International Journal of Applied Operational Research
Vol. 4, No. 4, pp. 11-26, Autumn 2014

Journal homepage: ijorlu.liau.ac.ir

MILP Formulation and Genetic Algorithm for Non-permutation Flow Shop Scheduling Problem with Availability Constraints

R. Ramezanian*

Received: 20 April 2014 ; **Accepted:** 5 September 2014

Abstract In this paper, we consider a flow shop scheduling problem with availability constraints (FSSPAC) for the objective of minimizing the makespan. In such a problem, machines are not continuously available for processing jobs due to preventive maintenance activities. We proposed a mixed-integer linear programming (MILP) model for this problem which can generate non-permutation schedules. Furthermore, an improving heuristic method and a genetic algorithm (GA) based heuristic are developed to evolve optimal or near optimal solutions. To obtain better and more robust solutions, The Taguchi method is performed for tuning the parameters of genetic algorithm. The MILP model can be used to compute optimal solutions for small-sized problems or to test the performance of solution algorithms. The presented methodology is evaluated for the solution quality. According to computational experiments, the GA can reach good-quality solutions in reasonable computational time, and can be used to solve large scale problems effectively.

Keywords: Flow Shop Scheduling, Availability Constraint, Mixed-Integer Linear Programming, Improving heuristic, Genetic Algorithm, Taguchi Method.

1 Introduction

Production scheduling is one of the most important tasks carried out in manufacturing systems which can be defined as the allocation production resources over time best to satisfy a pre-set criterion and seeks goals such as optimal sequence of jobs on machines, balanced machine utilization rate, and short average customer waiting time [1].

Scheduling and maintenance planning have separately received considerable attention in operations research literature and both of domains are successfully studied from theoretical and practical view. But, combining of scheduling and availability constraints has received much less attention. In this area it is usually assumed that machines and processors are continuously available during the whole planning horizon. However, in many real situations, machines may be unavailable due to breakdowns or preventive maintenance (PM) activities [2, 3]. The primary goal of preventive maintenance actions is to prevent the failure of equipment before it actually occurs. It can retain or restore a system to an acceptable operating condition. Although PM activities take time that could otherwise be used for production, delaying them may increase the probability of machine failure [4]. Therefore, combining scheduling decisions and preventive maintenance plans is necessary to achieve a global plan for real manufacturing systems.

*Corresponding Author. (✉)
E-mail: Ramezanian@kntu.ac.ir (R. Ramezanian)

R. Ramezanian
Department of Industrial Engineering, K.N. Toosi University of Technology, Tehran, Iran.

[Downloaded from ijaor.com on 2026-11-08]

Ma et al. [5] provided a thorough survey on deterministic scheduling problems with machine availability constraints. Albers and Schmidt [6] proposed an optimal algorithm if the next point in time where the set of available machine changes is known. Scheduling and preventive maintenance problems can be classified according to the shop environment. In this area, some works are related to the single machine problem (for instance see [4, 7, 8]). Zhao et al. [9] studied two parallel machines scheduling problem with an availability constraint where only one machine is unavailable in a fixed and known time period. They presented a fully polynomial-time approximation scheme for the problem to minimize the total weighted completion time. The two-machine flow shop scheduling problem (FSSP) with availability constraint is first studied by Lee [10]. He proved that this problem with an availability constraint only on one machine is NP-hard. Kubzin and Strusevich [11] studied a two-machine flow shop no-wait scheduling problem with maintenance. They presented a polynomial-time approximation scheme to solve this problem considering makespan as an objective function. Liao and Tsai [12] proposed heuristic approaches to minimize makespan in a two-machine flow shop with availability constraints. Hadda [13] considered two machine flow shop scheduling problem with several availability constraints where only the first machine is unavailable. He presented a polynomial-time approximation scheme for the problem, under the resumable scenario, to minimize the makespan. Zhao and Tang [14] studied two-machine no-wait flow shop scheduling with deteriorating jobs and machine availability constraints with makespan minimization. They assumed that there are unavailability intervals on only one machine. Ben Chihaoui et al. [15] studied the two-machine no-wait flow-shop scheduling problem, when every machine is subject to one non-availability constraint and jobs have different release dates. They proposed a branch-and-bound algorithm to solve the problem.

