ISSN 1 746-7233, England, UK
World Journal of Modelling and Simulation
Vol. 5 (2009) No. 4, pp. 261-271

Minimizing expected makespan in flowshop with on condition based maintenance constraint by integrating heuristics and simulation

E. Safari*, S. J. Sadjadi, K. shahanaghi

Department of Industrial engineering, Iran University of Science and Technology, Narmak, Tehran-16 15614, Iran

Abstract. The availability of the machine is one of the most important assumptions on many scheduling problems under any situation. However, in most actual life manufacturing settings a machine can be unavailable for many reasons, such as unforeseen scheduled preventive. In this paper, we study scheduling flowshops problem under 'condition based' maintenance constraints to minimize the expected makespan using simulation to tackle the randomness of the problem at hand. Heuristics and meta hueristics is used to tackle such an NP-hard problem. It is additionally supposed that machines suffer form degradation due to shocks, thus preventive maintenance should be done on machines. Furthermore, it is assumed that the degradation value of machine is known at inspection time. Overall, three adaptations of existing metaheuristic and hueristics methods are evaluated for the integration of preventive maintenance and they are applied to a set of 1080 instances. The preliminary results manifest the superiority of genetic algorithm over others.

Keywords: flowshop, on condition based maintenance, simulation, heuristics

1 Introduction

The classical flow shop problem with the makespan criterion is certainly the most often studied in scheduling theory. The problem is defined as follow: A set of n jobs, $J = \{1, 2, \cdots, n\}$, available at time zero has to be processed in a shop with m machines $M_1, M_2, \cdots, M_m$ in the same order. Each job is processed first on M_1, next on M_2, and so on, and finally on M_m.

Some other job closely related to production scheduling in industrial settings is maintenance, understood as the operations or techniques that allow maintaining or restoring equipment to a specific state and guaranteeing a given service. Scheduling of maintenance operations and production sequencing are normally treated separately. These activities are often in conflict. As the maintenance operations waste production time whereas delaying maintenance operations until the production sequence allows a period of free time may increase the probability of machines failure. During the scheduling we normally assume that the machine is available during the planning time horizon but this simple assumption may not hold for many real world applications. In real world problems machines may be unavailable due to maintenance, breakdown, and repair; since this unavailability may be stochastic or deterministic. Recently, many Maintenance Policy Condition based maintenance rules imply to decreasing in maintenance operation costs. Maintenance costs can be the largest part of any operational budget and it is extensively employed in production environment so it is necessary to consider this maintenance constraint on flow shop problem.

This paper deals with a practical and stochastic flow shop problem in which scheduling of jobs is done under the maintenance constraints i.e. condition based maintenance in which each machine suffers degradation due to shocks and when total degradation for each machines in inspection interval is reached to preventive maintenance thresholds. Then, preventive maintenance operations are accomplished. The inspection is planned at periodic $nT_I(n = 1, 2, 3, \cdots)$. It is assumed that we could determine the total degradation to

* Corresponding author. *E-mail address*: ehram_s@yahoo.com.

each machine only through inspection. We assume that shocks, PM time, increasing in machines degradation value due to each shock and restoration of machines after minimal preventive maintenance are stochastic in nature. Scheduling criteria is considered as minimizing the expected makespan. This scheduling environment is general model which could be employed in series work shop such as automobile, chemical and textile manufacturing systems.

In the literature two processing models have been considered: "resumable" and "non-resumabl". These terms are defined as follows[14]: Suppose the maintenance period interrupts the processing of a job. If the job continues processing after the machine becomes available again without losing any time or penalty for resumption, the problem was called "resumable". On the other hand, the problem was called "non-resumable" if the job has to restart from the beginning (all prior processing is wasted) when the machine becomes available again.In this paper we have two objectives: We first show how we can combine simulation and hueristics. The second heuristics are compared with others heuristics for different instances.

Since the publication of Johnson's paper[12] in 1954, various approaches have been proposed for flow shops sequencing problem with an objective of minimizing makespan. These approaches can be classified into different categories: optimization algorithms for the exact solution and heuristic algorithms for near optimal solution. Heuristics techniques have been proposed for solving flow shop characterized as either constructive (Nawaz et al.[18]) or improvement (Reeves[19], Ishibuchi et al.[11], Taillard[6, 7]). Constructive scheduling generates a schedule from scratch, and is usually considered as extensions of the Johnson's algorithm to solve flow shop problems. Improvement methods start from a solution generated by some sequence-generating heuristics and using suitable schemes improves this solution with the hope of approaching the global optimum. Improvement methods are generally based on meta-heuristic approaches such as Genetic algorithms (GA), Simulated Annealing (SA), etc.

