected makespan. Ghezavati and Said-Mehrabad [5] assumed the arrival and process time of parts in cellular manufacturing system as uncertainty and used queuing theory, robust optimization and stochastic programming to formulate and analyze their problem. They used a hybrid method based on genetic algorithm (GA) and simulated annealing to solve their stochastic and nonlinear problem. Kianfar et al. [22] presented flexible flow shop scheduling considering sequence-dependent setup time and the arrival time of jobs as uncertain and dynamic. The objective function of their problem was minimization of average jobs delay. They used hybrid genetic algorithm and novel dispatching rule method to solve their problems.

2.2. Stochastic modeling

Researchers have used a variety of methods for stochastic modeling. Tavakolli-Moghadam et al. [7] studied job shop scheduling problem for stochastic process time with normal distribution. They used the expected value method to model the problem. This method has been defined by adding an uncertainty constraint to the model. They presented a hybrid method to solve their problem. In this method, initial feasible solution was made using neural networks (NN). Then the initial solution was enhanced using simulated annealing (SA) metaheuristic algorithm. Jiuping and Zhou [23] and Gu et al. [24] also used this method for stochastic modeling of their problem. Elyasi and Salmasi [25] considered two different due date assignment and sequencing problems in single machine where the processing times of jobs were random variables. The authors consider two problems. The first problem was to minimize the maximum due date so that all jobs were stochastically on time. The results were then extended for two special cases of flow shop problem. Elyasi and Salmasi [26] studied two scheduling problems, single machine scheduling problem minimizing the number of tardy jobs and two machine flow shop scheduling problem with a common due date and minimizing the number of tardy jobs. They assumed that the processing times of jobs are independent random variables. The stochastic problems were solved based on chance constrained programming. Jang and Kelin [27] studied scheduling problem of single machine with the objective of minimizing the expected number of tardy jobs. The process time of jobs is uncertainty and follows normal distribution. They used chance constraint method to model their problem and then used a heuristic method to solve the problem. Ginzburg and Laslo [28] studied the scheduling problem in a flexible shop environment considering dispatching rule. They used chance constraint method to model their problem assuming the uncertain process time. Seo et al. [29] and Soroush [30] also used the chance constraint to model their problem. The method that we have used in this paper for stochastic modeling is similar to the one used by Dong et al. [29] for stochastic scheduling of single machine.

2.3. Hybrid flow shop scheduling with sequence dependent setup time

Proust et al. [31] studied the flow shop scheduling problem with two machines individually assuming setup time dependent on sequence and transfer time. Ashkin and Vakharia [32] expressed the role of GT in the cell production system for the first time. At first they specified family parts that should be associated with a cell. Furthermore they analyzed the parts that belong to a family and machines inside a cell. Li and Huang [33] studied the problem of $HF2\ (1^{(1)}, QM^{(2)})|fmls, S_{ij}|\ C_{max}$.The authors proposed the HFS with two stages in a GT environment. To solve the problem, they developed two heuristic algorithms and eight sequence rules for assigning parts to machines in each of two stages. The Similar problem of group scheduling with sequence-dependent setup times and minimizing the total completion time was studied by Logendran et al. [34] and Logendran et al. [35]. Schaller et al. [36] considered the problem of scheduling part families and jobs within each part family in a flow line manufacturing cell where the setup times for each family are sequence dependent and it is desired to minimize the makespan while processing jobs in each family together. Shahvari et al. [37] investigated the flexible flow shop sequence-dependent group scheduling problem with minimization of makespan. To efficiently solve the problem, they developed six metaheuristic algorithms based on tabu search (TS). A comparison shows that the best algorithm proposed in this research has better performance than the existing algorithm in the literature. Salmasi et al. [38] developed a mathematical programming model to minimize the total flow time of the flow shop sequence dependent group scheduling. They proposed a TS algorithm as well as a hybrid ant colony optimization (HACO) algorithm to solve the problem. A lower bounding (LB) method is also developed to evaluate the quality of the metaheuristic algorithms. They studied a similar problem with minimization of makespan as the objective in [39]. Hajinejad et al. [40] proposed a Particle Swarm Optimization (PSO) algorithm for a Flow Shop Sequence Dependent Group Scheduling (FSDGS) problem, with minimization of total flow

time as the criterion. The performance of the algorithm is compared with the best available metaheuristic algorithm in literature. The results show that the proposed algorithm has better performance. Salmasi and Logendran [6] presented the several heuristic algorithms based on TS to solve the multi-stage sequence-dependent group scheduling (SDGS) problem. Also a lower bounding technique based on relaxing the mathematical model for the original SDGS problem is applied to estimate the quality of the heuristic algorithms. The results of the experiment show that the TS can provide good quality solutions for the problems. Hendizadeh et al. [8] presented the various TS's for scheduling the part families and jobs within each family in a flow line manufacturing cell with sequence dependent family setup times to minimize makespan. The effectiveness and efficiency of the proposed TS were compared against the best meta-heuristic and heuristic algorithms reported so far for this problem on several test problems. Logendran et al. [9] studied the group scheduling in flexible flow shops. The authors developed a statistical model based on split-plot design to conduct the experiments to compare the performance of three different heuristic solution algorithms. Logendran et al. [10] considered group scheduling within the context of sequence dependent setup times in flexible flow shops. Researcher developed three different algorithms based on TS. Tavakolli-Moghadam et al. [6] used the philosophy of GT to schedule manufacturing cell systems. They used a metaheuristic algorithm based on the scatter search to solve the multi-objective scheduling problem. Dvir Shabtay et al. [41] studied a single machine scheduling problem in which the scheduler determines due dates for different jobs in a GT environment. In this field, using GT concept to schedule manufacturing cell systems, a number of studies have been done by researchers. As two of the last researches done in this field, are Karimi et al. [9] and Ghezavati and Mehrabad [5] studies.