In this research, we consider a general flow shop scheduling problem with availability constraints that is not restricted by the number of machines. Benbouzid-Sitayeb et al. [16, 17] considered joint production and preventive maintenance scheduling problem in permutation flow shop. They proposed several meta-heuristics such as genetic algorithm [16] and ant colony optimization [17] for solving this problem and they compared obtained results for optimality. The goal of these papers is to optimize an objective function which takes into account the criteria of maintenance and production at the same time. Ruiz et al. [18] provided tools in order to implicitly consider different preventive maintenance policies on machines in the permutation FSSP environment. The optimization criterion considered consists in minimizing the makespan. They evaluate six adaptations of heuristic and meta-heuristic methods for the consideration of preventive maintenance. Aggoune et al. [19] investigated a flow shop problem with availability constraints and proposed a heuristic based on genetic algorithm to solve the makespan and the total weighted tardiness minimization problems. They considered two variants to deal with the maintenance activities: either starting time of the maintenance tasks are fixed or the maintenance tasks must be performed on a given time window. Aggoune [2] also developed a heuristic based on genetic algorithm and tabu search for this problem to solve the makespan minimization. Later, Aggoune and Portmann [20] presented a temporized geometric method for solving the problem with two jobs. Perez-Gonzalez and Framinan [21] considered a permutation flow shop problem with the assumption of machine availability constraint at the beginning of the period. They developed a number of quick heuristics in order to solve this problem. Safari and Sadjadi [22] considered the flow shop scheduling problem with the condition-based maintenance for minimizing the expected makespan. To solve the problem, they used a hybrid algorithm based on the genetic and simulated annealing algorithms. Shoaardebilia and Fattahi [23] considered multi-

[Downloaded from ijaor.com on 2026-11-08]

objective three-stage assembly flow shop scheduling problem with machine availability constraints. They implemented multi-objective meta-heuristics are presented to solve the problem with Two performance measures including minimizing total weighted completion times and sum of weighted tardiness and earliness. Vahedi-Nouri et al. [24] investigated flow shop scheduling problem with tow practical condition including learning effects and machine availability constraints to minimiz the total flow time as a performance measure. They proposed a heuristic to solve the problem that is able to find non-permutation solutions.

Allaoui and Artiba [3] considered the two-stage hybrid flow shop scheduling problem to minimize the makespan criterion under maintenance constraints. There are only one machine on the first stage and m machines on the second stage in the considered shop. A branch-and-bound algorithm is presented and the performance of three heuristics is evaluated. Besbes et al. [25] developed an approximate approach based on genetic algorithm for hybrid flow shop scheduling problems under availability constraints with the makespan minimization as the objective function. Allaoui and Artiba [26] studied hybrid flow shop scheduling with availability constraints to minimize makespan. They developed a branch and bound algorithm to solve the two-stage hybrid flow shop. Wang and Liu [27] considered a bi-objective integrated production scheduling and preventive maintenance with non-resumable jobs in a two-stage hybrid flow shop. They considered sequence-dependent setup and preventive maintenance on the first stage machine. Due to the complexity of the problem, a multi-objective tabu search algorithm is adapted to solve the problem.

The aim of this paper is to successfully formulate integrated scheduling and availability constraints for the flow shop problem. We present a MILP model for this problem that can generate non-permutation schedules. A GA based heuristic is also proposed to evolve optimal or near optimal solutions. It is possible to obtain optimal solutions for small-sized problems using the MILP model using optimization solver. In addition, it helps to test the performance of presented meta-heuristic algorithm by comparing the results.

The remainder of the paper is organized as follows: In section 2 we present proposed MILP model for FSSP with availability constraints. Section 3 contains the scheduling approach to determine sequence of jobs and PM tasks. Section 4 describes the genetic algorithm for solving our proposed model. Section 5 presents the computational results acquired and, finally, Section 6 provides conclusions and suggestions for further researches.

2 Mathematical formulation of FSSP with availability constraints

In this section, a proposed MILP for FSSPAC is presented. A n-job, m-machine, L_i-PM Task non-permutation FSSPAC with minimized makespan (C_{max}) as the objective can be presented using the classical notation $F|availability\ constraints|C_{max}$ [28]. In the flow shop scheduling problem (FSSP), there are a set of jobs that have to be processed on a number of sequential machines. Each job has to be processed on all machines and the processing routes of all jobs are the same i.e., the operations of any job are processed in the same order. The job sequence of each machine has to be identified to minimize (or maximize) a specific performance measure (usually minimizing the makespan or total flow time). In the permutation FSSP, all jobs must enter the machines in the same order, while in non- permutation (general) FSSP, jobs can have different sequence on each machine.

Hypotheses considered in this paper are summarized as follows:

- All n jobs to schedule are independent.
- Jobs have no associated priority values.