The presence of preventive maintenance periods in machine availability was treated by several researchers: Lee[14] proved that the problem with one preventive period on only one machine is NP-hard in the ordinary sense. He developed a pseudo-polynomial dynamic programming model if the period is only on the first machine. He also proposed two heuristic algorithms of time complexity O (n log n), and provided their error bound analysis. Schmidt[21] investigated scheduling problems with limited machine availability in greater detail. He provided results for one machine, parallel machines and flow shops. He analyzed some results from enumerative optimization algorithms and heuristics. The problem with some periods of preventive maintenance on two machines in resumable case was considered by Blazewicz et al.[5] They analyzed constructive and local search based heuristic algorithms. Glazebrook[8] investigated a single-machine problem subject to breakdowns and formulated it as a cost-discounted Markov decision process. Birge et al.[4] considered more general breakdown processes. Allahverdi et al[1] showed that a problem with parallel machines subject to random breakdowns could be converted into an equivalent deterministic parallel-machine problem with modified processing time, when the problem was to minimize the expected mean flow time and the breakdowns follow a generalized Poisson process. Allaoui et al.[9] considered hybrid flow shop with maintenance constraints and scheduled jobs in machine by integrating simulation and optimization methods and then compared SA and neh heuristics. Ruiz et al.[20] considered two popular preventive maintenance methods in flow shop scheduling problem; also they compared different different meta-heuristics and hueristics that existed in literature.

The reminder of this paper is organized as follows. In the section 2, a study of CBM and the modeling is presented. Problem assumptions are described in section 3. Section 4 describes the heuristics that are applied in this paper. In section 5, combination of simulation and heuristics is denoted. Finally, the experimental studies are performed in section 6.

2 Condition based maintenance

The earliest maintenance technique was based on a simple rule: Don't do anything until a machine fails. This fact was used for many years until people came to develop some more sophisticated maintenance methods called Preventive Maintenance (PM). There are two basic methods for PM: The time based maintenance[24] (TBM) and the condition based maintenance[15] (CBM). TBM usually suffers from a primary problem which

sets a periodic interval to perform preventive maintenance regardless of the health status of a physical asset. With the rapid development of modern technology, products have become more and more complex while better quality and higher reliability are required. This makes the cost of PM higher and higher. Eventually, PM has become a major expense of many industrial companies. Therefore, people change their strategy to use CBM which is normally less costly. CBM recommends maintenance actions based on the information collected through condition monitoring. CBM attempts to avoid unnecessary maintenance tasks by taking maintenance actions only when there is evidence of abnormal behaviors of a physical asset. A CBM program, if properly established and effectively implemented, can significantly reduce maintenance cost by reducing the number of unnecessary scheduled preventive maintenance operations. In fact, any action in CBM depends entirely on the status of the machine. The outcome of our actions could be divided into three statues: No Action, Minimal Preventive Maintenance (MPM) and Basic Preventive Maintenance (BPM).

This paper adopts the CBM policy for a cumulative degradation model[23] where a machine suffers degradation due to shocks, and does BPM when the total amount of additive degradation exceeds a level called k. The inspections occur in a pre-specified intervals $nT_I(n = 1, 2, \cdots)$ to prevent failures, where $T_I(> 0)$ is the inspection interval time. If the total degradation exceeds a threshold level $Z(0 < Z \leq K)$ at time nT, the MPM is performed. Otherwise, no PM is performed.

In this paper we assume that shocks are the main reason of degradation for each machine. The number of shocks occurrence in jth inspection interval N_j has Poisson distribution. In additional, random variables $\{W_j\}(j = 1, 2, \cdots)$ denotes the amount of degradation due to jth shock and is considered to have an exponential distribution. Degradation of each machine in a period d_p is also considered as: $d_p = \sum_{j=1}^{N_p} W_j$.

Recovery value (Decreasing the degradation value after PM operations) Δ_{deg} when operation maintenance is executed, has lognormal distribution[3]. Also degradation values after MPM and BPM operations d_0 are equal to $d - \Delta_{\text{deg}}$ and 0, respectively. The amount of time needed for MPM and BPM operations are T_m, T_b respectively, have lognormal distribution, where time needed for operations of BPM are more than MPM and we have $d_t = d_0 + d_p$. Where d_0 is identified at the end of each period for the next period as follows:

$$d_0 = \begin{cases} d_t - \Delta_{\text{deg}} & Z \leq d_t < K \\ d_t & 0 \leq d_t \leq Z \\ 0 & d_t \geq K \end{cases}$$

3 Assumptions

- A job consists of several operations, each of which has to be performed on a specified machine.
- All jobs are available for processing at time zero.
- Setup times between operations are negligible or included in the processing times (sequence-independent setup times).
- No machine can process two jobs at the same time.
- Jobs are always processed without error.
- Job processing times are known in advance.
- Each job is processed on one machine at a time.
- There are no preference constraints among the jobs.
- There is only one type of machine.
- Infinite buffers exist between machines.
- No preemption of jobs is allowed.
- Machines are not available at all times due to MPM and BPM operations.
- Machine inspections are planned at periodic times $nT(n = 1, 2, \cdots)$.
- All machines have the same mean values shocks, degradation for each shocks, MPM time, BPM time, recovery value.
- MPM and BPM threshold are same for all machines.

- The inspection operations do not disrupt the operations of a job (the job can continue its processing while the inspection operations are being done).