2.4. Multi-objective metaheuristic

Different metaheuristic algorithms have been proposed to optimize problem having two or several objectives. Vector evaluated genetic algorithm (VEGA) was the first method based on GA proposed by Schaffer [42] to find the pareto solutions in optimized multi-objective problems. Murata et al. [43] proposed Multi-Objective Genetic Algorithm (MOGA) for the first time. This algorithm uses stochastic weights assigned to objectives and transforms multi-objective problems to a single objective problem. Deb et al. [44] proposed NSGAII with elitism strategy. In this algorithm to have the variety in solution, crowding distance has been used to measure population density. A variety of methods to solve multi-objective has been expressed by researchers. VEGA [42], MOGA [45], Niched Pareto Genetic Algorithm (NPGA) [46], Non-dominated Sorting Genetic Algorithm (NSGA and NSGAII) [47], Pareto Stratum-Niche Cubicle Genetic Algorithm (PS-NC GA) [48], Multi-Ple Objective Genetic Local Search (MOGLS) [49], Strength Pareto Evolutionary Algorithm (SPEA and SPEAII) [50], Multi-Objective Scatter Search (MOSS) [51]. In this paper NSGAII and MOGA algorithms have been used to find the pareto solution set.

3. Problem definition

This section first defines the problem. The same deterministic model proposed by Karimi et al. [9] is used. Then assumptions and stochastic modeling are introduced. The approach transforms the stochastic optimization problem to equivalent non-linear deterministic problems, which are then solved using appropriate deterministic optimization techniques.

3.1. Notations

Following defines the notations used to develop the stochastic model.

n: number of jobs to be scheduled
g: number of serial stages
n_G: number of groups
g_i: last stage visited by job i, $i \in \{1,2,\ldots,n\}$
m^t: the number of machines in parallel at stage t, $t \in \{1,2,\ldots,g\}$
P_j^t: processing time for all jobs in group j at stage t, $j \in \{1,2,\ldots,n\}$
P_{ij}^t:processing time for job i in group j at stage t (assumed to be integral)
d_i: due date of job i
S_{pj}^t:setup time for group j on stage t, if group j is processed immediately after group p j, $p \in \{1,2,\ldots,n_G\}$
C_i^t:completion time of job i at stage t
CG_i^t:completion time for group i at stage t

3.2. Assumptions

The problem studied in this paper considers the following assumptions:

- All jobs are available at the beginning of planning horizon.
- Due dates of jobs in each group follow a normal distribution with the specified parameters (a normal distribution is considered for each group).

- Each stage has at least one machine, and at least one stage must have more than one machine. According to these, different groups may be processed simultaneously in one stage.
- Setup time for each group is considered not for jobs within group. This setup time is sequence dependent.
- Job processing cannot be interrupted, until the completion of job.
- Machines are always available.
- Buffer capacity between stages, before the first and after the last stages, is infinite.
- Jobs are available for processing at a stage immediately after processing completion at the previous stage.
- Each machine can only process one task at a time.
- A job cannot be processed simultaneously on more than one machine.
- Parallel machines at each stage in terms of capacity and processing speed, are identical.
- There is no travel time between stages.
- A job might skip any number of stages but each of groups should pass through all the stages and cannot skip any stage.

3.3. Multi-objective optimization

Many of the real world problems consider simultaneous optimization of different objectives. These objectives are usually in competition and conflict with each other. Multi-objective optimization provides a solution set using the confliction between objectives that is called pareto-optimal. This solution set includes solutions that no other better solution than them could be found when all objectives are considered. A minimization problem having P as decision variable and q as objective ($q > 1$) is presented here.

$$\text{Min } y = f_{(X)} = (f_{1(X)} f_{2(X)}, \ldots, f_{q(X)}), \tag{3}$$

where $X \in R^p$

Solution a is said to dominate solution b if and only if:

$$(1)\ f_{i(a)} \leq f_{i(b)} \quad \forall\, i \in \{1,2,\ldots,q\}$$
$$(2)\ f_{i(a)} \prec f_{i(b)} \quad \exists\, i \in \{1,2,\ldots,p\}.$$

Solutions that dominate others but do not dominate themselves are called non-dominated solutions [7]. According to their definition, vector a is a globally pareto-optimal solution if vector b does not exist such that b dominates a. Corresponding image of pareto-optimal set in the objective space is called the pareto-optimal frontier.

3.4. Stochastic modeling

This paper assumes that job's due date is stochastic and follows a normal distribution. Due date of every job is dependent on the job process time in all of stages. Because the first job of every group needs setup time and this time is dependent on the sequence, due date of every job is dependent on the process time and setup time in all stages. In this section, at first we obtain the average and variance of a job due date of each group and other jobs of the group will follow this normal distribution. Indeed, a normal distribution with the specified parameters is assumed for each group. Then we start stochastic modeling of our problem. The following steps are intended to calculate the average and variance of due date.

(1) The total processing time is calculated separately for a job of each group:

$$P_{ri} = \sum_{t=1}^{g} P_{ij}^{t}, \quad i = 1,2,\ldots,n,\ j = 1,2,\ldots,n_G,\ t = 1,2,\ldots,g, \tag{4}$$

where P_{ri} is total processing time for a job of each group in all stages.

(2) We calculate the average setup times for all possible scenarios of sequences of groups in every stage and then add them up.

$$T_{sj} = \sum_{t=1}^{g} \frac{\sum_{p=1,p\neq j}^{n_G} S_{pj}^{t}}{n_G - 1}, \quad j = 1,2,\ldots,n_G, \tag{5}$$

where T_{sj} is the average of setup times for group j. In this paper, setup time for the first job in every group is considered equal to group setup time to which desired job belongs.

S_i : Setup time for job i $\quad S_i = T_{sj} \quad \forall\, i \in j$th group.

(3) The mean due date for a job of each group is calculated by Eq. (6):

$$\mu_{d_i} = \frac{(1 + 2 \times random) g (S_i + P_{ri})}{(n_G \times \min(m^t))}, \quad i = 1,2,\ldots,n, \tag{6}$$

random is random number in (0,1) interval. m^t is a vector which stores the number of parallel machines at each stage.

(4) Also due date variance for a job of each group is given by Eq. (7):

$$\sigma_{d_i}^2 = (g + n_G - 1)(P_{ri} + S_{ti}) \times 0.4. \tag{7}$$

Thereupon, due date of a job of each group has a normal distribution with mean and variance of:

$$d_i \sim N\left(\frac{(1 + 2 \times random)g(S_i + P_{ri})}{(n_G \times \min(m^t))}, (g + n_G - 1)(P_{ri} + S_{ti}) \times 0.4\right). \tag{8}$$

Minimizing makespan is the first objective that we consider for this problem defined as:

$$C_{max} = \max\{C_i\},$$

where C_i is the completion time of job i.