[Downloaded from ijaor.com on 2026-11-08]

- One machine can process at most one job or PM task at a time.
- Each job is processed on at most one machine at a time.
- No more than one operation of the same job can be executed at a time.
- Scheduled maintenance is allowed (Machine is not available at all times).
- Each maintenance task has a predefined time window, in which the completion time of the task can be moved within (Each PM task has to be completed within its pre-specified time window).
- When a PM task is performed on a machine, no operation can be processed on that machine.
- Processing of each job cannot be started before its release time.
- The setup time for the operations is sequence-independent and is included in the processing time
- Preemption and splitting of any particular job is not allowed: a job, once started on the machine, continues in processing until it is completed.
- There is no travel time between stages; jobs are available for processing at a stage immediately after completing processing at the previous stage.
- Jobs are allowed to wait between two stages, and the storage is unlimited.
- All programming parameters are deterministic and there is no randomness.
- There is only one of each type of machine.

The notation used in this paper is summarized in the following:

Indices

i: Index of machines, $i = 1, 2, \ldots, m$;
j,h: Job index of jobs, $j,h = 1, 2, \ldots, n$;
l: Index of maintenance tasks, $l = 1, 2, \ldots, L_i$;

Parameters

n: Total number of jobs;
m: Total number of machines;
L_i: Total number of preventive maintenance tasks on machine i;
PM_{il}: The lth preventive maintenance task on machine i;
t_{ij}: Processing time of job j on machine i;
p_{il}: Duration of the maintenance task PM_{il};
R_j: Release date of job j;
M: Large constants ($M \rightarrow \infty$);
EM_{il}: The early completion time of the maintenance task PM_{il};
LM_{il}: The late completion time of the maintenance task PM_{il};

Decision variables

S_{ij} : Starting time of job j on machine i;
C_{ij} : Completion time of job j on machine i;
FM_{il}: Completion time of the maintenance task PM_{il};
Y_{ihj}: A binary variable that is equal to 1 if job j is processed after job h when processing on machine i, 0 otherwise;
Z_{ijl}: A binary variable that is equal to 1 if job j is processed before maintenance task l when processing on machine i, 0 otherwise;

In this study, we manage to minimize the makespan criterion (C_{max}). Proposed model is given as follows:

[Downloaded from ijaor.com on 2026-11-08]

$$Min\ z = C_{\max} \quad (1)$$

$$C_{\max} \geq C_{mj}; \quad \forall j \quad (2)$$

$$C_{ij} = S_{ij} + t_{ij}; \quad i = 2,...,m, j = 1,..,n \quad (3)$$

$$S_{(i+1)j} \geq C_{ij}; \quad i = 1,...,m-1, j = 1,..,n \quad (4)$$

$$S_{ij} \geq C_{ih} - (1 - Y_{ihj})M; \quad \forall i, j, h \quad (5)$$

$$S_{ih} \geq C_{ij} - Y_{ihj} M; \quad \forall i, j, h \quad (6)$$

$$C_{ij} - t_{ij} - FM_{il} + Z_{ijl} M \geq 0; \quad \forall i, j, l \quad (7)$$

$$FM_{il} - p_{il} - C_{ij} + (1 - Z_{ijl})M \geq 0; \quad \forall i, j, l \quad (8)$$

$$EM_{il} \leq FM_{il} \leq LM_{il}; \quad \forall i, l \quad (9)$$

$$S_{1j} \geq R_j, \quad \forall j \quad (10)$$

$$Y_{ijh}, Z_{ijl} = \{0,1\}, \quad \forall i, j, h, l \quad (11)$$

The objective function (1) considers the minimization of the makespan. The constraint set (2) ensures that the makespan is equal to the maximum completion time of any job. The constraint set (3) corresponds to the computation of the completion time of job. The constraint set (4) forces to start the processing of each job only when it has been completed on the precedent machine. The constraint sets (5) and (6) force to start the processing of each job only when its precedent job has been completed on the same machine. Preventive maintenance is a set of preplanned actions performed to prevent the potential failure of equipments before it actually occurs. On each machine if a job is processed before/after a maintenance task then the finish/start time of that job must be less/greater than the maintenance start/finish time which are mentioned in the constraint sets (7) and (8). The constraint set (9) ensures that a maintenance task is performed in the corresponding time window. The constraint set (10) bounds the job starting times to be after job release times in the system. The relation (11) is a logical constraint.

Consider a FSSP with two machines, four jobs and two maintenance tasks on machine 1 and one maintenance task on machine 2. The processing times of jobs on machines and other required data are given in Table 1. The optimal sequence of jobs and maintenance tasks on machines for minimizing the makespan is illustrated in Figure 1.