4 Hueristics

4.1 NEH1 heuristic

For a frow shop problem, Nawaz, Enscore, and Ham[18] propose that a job with big total processing time should have higher priority in the sequence to minimize the makespan. A schedule is developed by the job-insertion technique. Regarding makespan minimization, all comparison studies concluded that the NEH heuristic is the most efficient. In this paper we develop a heuristic namely, NEH1, based on NEH heuristic as fallows: NEH1 heuristic consists of two steps: order n jobs in decreasing processing times on machine, that it has maximum load. Within the second step, a job sequence is constructed by evaluating the partial schedules originating from the initial order of the first step. Suppose a sequence already determined for the first k jobs, $k+1$ candidate sequences are obtained by inserting job $k+1$ in the $k+1$ possible slots of the current sequence. Out of these $k+1$ sequences, the one yielding the minimum makespan is kept as relative sequence for these first $k+1$ jobs given by step one. In The selected partial sequence, for d times, two point of this sequence is selected randomly and exchanged their positions.Then, job $k+2$ from phase I is considered analogously and so on until all jobs have been sequenced.

4.2 Simulated annealing

SA has its origin in statistical physics, where the process of cooling solids slows until they reach a low energy state called annealing. It was originally proposed by Metropolis[16] and was first applied to combinatorial optimization problems by Kirkpatrick[13].

The SA process starts with an initial solution S_0 the temperature parameter is set to an initial value T_0, high enough so that all the transitions (moves) can be allowed. At the kth step with a current solution S_k, step with a current solution S_k in the neighborhood of $S_k^{'}$ and it evaluates the two values of the objective function $H(S_k)$ and $H(S_k^{'})$ associated with S_k and $S_k^{'}$, respectively. Any downhill move is accepted and the process continues from the new point. An uphill move may be accepted to escape from local optima. This uphill decision is determined by a sequence of random numbers with a controlled acceptance function which normally is assumed to be exponential and this probability function is typically a Boltzmann-type distribution. Thus, a move from point S_k to another point $S_k^{'}$ resulting in a variation $\Delta H = H(S_k^{'}) - H(S_k)$, is still accepted if $r < p(\Delta H)$; where $p(\Delta H) = \min(1, e^{\frac{-\Delta H}{T_k}})$ and $r \in [0,1]$ is a uniform random number. The algorithm proceeds by attempting a certain number of neighborhood moves at temperature T_k, while the temperature parameter is gradually dropped. The procedure is repeated until a stopping criterion is reached Ishibuchi et al.[11] apply modified SA heuristic flow shop problems to minimize the makespan. They show that this algorithm had robust performance with respect to the choice of a cooling schedule. This algorithm is organized as follows: Initial solution generate by random. Next solution in SA is selected based on the first solution that improves current solution in generated k solutions in neighborhood of current solution. Neighborhood structure of this algorithm is shift neighborhood, in which two jobs is selected along sequence of current solution randomly, then insert jobs in position 1 in to position 2 and after that the jobs is shifted one unit in right direction. X_{best} then records the best solution as the proposed approach progresses. T is decreased after running I_{itr} repetitions from the previous decrease, according to a formula $T \leftarrow \alpha T$, where $0 < \alpha < 1$. When T is decreased once. If T is lower than T_f, the algorithm is terminated. Preliminary experiments showed that if values of initial T, k, α, I_{iter} and T_f is considered 100, 20, 0.98, 25 and 1.7, respectively, The algorithm could be obtained better results related to other parameter settings.

4.3 Genetic algorithm

Genetic algorithms (GAs) have been used successfully to find near-optimal solutions for a wide variety of optimization problems and it was introduction by Holland[10] and Revees[19] applied it on flow shop scheduling. GAs is intelligent stochastic optimization techniques based on the mechanism of natural selection and

genetics. GAs start with an initial set of solutions, called population. Each solution in the population is called a chromosome (or individual), which represents a point in the search space. The chromosomes are evolved through successive iterations, called generations, by genetic operators (selection, crossover and mutation) that mimic the principles of natural evolution. In a GA, a fitness value is assigned to each individual according to a problem-specific objective function. The new individuals, called offspring, are created and survive with chromosomes in the current population, called parents, to form a new population. The main characteristics of the GA implementation for this paper are written as follows:

(1) Representation mechanism: permutation strings are used, i.e., strings of integers, with each string corresponding to a unique flowshop scheduling solution.

(2) Initial population: seeding mechanism is used to generate the initial population. Assuming a population of M chromosomes and are generated at random.

(3) Genetic operators: two genetic operators are used: crossover and mutation operator. Two point crossovers is considered as crossover operator, a pair of crossing points is randomly selected along of the first parent then the sequence of the jobs between the selected crossing points are copied into the child, and the remaining jobs are taken from the second parent in the order of their appearance. Then generated child is replacing with parent that has the biggest makespan. Also mutation operator is displacement mutation. This operator selects randomly a sub-sequence of jobs and inserts it in a random position. In facts, the selected sub-sequence of jobs is moved left or rights a (randomly chosen) number of positions.