The second objective in this paper is the total tardiness that is equal to $\sum_{i=1}^{n} W_i T_i$. A job will be tardy when its due date is shorter than its completion time in the last station. Because in this paper due date is also assumed stochastic, we have the following equation for the second objective function.

$$\min f_2 = \sum_{i=1}^{n} W_i T_i = \sum_{i=1}^{n} W_i P_r(d_i \prec C_i^{g_i}). \tag{9}$$

Since due date has a normal distribution, the above expression can be rewritten as:

$$\min f_2 = \min \sum_{i=1}^{n} W_i \varphi\left(\frac{(C_i^{g_i} - \mu_i)}{\sigma_i}\right). \tag{10}$$

Cumulative distribution function for standard normal random variable is $\varphi(\alpha)$. Our approximation of objective function f_2 based on assumptions is to minimize the $\varphi(\alpha)$. Hence, we want to minimize α.

$$\min f_2 = \min \sum_{i=1}^{n} W_i \left(\frac{(C_i^{g_i} - \mu_i)}{\sigma_i}\right). \tag{11}$$

If we want to minimize the total earliness, the above statement will change as Eqs. (12) and (13):

$$\min f = \sum_{i=1}^{n} W_i P_r(d_i \succ C_i^{g_i}), \tag{12}$$

$$\min f = \min \sum_{i=1}^{n} W_i \left(1 - \varphi\left(\frac{(C_i^{g_i} - \mu_i)}{\sigma_i}\right)\right), \tag{13}$$

where $\varphi(\alpha)$ is the cumulative distribution function for the standard normal random variable. Our approximation of objective function f is based on minimizing $(1 - \varphi(\alpha))$ which needs the α to be maximized. The above statement will change as Eq. (14):

$$\min f = \max \sum_{i=1}^{n} W_i \left(\frac{(C_i^{g_i} - \mu_i)}{\sigma_i}\right). \tag{14}$$

For jobs inside a group processed at first, the mean due date may be near zero according to their process time and setup time. Because the due date values are selected from normal distribution in a stochastic manner, there is a probability that due date be taken negative. To avoid this, we can apply the following conditions:

The due date mean is selected from the uniform distribution $[a, b]$ with condition of $a, b > 0$. But still there is a possibility that selected d_i is negative. So the variances are also uniformly distributed yet with the restriction that 99% of processing times are required to be positive. That is, for a given μ_i,σ_i is randomly selected from the uniform distribution $\left[0, \frac{\mu_i}{2.33}\right]$. Therefore the probability of choosing a negative value for due date is:

$$P_r(d_i \prec 0) = \varphi\left(-\frac{\mu_i}{\sigma_i}\right) \prec \varphi(-2.33) = 0.01. \tag{15}$$

The due date with 99% probability is always positive.

4. Proposed algorithms for the model

This section develops two metaheuristic algorithms (NSGAII and MOGA) based on GA for our problem. First, essential concepts about the solution representation method and details of crossover and mutation operators are reviewed. Then each metaheuristic algorithms used in this paper is explained.

4.1. Solution representation

In the metaheuristic algorithms the first step to solve a problem is usually to specify the way of solution representation. Bean [52] proposed a new method to represent solutions in GA named random keys. In this method a random number is assigned to each gene of a chromosome. In this form, the integer part of each gene represents the machine that a job should be assigned to it and the fractional part determines the sequence of jobs on any machine in such a way that jobs with smaller decimal part will be processed sooner. In this paper for coding every chromosome we have used a matrix method to show any element of population in a way that every matrix row is representative of a group of that chromosome. To assign groups to machines in the first stage, the applied method is inspired by a method proposed by Norman and Bean [53]. The sequence of groups and jobs in each group is determined as follows:

For each group a random number between $[1,m^1+1]$ is generated that $m^1$is the number of machines in the first stage. The integer part of the generated number represents the number of machines that group should be processed on and the fractional part of this number is to determine the sequence of jobs inside each group for that specific machine. Also in order to determine the sequence of jobs in each group, random numbers are generated between [0,1]. The job having smaller number is processed sooner and in this way the sequence of jobs in each group is also determined. In the next stages, assignment of groups and jobs to machines is done based on their completion time in the previous stage. As an example, it is assumed that there are five groups of jobs and the number of jobs in these groups is 3, 2, 4, 3 and 3 respectively. And also there are three machines in the first stage. Every row in the matrix (shown in Fig. 1) is representative of a group. The first column is used to assign the groups to the machines and also to determine the sequence of groups. The second column and the others are each one representative of one job associated with every group. An example of the proposed chromosome is depicted in the Fig. 1.

The sequence of tasks and groups on the first machine will be as follows:

Machine 1: $G_3(J_3,J_1,J_4,J_2)$,

Machine 2: $G_5(J_2,J_1J_3)$, $G_1(J3,J_2,J_1)$,

Machine 3: $G_4(J_3,J_2J_1)$, $G_2(J_2,J_1)$.

4.2. Crossover and mutation operators

Following describes the crossover and mutation operators employed for NSGAII and MOGA algorithms.

4.2.1. Crossover

Crossover is a mechanism for diversification. The crossover creates two new children by combining both parent chromosomes' genes. For this study we perform uniform crossover. In order to do that, two chromosomes in the current generation are selected. We consider the chromosome with better fitness value as parent 1 and the other one as parent 2. Then a new chromosome with random number between (0, 1) should be generated and named as mask. The mask size should be equal to the maximum length of groups. If the generated number for one specific gene in mask is more than 0.7, the corresponding value of parent 1 is copied into it, otherwise the corresponding value of parent 2 is used (see Figs. 2 and 3).

4.2.2. Mutation

The remaining percent of the generation is produced using mutation operator. At first two groups of one chromosome is selected. The associated values with these two groups are replaced and then we determine two jobs from each group and exchange the associated values of these two jobs (see Fig. 6).

4.3. Non-dominated Sorting Genetic Algorithm (NSGAII)

Sirnivas and Deb [54] proposed NSGA algorithm. Their algorithm had disadvantages as follows: high complexity of calculations, the lack of elitism and need to determine a parameter named sharing parameter. Deb et al. [44] modified the NSGA procedure and proposed NSGAII. This new modified evolutionary algorithm eliminated the disadvantages of the previous algorithm to a great extent. The details of the used NSGAII algorithm in this paper are as follows:

Step 1. Representing solution: In this paper the way of representing solutions for desired problems is based on random numbers method explained in Section 4.1.