Table 1. Processing times, PM times and other data

		Job				Preventive maintenance					
		1	2	3	4	p_{i1}	p_{i2}	EM_1	LM_1	EM_2	LM_2
Machine	1	3	5	6	4	2	1	5	7	19	21
	2	5	4	6	3	2	--	20	22	--	--
Release time		2	7	5	4						

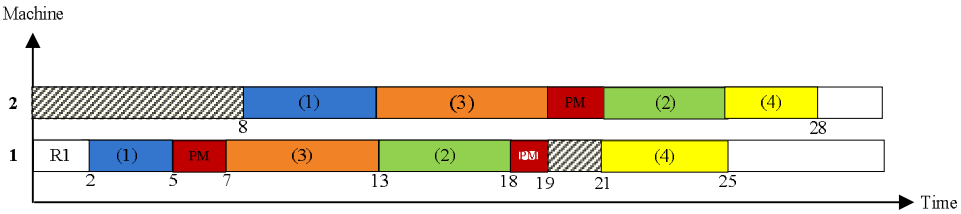


Fig. 1 Optimal solution for data set of Table 1 (Makespan=28)

[Downloaded from ijaor.com on 2026-11-08]

3 Scheduling approach

There are two strategies to plan joint production scheduling and preventive maintenance in the literature [29]. The first strategy consists of two steps: First the scheduling of the production jobs then the insertion of the maintenance tasks. The second one consists of simultaneously scheduling both maintenance and production activities.
To schedule job and maintenance sequencing an approach similar to the first strategy is adapted in this paper. First, maintenance tasks are assigned to the corresponding time windows then jobs are inserted in the free periods of time between maintenance tasks. It is due to the fact that maintenance tasks must be performed in a given time window.
To assign a maintenance task in a given time window, three ways could be used:

- The late completion time of the maintenance task PM_{il} is considered as the completion time of PM_{il} ($FM_{il} = LM_{il}$).
- The early completion time of the maintenance task PM_{il} is selected for the completion time of PM_{il} ($FM_{il} = EM_{il}$).
- Completion time of the maintenance task PM_{il} is assigned in a given time window, randomly.

When L_i maintenance tasks are assigned to machine i, planning horizon of machine i is decomposed on (L_i+1) subintervals. In this paper, to schedule a joint production and preventive maintenance problem, first, completion time of maintenance tasks are fixed on machines using one of the above-mentioned methods, then all jobs are assigned to the feasible subintervals according to the predefined job sequence. Feasible subintervals on a machine for assigning a job is determined with considering the precedence constraint and also the fact that the length of a subinterval have to be greater than processing time of the job on the machine. Note that after each insertion the length of the concerned subinterval is updated. For instance, consider job j to be assigned in the subinterval of $PM_{i,l+1}$ and PM_{il} and job j must be processed immediately after job h on machine i. The subinterval is feasible, if $\{min\ (FM_{i,l+1} - p_{i,l+1} - FM_{il},\ FM_{i,l+1} - p_{i,l+1} - C_{ih},\ FM_{i,l+1} - p_{i,l+1} - C_{i\text{-}1,j}) \geq t_{ij}\}$. If the constraint is not satisfied, next subinterval is checked until a feasible one is found. Finally, when the sequence of all tasks is determined, the maintenance tasks and operations are left-shifted as much as possible.

For example, let us consider example of section 2. Assume that $\{1, 3, 2, 4\}$ is the job sequence on two machines and first method of PM assigning ($FM_{il} = LM_{il}$) is selected. The procedure of scheduling approach is illustrated in Figures 2 and 3.

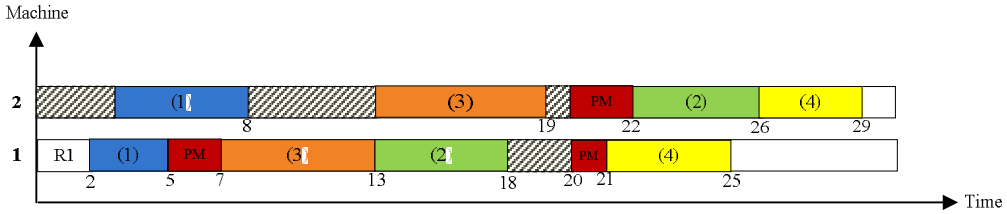


Fig. 2 Solution for data set of Table 1 before using left-shifting (Makespan=29)

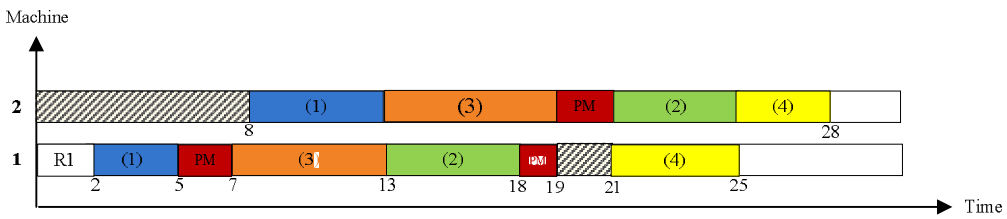


Fig. 3 Solution for data set of Table 1after using left-shifting (Makespan=28)