(4) Selection mechanism: selection of parent chromosomes is done by using tournament: the three chromosomes are randomly selected and chromosome with highest fitness (lower makespan), is selected for reproduction.

(5) Control parameters: Based on initial experiments, the parameters of the genetic algorithm are selected as follows: Population. Population size $= 100$, crossover rate $= 0.45$, mutation rate $= 0.05$.

(6) Termination condition: the Ga stops when maximum number of 1800 generation has been surpassed or algorithm do not improve solutions in 200 successive generations.

5 Combination of simulation and optimization

The majority optimization problems are too difficult to be solved by mathematical programming. Recent research for combining optimization and simulation has led to several hybrid methods to handle these complicated problems. These hybrid methods were originated from different areas, such as operations research and artificial intelligence. Nadoli et al.[17] used the concept of intelligent agents to simulate the manufacturing process. A. D. Talbi et al.[22] combined logical programming object-oriented approach, constraints programming, and simulation. A multimodal approach combining simulation, artificial intelligence (AI) and operations research was proposed in [2].

Since the problem considered in this paper is strongly NP-hard, we make use of Hueristics along with simulation technique (called simulator) to obviate the complexity of the problem at hand. The incorporation of condition base maintenance will intensify the complexity of the problem. Hence, to overcome the complexity of the problem which is of stochastic nature, we utilize simulation techniques to calculate the makespan. Here, we determine how we can tackle scheduling flow shop problem in which machines suffer from stochastic degradation to optimize the objective, based on expected makespan (Fig. 1).

Before explaining the algorithm, it is essential to notice some points:

- It is necessary that we determine the number of shocks for each machine in inspection intervals.
- In each inspection interval, it is required to calculate degradation value related to each shocks and cumulative value of degradation.
- For identifying total degradation of each machine, degradation in previous inspection interval is added to current one.
- In each inspection, it is needed to compare degradation value with thresholds.
- Succeeding that identifying maintenance policy, PM or basically repair time, they should be calculated.
- After PM operation, we should identify recovery value to subtract from degradation value.

- After basically repair, degradation value set to zero.
- It is necessary to note where inspection is occurred i.e. in middle or end time of particular processing job.
- Because of the fact that the problem has probabilistic nature, it is necessary to replicate the computation of simulator several times (nu.sim) for each sequence when all the features of the problem remain constant.
- After that all replication is done, mean of obtained makespan is considered as expected makespan to using of hueristics.
- Some notation

n	:	number of jobs;	d	:	degradation value when machine is inspected;
m	:	number of machines;	FT_{ij}	:	finish time of job j on machine;
σ	:	sequence of jobs;	σ_K	:	kth job of sequence σ;
W	:	degradation value for each shock;	t_{ij}	:	processing time of job j on machine i;
tms	:	remaining time to next inspection;	N	:	number of shock in each inspection period;
ST_{ij}	:	start time of job j on machine i.			

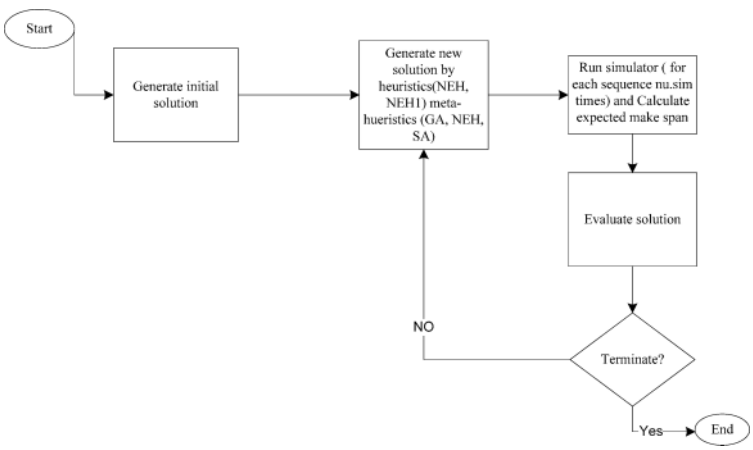

Fig. 1. Combining simulation and optimization

5.1 Simulator algorithm

The simulator algorithms employed in heuristics has the following steps for each sequence σ

Step 0. Initialize $i = 1,\ ST_{1,\sigma_1} = 0,\ FT_{0,j} = 0,\ d_0 = 0.$

Step 1. If $i \leq m$ {go to step 2}; Else {Finish the simulator procedure }.

Step 2. Set $d_0 = 0,\ j = 1$; Run degradation algorithm with initial degradation d_0.

Step 3. If $j \leq n$ {go to step 4}; Else {go step 14}.

Step 4. Set $g = \sigma_j,\ tms = tms - t_{ig},\ ST_{i,g} = \max(FT_{i-1,g}, ST_{i,g})$.

Step 5. if $tms > 0$ {go step 6}; Else {go step 7 }.

Step 6. Set $FT_{ig} = ST_{ig} + t_{ig},\ ST_{i,\sigma_{j+1}} = FT_{i,g}$ and go step 13.