Step 2. Generate initial population: We create a random population P_0 with N chromosomes.

	Machine	J_1	J_2	J_3	J_4
G_1	2.84	0.41	0.22	0.18	-
G_2	3.62	0.66	0.45	-	-
G_3	1.75	0.27	0.74	0.11	0.5
G_4	3.31	0.81	0.47	0.33	-
G_5	2.55	0.73	0.26	0.98	-

Fig. 1. Solution representation.

	Machine	J_1	J_2	J_3	J_4
G_1	2.84	0.41	0.22	0.18	-
G_2	3.62	0.66	0.45	-	-
G_3	1.75	0.27	0.74	0.11	0.5
G_4	3.31	0.81	0.47	0.33	-
G_5	2.55	0.73	0.26	0.98	-

Parent 1

	Machine	J_1	J_2	J_3	J_4
G_1	1.45	0.12	0.12	0.23	-
G_2	2.32	0.32	0.57	-	-
G_3	3.15	0.47	0.67	0.22	0.74
G_4	1.98	0.96	0.32	0.74	-
G_5	3.55	0.27	0.75	0.16	-

Parent 2

0.36	0.05	0.76	0.96	-
0.89	0.87	0.95	-	-
0.64	0.25	0.55	0.37	0.12
0.21	0.84	0.43	0.62	-
0.12	0.73	0.86	0.18	-

Mask

	Machine	J_1	J_2	J_3	J_4
G_1	2.84	0.41	0.12	0.23	-
G_2	2.32	0.32	0.57	-	-
G_3	1.75	0.27	0.74	0.11	0.5
G_4	3.31	0.96	0.47	0.33	-
G_5	2.55	0.27	0.75	0.98	-

Fig. 2. A crossover example.

Step 1: Let f_r =1(set front counter equals to 1)

Step 2: Calculate makespan and total tardiness of jobs for each $x \in POP$.

Step 3: For each solution x belonging to POP

3-1: Find n_x (the number of solutions which solution x is dominated by them)

3-2: Find S_x (Set of all solutions which are dominated by solution x)

3-3: If solution x is not dominated by any other solution ($n_x = 0$), it belongs to first non-dominated front 1.

Step4: Eliminate solution belonging to front f_r from POP.

Step5: Set $f_r = f_r + 1$, start identifying solutions of next front:

Step6: For each solution x belonging to last determined front (front (f_r-1)).

6-1: for each y belonging to S_y , set $n_y = n_y - 1$ and if $n_y = 0$, y belongs to f_rth front.

Step7: If all solution are assigned to fronts go Step 8, otherwise go to Step4

Step8: Termination of algorithm.

Fig. 3. Pseudo-code of fast non-dominated sorting.

Step1: Set $K=1$ (K is objective counter)

Step2: Calculate Kth objective function for each solution.

Step3: Sort S in an ascending order according to the value of Kth objective function.

Step4: For the first and last solutions of sorted set, distance values are equal to infinit, and for the others, use the Normalized difference in the function values of two adjacent solutions as crowding-distance.

Step5: If K equals to 2(the number of objectives in this study)go to Step 6, otherwise set $K=k+1$ and go to Step2

Step6: The value of crowding-distance for each solution is the summation of crowding-distance of that specific Solution for different objectives.

Fig. 4. Pseudo-code of crowding distance.

Step1: Encode solution by employing random numbers.

Step2: Create random population P of N chromosomes.

Step3: Rank solutions of P by the help of "fast *non-dominated sorting*" and considering minimization of two objectives.

Step4: Set fitness value of each solution equal to its nondomination level.

Step5: Create offspring population Q of N chromosomes.

5-1: use binary tournament selection.

5-2: do the crossover based on random numbers with the probability of r_c

5-3: do the mutation based on random numbers with the probability of r_m.

Step6: Make new population R by combining parent population P and offspring population Q. ($R=P \cup Q$)

Step7: If the stopping criteria is met go to Step 10.

Step8: By the use of new population R, create population P with N chromosomes based on crowded comparison operator.

Step9: Go to Step3.

Step10: Rank solutions of population R by applying "*fast non-dominated sorting algorithm*" and set the first non-domination level as the final Pareto-set.

Fig. 5. Pseudo-code of NSGAII.

Step 3. Rank solution: All solutions are assigned to different non-dominated levels using fast non-dominated sorting algorithm. This algorithm is as follows:

Step 4. Evaluation: For each solution, a fitness function equal to non-dominated front of that solution is assigned. For example, in a minimization problem, the fitness value for solutions located in level 1 is set equal to 1. For the second level solutions, the fitness value is set equal to two and this procedure goes on.

Step 5. Create population Q_0: Using binary tournament selection strategy, crossover and mutation operators, a population of solutions named Q_0 and having N chromosomes are created. Crossover and mutation operators are used as explained in Sections 4.2.1 and 4.2.2.

Step 6. Combination population P_0 and population Q_0: A new population named R_0 is produced from combination of parent populations P_0 and Q_0 that has $2N$ chromosomes. Combination of parent and offspring is repeated for all generations.

Step1: Encode solutions by employing random numbers.

Step2: Initialize a random population of N chromosomes.

Step3: Calculate the value of makespan and total tardiness and normalize their values for each solution.

Step4: Evaluate fitness function for each solution.

Step5: Calculate pareto-optimal solutions of this iteration and update the pareto-archive.

Step6: Apply reproduction using elitist strategy.

Step7: Apply roulette wheel selection strategy.

Step8: Apply crossover operator based on random numbers with the probability of r_c

Step9: Apply mutation operator based on random numbers with the probability of r_m

Step10: If the stopping criteria is met, it is the termination of algorithm, otherwise and go to Step3.

Fig. 6. Pseudo-code of MOGA.

Step 7. Ranking solutions of population R_t: Suppose that population R_t is the result of combination of populations P_t and Q_t. We rank all solutions of this new population and assign all solutions to non-dominated levels.