[Downloaded from ijaor.com on 2026-11-08]

4 The solution methods

Since the problem is strongly NP-hard, solving it with exact methods such as branch and bound will be costly and time consuming. For the small size problems, the presented MILP model is solved using the Lingo software. In order to obtain close-to-optimal solutions at a reasonable time, an improving heuristic method and a genetic algorithm are presented.

4.1 The proposed heuristic method

The proposed heuristic is an improving method that is begun with a good permutation solution and this solution will be improved in the next steps with obtaining a non-permutation schedule, hereafter called NP-heuristic. The steps of the NP-heuristic are described in Figure 4.

The procedure of NP-heuristic

1. The finish time of all maintenance activities on machines is considered to the latest possible completion time.
2. An initial solution with permutation structure is generated by NEH heuristic [30] with the consideration of maintenance activity and the objective function of this solution is considered as the best known solution.
3. **For** i = 1 to m
   For j = 1 to n
   For k=1 to n & $k \neq j$
   The start time of every maintenance activity is shifted as much as possible toward the completion time of the last job processed before it.
   The objective function of this solution is calculated and placed in set Π.
   The best solution obtained in set Π is compared and replaced with the best known solution if its objective value is less than the objective value of the best known solution.
   End
   End
   End

Return the best known solution.

Fig. 4 The procedure of the NP-heuristic

4.2 The Genetic Algorithm

The mathematical model formulated for the determination of job sequence to the FSSP with availability constraints (Section 2) belongs to mixed-integer linear programming (MILP) problem. A GA is proposed to evolve an optimal or near optimal solution for minimum makespan to the FSSP with availability constraints model. Genetic algorithm has been proven to be powerful optimization technique for constrained optimization and combinatorial

[Downloaded from ijaor.com on 2026-11-08]

optimization problems [31, 32]. In order to obtain better and more robust results, the Taguchi method is used for tuning the GA's parameters. Our implementation of GA based heuristic is presented as follows.

Design of Genes

Each gene is a job or PM task and the chromosome is a job and PM sequence vector on machines. The genotype representation of the chromosome and the physical meaning of them are outlined in Figure 5.

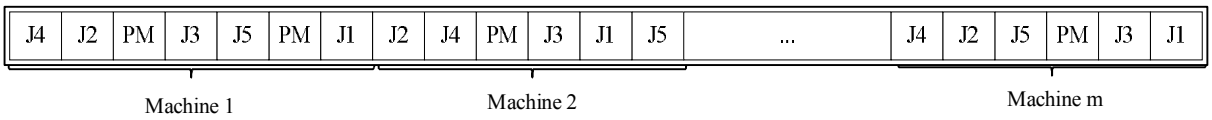


Fig. 5 Chromosome representation

Initial population

To make the initial population, first the finished time of all PM tasks are specified (assigned to the time windows) through one of the 3 above mentioned ways (Section 3) according to a given probability (0.1, 0.1 and 0.8, respectively). This procedure is repeated as many times as the number of initial population, so a set of chromosomes, which their PM task positions are fixed, is produced. Then, the sequence of jobs is generated using three methods. 10% of job sequences of initial population are obtained by NEH procedure [30]. NEH heuristic's schedule is used in the generation of initial population since the NEH schedule can generate suboptimal solution rapidly. 30% by Random Permutation Job Sequence (RPJS) and the rest of them are achieved by Random Non-Permutation Job Sequence (RNPJS). The diversity of the initial population can be retained due to the fact that a majority of job sequences are generated randomly.

Finally, each job sequence is matched to a member of PM task set and jobs are inserted in the free periods of time between maintenance tasks.

Selection

The selection provides the opportunity to deliver the gene of a good solution to next generation. There are various selection operators available that can be used to select the parents. In this study, the roulette wheel selection is employed.

The Genetic Operators

Crossover

Crossover is a process in which chromosomes exchange genes through the breakage and reunion of two chromosomes. Offspring of crossover should represent solutions that combine substructures of their parental solutions. In this study the order crossover is chosen [33]. We should note that crossover procedure is done on job sequence.

Mutation

Mutation operator generates an offspring by randomly modifying the parent's feature. Two mutation operators are chosen:

1. Choose one chromosome randomly and then choose two priorities on machines of the selected chromosome randomly, now replace selected jobs with each other.