Step 7. if $d \geq k$ {If $tms < 0$ {Set $tms = tns + t_{i,g}$; Go to step 8}; Else {Go to step 11}};
Else if $d \geq Z$ {If $tms < 0$ {Set $tms = tms + t_{i,g}$; Go to step 9}; Else {Go to step 12}};
Else {Go to step 10}.

Step 8. (a) Generate a lognormal random number as BPM time T_b; Set $d_0 = 0$,
$FT_{i,g} = ST_{i,g} + tms + T_b + ti, g$. (b) Run degradation algorithm with initial degradation d_0;
Set $tms = T_I - t_{i,g}, ST_{i.\sigma_{j+1}} = FT_{i,g}$; Go to step 13.

Step 9. (a) Generate two lognormal random numbers as MPM time T_m, recovery value $\Delta_{\deg}$;
Set $FT_{i,g} = ST_{i,g} + tms + T_m + t_{i,g}; d_0 = d - \Delta_{\deg}$; (b) Run degradation algorithm with initial degradation d_0; Set $tms = T_I - t_{i,g}; ST_{i,\sigma_{i,g}};$ Go to step 13.

Step 10. Set $FT_{i,g} = ST_{i,g} + t_{i,g}; tms = T_I + tms - t_{i,g}$; Run degradation algorithm with initial

degradation d; Go to step 13.

Step 11. (a) Generate a lognormal random number as BPM time T_b; Set $FT_{i,g} = ST_{i,g} + t_{i,g}, d_0 = 0$.
(b) Run degradation algorithm with initial degradation d_0; Set $tms = T_I; ST_{i,\sigma_{j+1}} = FT_{i,g} + T_b$;
Go step 13.

Step 12. (a) Generate two lognormal random numbers as MPM time T_m, recovery value $\Delta_{\deg}$;
Set $FT_{i,g} = ST_{i,g} + t_{i,g}; d_0 = d - \Delta_{\deg}$; (b) Run degradation algorithm with initial degradation d_0;
Set $tms = T_I; ST_{i,\sigma_{j+1}} = FT_{i,g} + T_m$; Go to step 13.

Step 13. Set $j = j + 1$; Go to step 3.

Step 14. Set $i = i + 1$; Go to step 1.

Now we consider that the jobs can be resumable after preventive maintenance. In order to caculate the makespan we can rewrite the steps of previous algorithm except steps 5, 7, 8 and 9. These steps are written as follows:

Step 5. If $tms > 0$ {Go to step 6}; Else {Go step 7}.

Step 7. If $tms < 0$ {Go to step 8}; Else {Go to step 11}; Else if $d \geq Z$ {If $tms < 0$ {Go step 9};
Else {Go to step 12 }}; Else {Go to step 10 }.

Step 8. (a) Generate a lognormal random number as basic maintenance time T_b then set $d_0 = 0$,
$FT_{i,g} = st_{i,g} + T_b + t_{i,g}$, (b) Run degradation algorithm with initial degradation set d_0;
Set $tms = tms + T_I, ST_{i,\sigma_{j+1}=FT_{i,g}}$; Go to step 13.

Step 9. (a) Generate two lognormal random numbers as MPM time T_m, recovery value $\Delta_{\deg}$;
Set $FT_{i,g} = ST_{i,g} + T_m + t_{i,g}; d_0 = d - \Delta_{\deg}$; (b) Run degradation algorithm with initial
degradation d_0; Set $tms + T_I; ST_{i,\sigma_{j+1}=FT_{i,g}}$; Go to step 13.

5.1.1 Degradation algorithm with initial degradation d_0

The degradation algorithm is used in simulator algorithm which has the following steps:

Step 1. generate a Poisson random number as shock number N.

Step 2. $d = d_0$,

Step 3. for $k = 1$ to N:
(a) Generate an exponential random number as degradation for shock k, W.
(b) $d = d + W$.
End for (k)

Step 4. return to simulator algorithm.

6 Experimental studies

In this section, we address the computational results obtained from our proposed algorithm and other algorithms employed in this paper. We first determine the CBM parameters. In practice the parameter related to stochastic distribution of the number of shocks in inspection period, BPM and MPM operation times, recovery value and degradation value for each shock are specified according to the information collected through monitoring. Also threshold values are determined according to life distribution of machines or empirical data. Without the loss of generality, we suppose that the number of socks in each inspection period and degradation value for each shock have Poisson and exponential distribution with 20 and 0.5 mean values, respectively. Also recovery value after MPM operations is fixed using lognormal distribution with mean 3 and variance value of 0.1. This parameter is selected such that the generated random number from this distribution is less than MPM threshold and more than zero. The settings for all parameters are fixed for the whole configurations of experiments. In order to evaluate the effectiveness of the proposed meta-heuristic with the other meta-heuristics and heuristics, it is necessary to consider MPM, BPM duration and the inspection period with different parameter settings. Since, the lengths of both periods of inspection and preventive maintenance might influence the efficiency and effectiveness of the algorithm, we consider different levels of these parameters for our problem. We have $n\{20, 30, 40\}$ and $m = \{20, 30, 40\}$, therefore there are 9 combinations of n and m