Step 8. Create population P_{t+1}: Population P_{t+1} is created for next repetition. To create population P_{t+1}, we first use the first non-dominated level of population R_t, because solutions of this level are the best solutions of current generation. It should be noted that to create population P_{t+1}, N solutions are needed. Therefore if a number of solutions belonging to levels 1 is lesser than N, the solutions of level 2 is required to use. If the sum of solutions of levels 1 and 2 is lesser than N, the solutions of level 3 will be used and this procedure will go on. Suppose that level 'm' is the last non-dominated level used to create population P_{t+1}, in this condition the value of S equals to the sum of the solutions of levels 1,2,...,$m-1$. If $|level\acute{m}|$ is the number of solutions of level 'm' the following condition must be satisfied:

$$S + |level'm'| \prec N, \quad S = |level'1'| + \ldots + |level'm-1'|. \tag{16}$$

Crowding distance assignment and crowded-comparison operator to arrange the solutions of level 'm' in a decreasing manner are used. Then we use the best solutions of level 'm' to create chromosomes of population P_{t+1} equal to the number of $N-S$. Now the population of P_{t+1} that includes the best solutions of previous generation of chromosomes is produced. Crowding distance algorithm is illustrated as Fig. 4.

Step 9. Creating the population of Q_{t+1}: Once again, tournament double selection strategy and mutation and crossover operators on P_{t+1} are used to create Q_{t+1} population.

Step 10. Termination condition: The above process is continued until the stopping criteria are met. We have considered a specific number of generation repetitions as the stopping criterion. Fig. 5 shows the pseudo-code of NSGAII algorithm used in this paper.

4.4. Multi Objective Genetic Algorithm (MOGA)

Murata et al. [43] proposed multi-objective genetic algorithm for the first time to search for optimum pareto solutions for two and three objective scheduling problem. The steps of this algorithm are described here:

Step 1. *Solution representation*: According to what is stated in Section 4.1.

Step 2. Start: To start algorithm, a number of parameters like generation size, rate, mutation r_m, crossover rate r_c, should be set. Each generation includes a population of chromosomes that different numbers are assigned to chromosomes according to problem size (small, medium, and large). On the other hand, the number of generations repeated for a problem is different according to the problem size. The initial population in the first generation is produced in a stochastic manner.

Step 3. Evaluation: After production of the first generation, the created solutions in this generation are evaluated. This evaluation should be done based on the performance criteria. At first the value of each objective function is calculated for every solution. The objective functions may have different units, requiring the use of normalized values of objective functions that are expressed as Eqs. (17) and (18):

$$f_1(x) = \frac{C_{\max}(x) - \text{best}(C_{\max})}{\text{worst}(C_{\max}) - \text{best}(C_{\max})}, \tag{17}$$

$$f_2(x) = \frac{TT(x) - \text{best}(TT)}{\text{worst}(TT) - \text{best}(TT)}. \quad (18)$$

For solution x, $C_{max}(x)$, and $TT(x)$ are representative of numerical value of first and second objective functions. Worst (C_{max}) and best (C_{max}) are of the worst and best values of $C_{max}(x)$, respectively. Best (TT) and worst (TT) are representative of the best and worst values of objective function TT. $f_i(x)$ is the normalized value of the i objective function to x. Total weight of the objective function is used to evaluate. In this way we can evaluate solution of x using the following equation.

$$W_1 f_1(x) + W_2 f_2(x) = f(x). \quad (19)$$

Step 4. Reproduction strategy: In each generation the solutions that have the best fitness function value (total weight function) are stored and copied into the next generation.

Step 5. Selection: Selection is done based on the roulette selection strategy.

Step 6. Crossover and mutation: Crossover and mutation operators are done according to what was done in Section 4.2.

Step 7. Update the pareto-archive: A pareto-optimal archive is considered and in the end of each generation, the optimum solutions for that generation are calculated and the pareto archive gets updated.

Step 8. Stopping criteria: At the end of each generation the stopping criteria are examined to see if they are met. A specific number of repetitions as the stopping criteria is considered.

5. Experimental design

This section compares our proposed algorithms with each other. These algorithms have been coded with MATLAB R2008b.

5.1. Data generation

In this paper, three different sizes have been used for the problem: small, medium, and large. Therefore the size of problem is considered as one of the most important factors in three small, medium, and large levels. In this paper to generate example problems in order to evaluate the performance of the algorithms, Table 1 has been created.

In this paper, it is supposed that the number of jobs is equal in all groups. For example if the total number of works is 12 and we have 4 groups, then we will have 3 jobs in each group. The procedure of due date generation has been mentioned before.

5.2. Performance measure for multi-objective algorithms

In order to evaluate the performance of algorithms used in this paper, we need meter that provides a quantitative measurement to compare different sets of pareto solution. For this reason three metrics are considered, including:

5.2.1. Mean Ideal Distance (MID)

One of the methods of comparison between pareto solution sets is measuring their distance from the optimum solution sets. As it has been stated before, this set is not always accessible in multi-objective problems and therefore the main disadvantage of these methods is their dependence on the existence of the optimum solutions. In this paper a modified method resulting from this pattern has been proposed to overcome this disadvantage that does not need data of optimum solutions.

Table 1
Characteristics of test problem.

Factors	Level		
	Small	Medium	Large
Number of stage	3,4	5,6	7,8
Number of group	4,5	7,8	11,12
Number of job	12,15	49,64	110,144
Processing time	U(5,75)	U(5,75)	U(5,75)
	U(5,100)	U(5,200)	U(5,150)
	U(5,150)	U(5,300)	U(5,300)
	U(5,200)	U(5,400)	U(5,400)
	U(5,300)	U(5,450)	U(5,5000)
Setup time	U(5,25)	U(5,25)	U(5,25)
	U(5,50)	U(5,75)	U(5,100)
	U(5,75)	U(5,100)	U(5,150)
	U(5,100)	U(5,200)	U(5,200)
	U(5,150)	U(5,250)	U(5,250)
Maximum number of machines in each stage	2	3	4

Table 2
Parameter setting results.

Algorithm	Problem size	Population size	Number of iteration	Crossover rate	Mutation rate
MOGA	Small	100	400	0.8	0.1
	Medium	150	400	0.82	0.08
	Large	200	400	0.85	0.05
NSGAII	Small	100	400	0.8	0.2
	Medium	150	400	0.85	0.15
	Large	200	400	0.9	0.1

Table 3
Evaluation of non-dominated solutions of proposed algorithms.