[Downloaded from ijaor.com on 2026-11-08]

2. Choose one chromosome randomly and then choose two priorities on machines of the selected chromosome randomly, now exchange sequence of jobs there are between two selected priorities.

In our implemented GA, two mutation operators are used to ensure that the diversity can be enhanced and the search region can be extended.

Fitness Function

The fitness function is the same as the objective function which is defined in Section 2, namely makespan. In the presented GA the lower fitness function is desired.

Termination condition

The search process stops if the number of iterations is greater than maximum number of generations, a priori fixed constant.

5 Computational results

In order to evaluate the performance of the GA proposed in the previous section, experiments are performed on randomly generated instances. Furthermore, for the smaller size problems of FSSPAC (i.e., the number of machines and jobs are small), LINGO Optimization solver is used to find out the optimal solution and compared with the corresponding GA results. Due to the related problem belongs to NP-hard problems, the computational time rises exponentially as either the number of variables and constraints increase. Therefore, such exact solvers are limited by the problem size and cannot be used to solve medium and large problem instances.

In this section some results for several instances is reported. We generate random problem instances for number of jobs and number of machines from 2×2 to 20×20. Job processing times on each machine are drawn from discrete uniform distribution in the interval [1–100]. We consider two maintenance tasks on each machine for large problems ($n \geq 10$) and reduce it to one task for smaller ones. These tasks are generated randomly for each machine. The maintenance tasks occur after at least one operation and before at least one operation. The duration of a maintenance task on a machine is the average of the processing times of operations on this machine and a maximum shifting of 50 time units to each time windows maintenance task is allowed.

The genetic algorithm and developed heuristic are coded in MATLAB R2007(b) and all tests are conducted on a PC at Core Due 2 GHz with 1.0 GB of RAM.

5.1 Taguchi experimental design

In this paper, the Taguchi method is used to calibrate the parameters of meta-heuristic algorithm. This method has been presented by Taguchi in early 1960s, and it can be used in the designing of processes. The orthogonal arrays of the method are used for the study of a large number of factors with a few experiments. In this method, there are two classes of factors including controllable factors and noise factors in which the first one and second one will be placed in inner and outer orthogonal array, respectively. The noise factors are the factors that cannot be controlled. The measured values of quality characteristics obtained from the experiments will be converted to signal/noise ratio (S/N). The S/N ratio is used to determine the optimal parameter level combinations. The Taguchi method seeks to minimize

[Downloaded from ijaor.com on 2026-11-08]

variances of quality characteristics resulted from S/N ratio, which it is the reason of that parameter design is also called robust design [34].

Designing procedure of Taguchi can be explained as follows [35]:

- The influences of the controllable factors are evaluated over the S/N ratio and mean of response.
- For the factors that have significant impact on the S/N ratio, the levels that increase the S/N ratio will be selected.
- Each factor that does not have any significant impact on S/N ratio and has significant impact on mean of responses, the level where it's mean of response is closer to objective point will be selected.
- Factors with no significant impact either on S/N ratio or on mean of response is considered as economical factors and levels that decrease cost of computation will be selected.

Quality characteristic of this paper is relative percentage deviation (RPD), which prefers "the smaller-the better" type. The S/N ratio for "the smaller-the better" characteristic is determined by the formula is presented below [36]:

$$\eta_j = -10\log\left(\frac{1}{n}\sum_{i=1}^{n} y_i^{\,2}\right) \tag{12}$$

where j, y_i and n denote the trial number, response variable and the number of replications, respectively.

GA parameter tuning

The factors that can have significant effects on the genetic algorithm and their proper levels are presented in Table 2. Initial trials reveal that the levels considered for the factors in Table 2 produce better results in comparison with other values.

Table 2 GA's factors and their considered levels

Factor	Notation	level	Value
Population size	Popsize	3	100, 150, 200
Crossover rate	P_c	3	0.75, 0.8, 0.85
Rate of first mutation	P_{m1}	3	0.025, 0.05, 0.075
Rate of second mutation	P_{m2}	3	0.025, 0.05, 0.075
Number of iterations	NI	3	250, 500, 750

3^5=243 experiments are required for the full factorial design in GA. According to the computational time and cost, this kind of experimental design is not economical. As mentioned above, the fractional design (Taguchi method) is used instead of full factorial design. For the selection of a suitable orthogonal array, degrees of freedom should be calculated. The proper orthogonal array is $L_{27}(3^5)$. The selected Taguchi design for GA has 27 different level combinations of control factors. For every trial, five random instances with different size are considered and each of them is replicated three times in order to obtain more reliable results. The relative percentage deviation (RPD), as a common response variable, is used to compare the methods. RPD is calculated as follows:

$$RPD = \left(\frac{OV_i - OV_{\min}}{OV_{\min}}\right) \times 100 \tag{13}$$

where OV_{min} and OV_i are the best found objective value for a particular instance and the objective value obtained for the ith trial, respectively.