where the processing times t_{ij} is distributed as a $U[1, 99]$ as it is common in flow shop scheduling research. In this paper, for each combination of jobs and machines, different values of inspection interval and necessary times of operations of minimal and basic preventive maintenance are considered. Hence, we consider PB1 for operation times of minimal and basic maintenance with log normal distribution with means 3.0 and 3.65 and variances of 0.1. We also consider PB2 for operation times of minimal and basic maintenance with log normal distribution with means of 3.65 and 4.35 and variances of 0.1. This setting is shown by PB1, PB2, respectively where the average MPM duration is 40%, 80%, and BPM duration is 80%, 160% of the average of processing times for PB1, PB2 respectively. Inspection period is set $\{150.250, 350\}$. For each configuration n, m, T_1 and PB, we run 10 different instances which result in a total set of 960 instances. Since time needed to run a simulator is great, thus the number of runs for simulator is considered 2. Our approach with the listed data runs on a Pentium IV 3.2 GHz. We developed program of this experiment on Microsoft visual Studio 2005 C++.

6.1 Experimental results

Once the expected makespan of each instance has been obtained for each algorithm, the best solution obtained for each instance by any of the six algorithms is selected and it is called $S_{\min}$. With this, we can calculate the relative percentage deviation (RPD) with respect to this best solution with the following expression:

$$\text{Rlative percentage diviation(RPD)} = \frac{\text{S}_{\text{algorithm}} - \text{S}_{\min}}{\text{S}_{\min}} \times 100$$

where $S_{\text{algorithm}}$ is expected makespan attained for considered algorithm and instances. RPD help us to compare algorithms because the RPD values denote relative distance of algorithm solution from best solution obtained for special instance, clearly, lower values of RPD are preferred.

6.1.1 Non-resumable case

In Tab. 1 we summarize the average relative percentage over 540 test problems for the three heuristics GA, SA and NEH1. From this table, we can see, the GA reaches better results related to other algorithms in all instances. Also SA has better performance than NEH1. For validation of the obtained results that is shown in Tab. 1, statistical analysis is performed i.e. Multifactor ANOVA in which factors are n, m, T_I, Pb and response variable is RPD. Preliminary experiments demonstrate that ANOVA's hypothesis i.e. normality, homogeneity of variance and independence of residuals are not true, so we use Box-Cox transformation to all RPDs. Means plot and least significant difference (LSD) for all the considered algorithms are illustrated in Fig. 2. LSD intervals plotted for two given algorithms depict that the all algorithms have no statistically significant differences from other algorithms.

As it can be seen from Fig. 3 and Fig. 4, the inspection period duration and PM time affected performance of algorithms related to others. However, the performance of GA is superior related to other algorithms and difference of RPD means GA, SA and NEH1 are statistically sinificant in all levels of inspection periods and PM times.

6.1.2 Resumable case

The average relative percentage over 540 test problems for the three heuristics GA, SA and NEH1 wih resumable case is shown In Tab. 2. As it can been seen, the GA reaches better results related to other algorithms in all instances. Also SA has better performance than NEH1.

Means plot and least significant difference (LSD) for all the considered algorithms is illustrated in Fig. 5. LSD intervals plotted for two given algorithms depict that the all algorithms have no statistically significant differences from other algorithms.

As it can be seen from Fig. 6 and Fig. 7, the inspection duration and PM time affected performance of algorithms related to others. However, the performance of GA generares superior results related to other algorithms and means difference of this algorithms with others in all considered levels are statistically sinificant.

Table 1. Average RPD value for the three alghorithms grouped by n and m in non resumable case

Instance	GA	SA	NEH1
20×20	0.29	1.96	7.45
20×30	0.27	2.11	7.58
20×40	0.15	2.97	5.18
30×20	0.10	3.31	6.94
30×30	0.18	2.70	6.27
30×40	0.05	2.96	5.02
40×20	0.11	3.69	7.28
40×30	0.00	3.75	5.27
40×40	0.06	3.05	4.31
average	0.13	2.94	6.20

Table 2. Average RPD value for the three alghorithms grouped by n and m in resumable case

instance	GA	SA	NEH1
$20\times$ 20	0.44	2.26	3.60
$20\times$ 30	0.36	2.57	3.51
$20\times$ 40	0.79	3.01	3.00
$30\times$ 20	0.16	3.24	4.69
$30\times$ 30	0.36	3.45	4.11
$30\times$ 40	0.52	3.06	2.98
$40\times$ 20	0.06	3.64	4.98
$40\times$ 30	0.35	2.63	3.71
$40\times$ 40	0.63	2.51	2.93
average	0.41	2.93	3.72

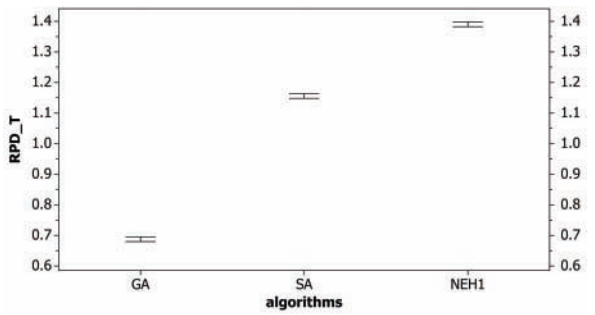

Fig. 2. Mean plot and LSD intervals (at the 95% confidence) of algorithms for non resumable case