Problem	Size	Stage	Group	MID			NPS			CPU time		
				NSGAII	MOGA	MPGA	NSGAII	MPGA	MOGA	NSGAII	MOGA	MPGA
1	*Small*	L	L	908.349	1204.61	1112.29	5	4	4	13.83	96.44	77.07
2		L	H	1250.04	1345.34	1268.27	8	3	5	14.26	112.56	98.49
3		H	L	1321.43	1361.72	1287.76	6	3	5	14.73	108.12	93.96
4		H	H	134.52	1445.33	1365.92	9	4	7	18.87	135.67	128.72
5	*Medium*	L	L	1452.34	1483.42	1596.55	11	5	6	224.5	958.59	1650
6		L	H	3245.64	4047.52	3805.63	13	7	5	258.29	1121	1752
7		H	L	3098.34	4912.44	5241.19	16	6	8	300.5	1089	1945
8		H	H	4019.33	7849.62	5986.51	15	8	10	483.72	1204	2132
9	*Large*	L	L	5468.53	8254.67	6350.48	17	12	15	1345	4100	3657
10		L	H	8405.38	12754	10384	19	13	17	1607	4655	4192
11		H	L	10532	11326	9214.94	23	11	18	1744	4521	4233
12		H	H	12612	15012	13215	18	14	20	2039	6977	4269
Average				4471.32	5916.39	4570.17	13.33	7.5	10	670.4	2089.86	2019.2

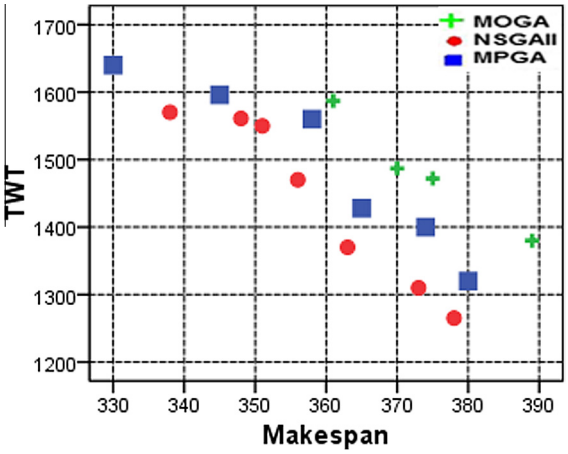

Fig. 7. A single run of the non-dominated solutions for small size problem.

For this purpose we have considered the point (0,0) as the ideal point and the distance between each pareto solution and this ideal point is being calculated. The average of these distances is called *MID*. Less *MID* indicates that non-dominated set we have is getting better.

$$MID = \frac{\sum_{i=1}^{n} C_i}{n} \qquad C_i = \sqrt{f_{1i}^2 + f_{2i}^2}, \tag{20}$$

n is the number of non-dominated set, C_i is distance between the ith non-dominated solution and ideal point. f_{1i}, f_{2i} are the values of ith non-dominated solution for first and second objective functions respectively.

5.2.2. Number of Pareto-Solutions (NPS)

This criterion is representative of the number of non-dominated solutions in pareto set resulted from the algorithm.

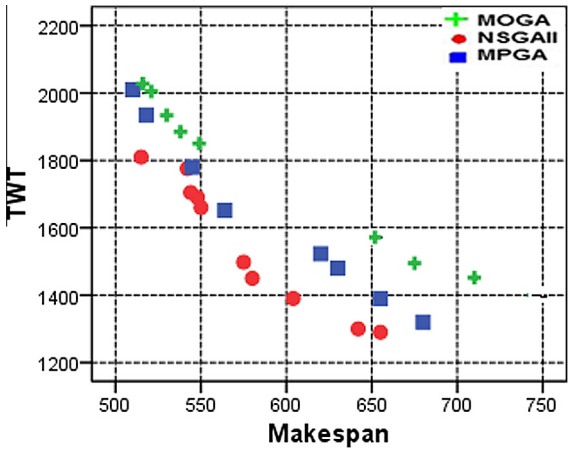

Fig. 8. A single run of the non-dominated solutions for medium size problem.

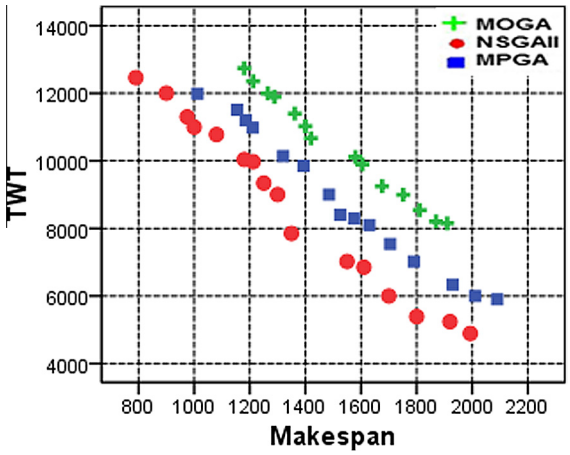

Fig. 9. A single run of the non-dominated solutions for large size problem.

5.2.3. CPU time

This metric represents the amount of time required by each algorithm to achieve the pareto set.

5.2.4. Parameter setting

The value of different parameters is one of the effective factors on the performance of each algorithm. Assigning the suitable values to each parameter will result in better outputs. In addition different combination of parameter values may result in obtaining a non-dominated solution set with different abilities. A number of experiments have been done using different sets of parameters in order to set the best values for parameters of the two developed algorithms. For this purpose, we considered three problems with three different sizes and determined the best level of each parameter using practical study. Paying attention that two algorithms have been considered in this paper, the proposed parameters for each of these algorithms are different. These parameters are shown in Table 2.

5.3. Experimental results

In this section we are going to compare the proposed algorithms with the MPGA of Karimi et al. [9]. The proposed algorithms and MPGA are coded in MATLAB R2008b and run with an Intel Pentium IV dual core 2 GHz PC at 2 GB RAM under a Microsoft Windows XP environment. With respect to the different set of inputs, each instance is run 10 times and each digit is average of 10 runs of each problem. For each of the algorithms three performance metrics have been considered and their results are shown in Table 3. Figs. 7–9 show the non-dominated solutions of a single run by proposed algorithms, NSGAII and MOGA, and its superiority to benchmark MPGA for problems of small, medium, and large sizes, respectively.