[Downloaded from ijaor.com on 2026-11-08]

The RPD value is transformed into the S/N ratio. The mean S/N ratio and mean RPD value is calculated for each level of control factors and they are plotted versus the control factors in Figures 6 and 7, respectively. To test the statistical significant of control factors, the analysis of variance (ANOVA) is carried out. Tables 3 and 4 show the analysis of variance associated with mean S/N ratio and mean RPD, respectively.

According to Table 3, the factors of second mutation rate (P_{m2}) and number of iterations (NI) have significant impact on the S/N ratio. Therefore, these factors are considered as control factors and based on Figure 6, 0. 05, 500 with highest values of S/N ratio are selected as the optimal value for them, respectively. Also Table 4 shows that the factors of crossover rate (P_c) and second mutation rate (P_{m2}) have significant impact on the RPD values. Therefore, according to these tables, factor crossover rate is considered as the adjustment factors and based on Figure 7, the levels with lowest values of mean RPD (0.85) are considered as the optimal values of this factor. The factors of population size (Popsize) and first mutation rate (P_{m1}) neither have a significant effect on the S/N ratio nor on the mean RPD values. Thus, they are recognized as economical factors and a level of them which results in shorter computation time or higher quality solution is selected. Accordingly, the optimal value of each factor is shown in Table 5.

Table 3 ANOVA for S/N ratio in GA

Source	df	SS	MS	F	P
Popsize	2	4.56	2.281	0.02	0.983
P_c	2	526.36	263.181	1.99	0.188
P_{m1}	2	121.64	60.819	0.46	0.645
P_{m2}	2	882.79	441.393	3.33	0.078
NI	2	1062.24	531.122	4.01	0.053
Residual Error	10	1325.61	132.561		
Total	20	3923.2			

Table 4 ANOVA for mean RPD

Source	df	SS	MS	F	P
Popsize	2	0.6447	0.3223	0.89	0.439
P_c	2	2.5548	1.2774	3.55	0.069
P_{m1}	2	0.5378	0.2689	0.75	0.499
P_{m2}	2	2.9336	1.4668	4.07	0.051
NI	2	0.7937	0.3969	1.1	0.37
Residual Error	10	3.603	0.3603		
Total	20	11.0676			

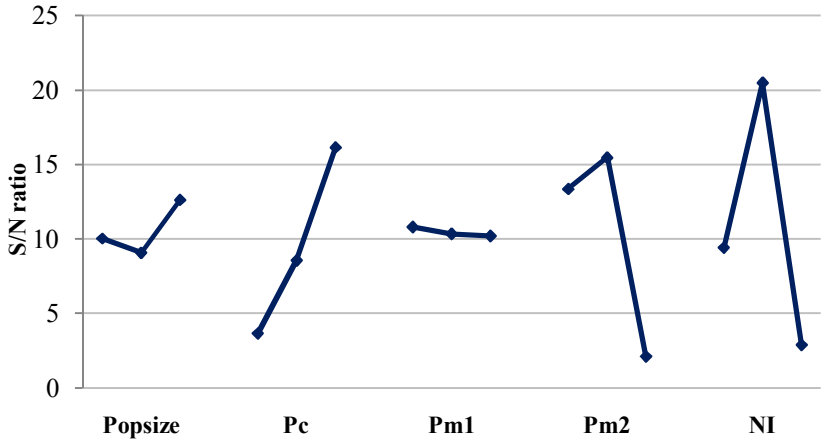


Fig. 6 The mean S/N ratio plot at each level for objective function values in GA

[Downloaded from ijaor.com on 2026-11-08]