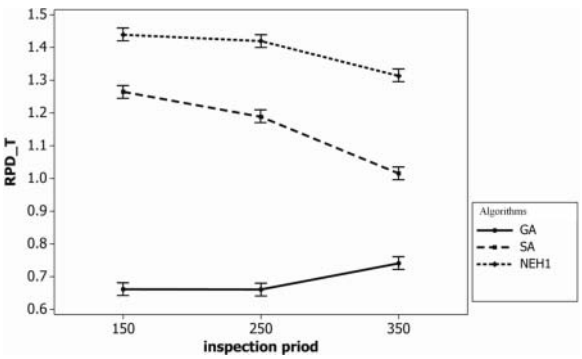

Fig. 3. Mean plot and LSD intervals (at the 95% confidence) for interaction of the algorithms and inspection periods for non reumable case

In inspectin period fator, the results of NEH1 when the inspection period increases, is been better such that reaches to RPD mean of SA in 350. Also when PM duration increases, the RPD means is been worse related to NEH1 and mean difference is statistically sinificant.

Table 3. Time results

Instance	Time results for SA	Time results for GA	time results for NEH1
$20\times$ 20	05:38	05:04	00:01
$20\times$ 30	08:13	07:24	00:01
$20\times$ 40	11:34	10:25	00:01
$30\times$ 20	08:42	07:49	00:02
$30\times$ 30	13:32	12:11	00:02
$30\times$ 40	17:30	15:45	00:03
$40\times$ 20	12:25	11:11	00:04
$40\times$ 30	18:21	16:31	00:05
$40\times$ 40	25:25	22:52	00:07

6.1.3 Time results

Tab. 3 presents the time results for heuristics in which time needed to run heuristics on instances has been given. The Elements of this table is considered as (minuts: second). Form this table it can been seen time needed to run NEH1 algorithm for an instance is less than10 second for all instances, also these times less than other algorithms run time. NEH1 can be substitute for other algorithms when there is no enough time to run them. GA algorithm has running time more than NEH heuristics and less than SA heuristics.

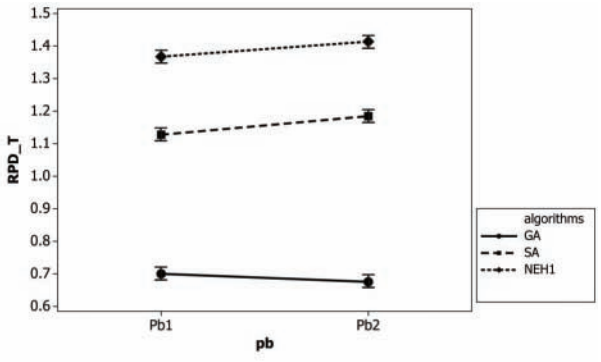

Fig. 4. Mean plot and LSD intervals (at the 95% confidence) for interaction of the algorithms and PM time for non resumable case

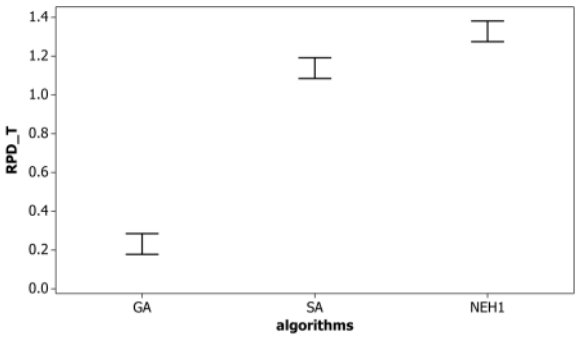

Fig. 5. Mean plot and LSD intervals (at the 95% confidence) of algorithms for non resumable case

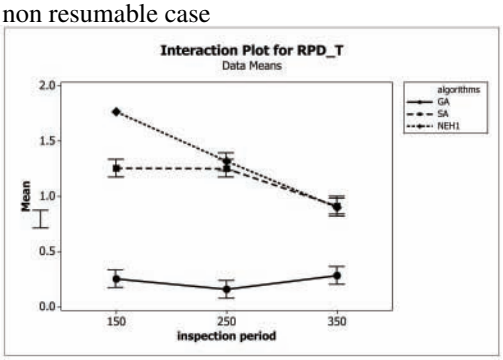

Fig. 6. Mean plot and LSD intervals (at the 95% confidence) for interaction of the algorithms and inspection periods for non reumable case