6. Conclusion

Scheduling is one of the important tasks in production planning systems. One of the complicated scheduling environments is HFS environment that is also very useful in real world. The assumptions made in this paper and makes it distinguish from other studies is considering the group setup time dependent on sequence and due date as being stochastic. Considering due date as stochastic makes this study closer to the real world problems. In this paper due date is assumed stochastic and a new stochastic modeling has been developed. If we apply new production philosophies like cell production and group technology too, the need to consider setup time in scheduling would appear more noticeable. We can consider cell production as an application of group technology. When jobs are classified in different groups, there are usually sequence-dependent setup times between groups. Assignment of several groups of jobs to a set of machinery that has cell configuration will result in a problem that is known as group scheduling. Therefore group scheduling problem of sequence-dependent setup times in hybrid flow shop environment is one of significant problems in different industries. On the other hand because most of optimization problems in the real world have more than one objective, we considered scheduling problem with sequence-dependent group setup times in hybrid flow shop environment having two objectives of minimization of makespan and total tardiness. Because the desired problems in this study are *NP-hard* and exact solution methods cannot solve them in real environments, we developed two multi-objective evolutionary MOGA and NSGAII for our desired problem and compared the results with MPGA algorithm used in similar problem. The results indicated that NSGAII will obtain better solution sets in shorter time. The future work is to change the objectives that we used in this paper, assumption of uncertainty for other parameters and extension of proposed algorithms to other environments of scheduling.

References

[1] C. Rajendran, D. Chaudhuri, Scheduling in *n*-jobs *m* stage flow shop with parallel processors to minimize makespan, Int. J. Prod. Econ. 27 (2) (1992) 137–143.
[2] R. Linn, W. Zhang, Hybrid flow shop scheduling: a survey, Comput. Ind. Eng. 37 (1) (1999) 57–61.
[3] E.L. Ulungu, J. Techem, P.H. Fortemps, D. Tuytten, MOSA method: a tool for solving MOCO problems, J. Multi-Criteria Decision Analy. 8 (1999) 221–236.
[4] J. Behnamian, S.M.T. Fatemi Ghomi, Hybrid flow shop scheduling with machine and resource-dependent processing times, Appl. Math. Model. 35 (2011) 1107–1123.
[5] V.R. Ghezavati, M. Saidi-Mehrabad, An efficient hybrid self-learning method for stochastic cellular manufacturing problem: a queuing-based analysis, Expert Syst. Appl. 38 (2011) 1326–1335.
[6] R. Tavakkoli-Moghaddam, N. Javadian, A. Khorrami, Y. Gholipour-Kanani, Design of a scatter search method for a novel multi-criteria group scheduling problem in a cellular manufacturing system, Expert Syst. Appl. 37 (2010) 2661–2669.
[7] R. Tavakkoli-Moghaddam, F. Jolai, F. Vaziri, P.K. Ahmed, A. Azaron, A hybrid method for solving stochastic job shop scheduling problems, Appl. Math. Comput. 170 (2005) 185–206.
[8] J.N.D. Gupta, W.P. Darrow, The two-machine sequence-dependent flow shop scheduling problem, Eur. J. Oper. Res. 24 (1986) 439–446.
[9] N. Karimi, M. Zandieh, H.R. Karamooz, Bi-objective group scheduling in hybrid flexible flow shop: a multi-phase approach, Expert Syst. Appl. 37 (2010) 4024–4032.
[10] T.C.E. Cheng, Optimal due-date assignment for a single machine sequencing problem with random processing times, Int. J. Sys. Sci. 17 (1986) 1139–1144.
[11] A. Elyasi, N. Salmasi, Stochastic flow shop scheduling with minimizing the expected number of tardy jobs, Int. J. Adv. Manuf. Technol. 66 (2013) 337–346.
[12] J. Kamburowski, Stochastically minimizing the makespan in two-machine flow shops without blocking, Eur. J. Oper. Res. 112 (1999) 304–309.
[13] J. Kamburowski, On three-machine flow shops with random job processing times, Eur. J. Oper. Res. 125 (2000) 440–449.
[14] C. Rajendran, O. Holthaus, A comparative study of dispatching rules in dynamic flow shops and job shops, Eur. J. Oper. Res. 116 (1) (2005) 361–370.
[15] L. Tang, W. Liu, J. Liu, A neural network model and algorithm for the hybrid flow shop scheduling problem in a dynamic environment, J. Intel. Manuf. 16 (3) (2005) 361–370.
[16] M. Gourgand, N. Grangeon, S. Norre, A contribution to the stochastic flow shop scheduling problem, Eur. J. Oper. Res. 151 (2003) 415–433.
[17] K.R. Baker, D. Altheimer, Heuristic solution methods for the stochastic flow shop problem, Eur. J. Oper. Res. 216 (2012) 172–177.
[18] C. Almeder, R.F. Hartl, A metaheuristic optimization approach for a real-world stochastic flexible flow shop problem with limited buffer, Int. J. Prod. Econ. 145 (2013) 88–95.
[19] Wang, L. Zhang, D.Z. Zhang, A class of hypothesis-test-based genetic algorithms for flow shop scheduling with stochastic processing time, Int. J. Adv. Manuf. Technol. 25 (2005) 1157–1163.
[20] Harun. Aydilek, Ali. Allahverdi, Two machine flow shop scheduling problem with bounded processing times to minimize total completion time, Comput. Math. Appl. 59 (2010) 684–693.
[21] F. Ahmadizar, M. Ghazanfari, S.M.T. Fatemi Ghomi, Group shops scheduling with makespan criterion subject to random release dates and processing times, Comput. Oper. Res. 37 (2010) 152–162.
[22] K. Kianfar, S.M.T. Fatemi Ghomi, A. Oroojlooy Jadid, Study of stochastic sequence-dependent flexible flow shop via developing a dispatching rule and a hybrid GA, Eng. Appl. Artif. Intel. 25 (2012) 494–506.
[23] Jiuping Xu, Xiaoyang Zhou, A class of multi-objective expected value decision-making model with bi-random coefficients and its application to flow shop scheduling problem, Inform. Sci. 179 (2009) 2997–3017.
[24] Jinwei Gu, Manzhan Gu, Cuiwen Cao, Xingsheng Gu, A novel competitive co-evolutionary quantum genetic algorithm for stochastic job shop scheduling problem, Comput. Oper. Res. 37 (2010) 927–937.
[25] A. Elyasi, N. Salmasi, Due date assignment in single machine with stochastic processing times, Int. J. Prod. Res. 51 (2013) 2352–2362.
[26] A. Elyasi, N. Salmasi, Stochastic scheduling with minimizing the number of tardy jobs using chance constrained programming, Math. Comput. Model. 57 (2013) 1154–4464.
[27] Wooseung Jang, C.M. Klein, Minimizing the expected number of tardy jobs when processing times are normally distributed, Oper. Res. Lett. 30 (2002) 100–106.
[28] D. Golenko-Ginzburg, Z. Laslo, Chance constrained oriented dispatching rules for flexible job shop scheduling, Comput. Model. Technol. 8 (2004) 14–18.
[29] K. Dong Seo, C.M. Klein, Wooseung Jang, Single machine stochastic scheduling to minimize the expected number of tardy jobs using mathematical programming models, Comput. Ind. Eng. 48 (2005) 153–161.
[30] H.M. Soroush, Minimizing the weighted number of early and tardy jobs in a stochastic single machine scheduling problem, Eur. J. Oper. Res. 181 (2007) 266–287.

[31] C. Proust, J.N.D. Gupta, V. Deschamps, Flow shop scheduling with setup, processing, and removal time separated, Int. J. Prod. Res. 29 (1991) 479–493.
[32] R.G. Askin, A.J. Vakharia, Group technology-cell formation and operation in the automated factory, in: D.I. Cleland, B. Bidanda (Eds.), Handbook: Technology and Management, TAB Books, New York, 1991, pp. 317–366.
[33] S. Li, W. Huang, A two-stage hybrid flow shop with uniform machines and setup times, Math. Comput. Model. 27 (1998) 27–45.
[34] R. Logendran, P. Deszoeke, F. Barnard, Sequence-dependent group scheduling problems in flexible flow shop, Int. J. Prod. Econ. 10 (2006) 66–86.
[35] R. Logendran, N. Salmasi, C. Sriskandarajah, Two-machine group scheduling problems in discrete parts manufacturing with sequence-dependent setups, J. Comput. Oper. Res. 33 (2006) 158–180.
[36] J.E. Schaller, J.N.D. Gupta, A.J. Vakharia, Scheduling a flowline manufacturing cell with sequence dependent family setup times, Eur. J. Oper. Res. 125 (2000) 324–339.
[37] O. Shahvari, N. Salmasi, R. Logendran, B. Abbasi, An efficient tabu search algorithm for flexible flow shop sequence-dependent group scheduling problems, Int. J. Prod. Res. 50 (2012) 4237–4254.
[38] N. Salmasi, R. Logendran, M.A. Skandari, Total flow time minimization in a flowshop sequence-dependent group scheduling problem, J. Compu. Oper. Res. 37 (2010) 199–212.
[39] N. Salmasi, R. Logendran, M.A. Skandari, Makespan minimization in a flowshop sequence-dependent group scheduling problem, Int. J. Adv. Manuf. Technol. 56 (2011) 699–710.
[40] D. Hajinejad, N. Salmasi, R. Mokhtari, A fast hybrid particle swarm optimization algorithm for flow shop sequence dependent group scheduling problem, ScientiaIranica E 18 (2011) 759–764.
[41] Dvir Shabtay, Yisrael Itskovich, Liron Yedidsion, Daniel Oron, Optimal due date assignment and resource allocation in a group technology scheduling environment, Comput. Oper. Res. 37 (2010) 2218–2228.
[42] J.D. Schaffer, Multiple objective optimization with vector evaluated genetic algorithms, in: Proceedings of First International Conference on Genetic Algorithms and Their Applications, Carnegie-Mellon University, Pittsburgh, PA, USA. July 1985, pp. 93–100.
[43] T. Murata, H. Ishibuchi, H. Tanaka, Multi-objective genetic algorithm and its application to flow shop scheduling, Comput. Ind. Eng. 30 (1996) 957–968.
[44] K. Deb, S.A. Amrit Pratap, T. Meyarivan, A fast and elitist multi objective genetic algorithm NSGA-II, in: Proceedings of the Parallel Problem Solving from Nature VI Conference, Paris, France, 2000, pp. 849–858.
[45] C.M. Fonseca, P.J. Fleming, Genetic algorithms for multi-objective optimization: formulation, discussion, and generalization, in: S. Forrest (Ed.), Proceedings of the 5th International Conference on Genetic Algorithms, Morgan Kaufman Publishers, University at Urbana-Champaign, San Mateo, California, 1993, pp. 416–423.
[46] J. Horn, N. Nafpliotis, D.E. Goldberg, A niched Pareto genetic algorithm for multi-objective optimization, in: Proceedings of 1st IEEE ICC Conference, 1994 pp. 82–87.
[47] K. Deb, Multi-objective genetic algorithms: problem difficulties and construction of test problems, Evol. Comput. J. 7 (3) (1999) 205–230.
[48] C.J. Hyun, Y. Kim, Y.K. Kim, A genetic algorithm for multiple objective sequencing problems in mixed model assembly lines, Comput. Oper. Res. 25 (7–8) (1998) 675–690.
[49] A. Jaszkiewicz, Genetic local search for multiple objective combinatorial optimization, Technical Report RA-014/98, Poznan University of Technology, Institute of Computing Science, 1999.
[50] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: improving the strength pareto evolutionary algorithm, Optimiz. Cont. Appl. Ind. Prob. 65 (2001) 95–100.
[51] R.P. Beausoleil, Multi objective scatter search applied to non-linear multiple criteria optimization, Eur. J. Oper. Res. 169 (2006) 426–449.
[52] J. Bean, Genetic algorithms and random keys for sequencing and optimization, J. Comput. 6 (1994) 154–160.
[53] B.A. Norman, J.C.A. Bean, A genetic algorithm methodology for complex scheduling problems, Nav. Res. Logis. 46 (1999) 199–211.
[54] N. Sirnivas, K. Deb, Multi-objective function optimization using non-dominated sorting genetic algorithms, Evol. Comput. 2 (3) (1995) 221–248.