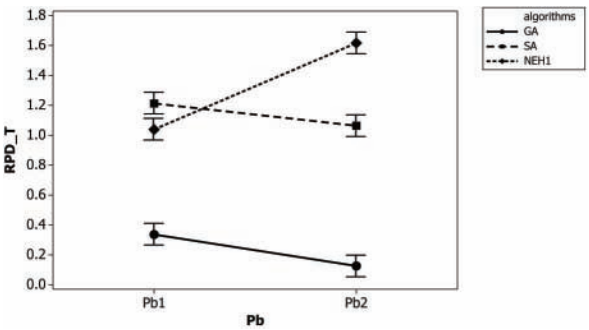

Fig. 7. Mean plot and LSD intervals (at the 95% confidence) for interaction of the algorithms and PM time for non resumable case

7 Conclusion

In this paper, we had studied the problem of scheduling a flowshop to minimize expected makespan based on condition based maintenance constraints. In fact many real life industry problems represent the double complexity (algorithmic and structural-functional). We have illustrated the approach of combining the simulation and the optimization to deal with this problem. NEH1, SA and GA and a simulator were used to construct this approach. These heuristics were applied to set of 1080 instances. The obtained results strongly manifest the superiority of GA over others in both cases resumable and non-resumable. Also ANOVA's analysis was done to denote statistically validation of results. In all levels of factors, GA achieved better results with respect to other algorithms. Also the results denoted that inspection period and PM time affected the performance of algorithms, thus it is necessary to explore PM constraint on flowshop problems. Also best and worst Time results belonged to NEH1 and SA, respectively.

References

[1] A. Allahverdi. Scheduling on m parallel machines subject to random breakdowns to minimize expected mean flow time. *Naval Research Logistics Quarterly*, 1994, **41**: 677–682.

[2] A. Artiba, E. Aghezzaf. Architecture of a multi-model system for planning and scheduling. *Journal of Computer Integrated Manufacturing*, 1997, **10**(5): 380–393.

[3] J. Barata, C. Soares, et al. Simulation modelling of repairable multi-component deteriorating systems for 'on condition' maintenance optimization. *Reliability Engineering and System Safety*, 2002, **76**: 255–264.

[4] J. Birge, J. Frenk. Single-machine scheduling subject to stochastic breakdowns. *Naval Research Logistics Quarterly*, 1990, **37**: 661–677.

[5] J. Blazewicz, J. Brei. Heuristic algorithms for two-machine flowshop with limited machine availability. *Omega*, 2001, **29**: 599–608.

[6] E.Taillard. Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research*, 1990, **47**: 65–74.
[7] E.Taillard. Benchmarks for basic scheduling problem. *European Journal of Operational Research*, 1993, **64**: 278–285.
[8] K. Glazebrook. Scheduling stochastic jobs on a single machine subject to breakdowns. *Naval Research Logistics Quarterly*, 1984, **31**: 251–264.
[9] H. Allaoui, A. Artiba. Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints. *Computers & Industrial Engineering*, 2004, **47**: 431–450.
[10] A. Hoyland, M. Rausand. System reliability theory-models and statistical methods. *Wiley*, 1994.
[11] H. Ishibuchi, S. Misaki, H. Tanaka. Modified simulated annealing algorithms for the flow shop sequencing problem. *European Journal Operational Research*, 1995, **81**: 388–398.
[12] S. Johnson. Optimal two- and three-stage production schedules with set up times included. *Naval Research Logistics Quarterly I*, 1954, 61–68.
[13] S. Kirkpatrik, C. Gelatt, M. Vecch. Optimization by simulated annealing. *Science*, 1983, **220**: 671–680.
[14] C. Lee. Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint. *Operational Research Letters*, 1994, **20**: 129–139.
[15] V. Legat, A. Zaludova, et al. Contribution to optimization of preventive maintenance. *Reliab Eng Syst Saf*, 1996, **51**: 259–266.
[16] N. Metropolis, A. R. Rosenbluth, et al. Equations of state calculations by fast computing machines. *Journal of chemical physics*, 1953, **21**: 1087–1092.
[17] G. Nadoli, J. Biegel. Intelligent manufacturing simulation agents tool (imsat). *ACM Transactions on Modeling and Computer Simulations*, 1993, **3**(1): 42–65.
[18] M. Nawaz, E. Enscore, L. Ham. A heuristic for the m-machine n-job flow shop sequencing problem. *Omega II*, 1983, 91–95.
[19] C. Reeves. A genetic algorithm for flowshop sequencing. *Computers and Operations Research*, 1995, **22**: 5–13.
[20] R. Ruiz, J. Garcia-Diaz, C. Marato. Considering scheduling and preventive maintenance in the flowshop sequencing problem. *Computers & Operations Research*, 2006, **34**: 3314 – 3330.
[21] G. Schmidt. Scheduling with limited machine availability. *European Journal of Operational Research*, 2000, **121**: 1–15.
[22] A. Talbi, H. Abdulrab. Lop2: Extension du language lop1 par le paradigme object. **in:** *Proceedings of the RJCIA Conference*, 1994, 155–165.
[23] T. Nakagawa. *Shock and degradation models in reliability theory*. Springer-Verlag, 2007.
[24] J. Vaurio. On time-dependent availability and maintenance optimization of standby units under various maintenance policies. *Reliab Eng Syst Saf*, 1997, **56**: 79–89.