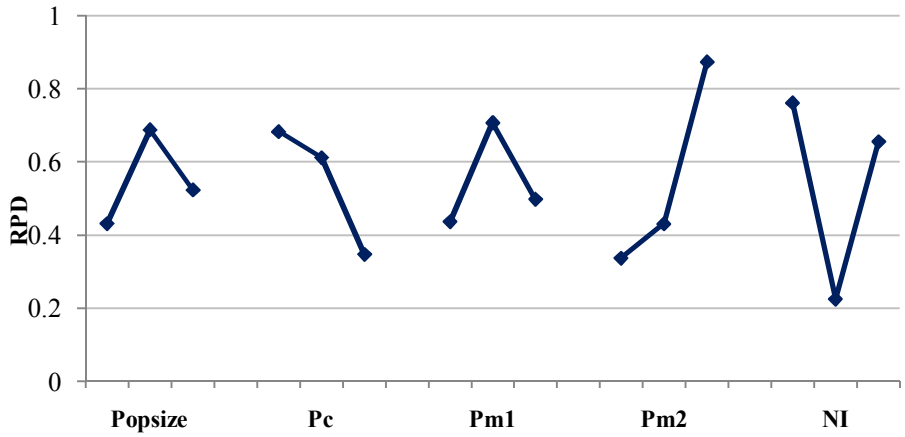


Fig. 7 Main effects plot for RPD values for algorithm calibration

Table 5 The optimal level of factors

Factor	Notation	Optimal level
Population size	Popsize	100
Crossover rate	P_c	0.85
Rate of first mutation	P_{m1}	0.025
Rate of second mutation	P_{m2}	0.05
Number of iterations	NI	500

5.2 Experimental results

To evaluate the effect of flexible availability constraints against constant availability constraints on objective functions, five instances of problem size ($m.n.L$) = (3.3.1) is examined with more details and the obtained results are shown in Tables 6. As mentioned in constraint set (9), completion time of maintenance task on each machine has to plan in predefined time window. For constant condition, two cases are considered. Completion time of maintenance task on each machine is equal to the pre-specified early completion time ($FM_{il} = EM_{il}$) or late completion time ($FM_{il} = LM_{il}$).

Table 6 Obtained results for flexible or constant availability constraints in problem size m=3, n =3 and L=1

No.	Flexible maintenance		$FM_{il} = EM_{il}$			$FM_{il} = LM_{il}$		
	Makespan	Time (s)	Makespan	Time (s)	Gap	Makespan	Time (s)	Gap
1	**330**	8	364	3	10.30%	350	2	6.06%
2	**350**	4	367	7	4.86%	356	1	1.71%
3	**302**	3	361	9	19.54%	320	3	5.96%
4	**297**	7	341	15	14.81%	312	5	5.05%
5	**263**	6	272	3	3.42%	292	2	11.03%
Average	308.4	5.60	341.0	7.40	10.59%	326.0	2.60	5.96%

[Downloaded from ijaor.com on 2026-11-08]

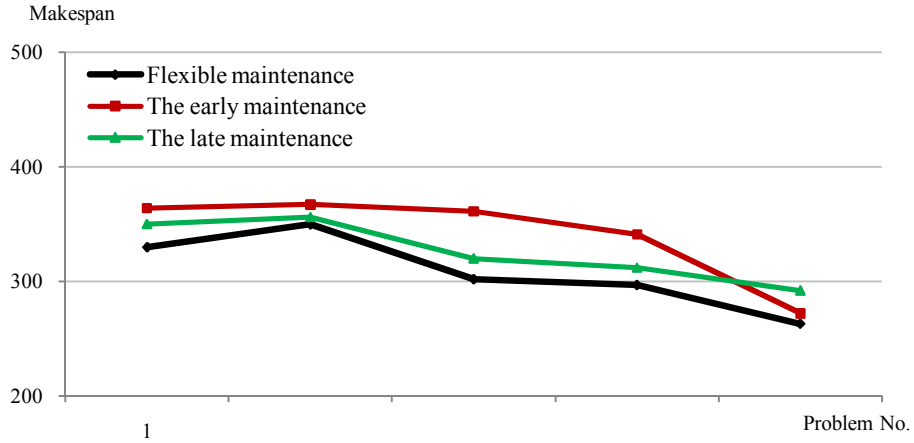


Fig. 8 Comparative result in flexible or constant maintenance conditions for the problem size (*m.n.L*)=(3.3.1)

Tables 6 and Figure 8 show that when flexible maintenance activity was considered in the model the better results obtained for makespan minimization.

The comparison for smaller size problems between the GA, modified SPT (M-SPT), modified NEH (M-NEH), NP-heuristic and LINGO is listed in Table 7.

Table 7 Comparison of exact solution with M-SPT, M-NEH, NP-heuristic and GA for small-sized problem

No.	Size (m.n.L)	Solver (LINGO8)		M-SPT		M-NEH		NP-heuristic		GA	
		C_{max}	Time (s)	RPD	Time (s)	RPD	Time (s)	RPD	Time (s)	RPD	Time (s)
1	(2.3.1)	**326**	1	16.90	0.030	5.61	0.055	0.28	0.098	**0.00**	0.72
2	(2.4.1)	**416**	46	18.25	0.025	10.58	0.047	6.49	0.087	**0.00**	0.96
3	(2.5.1)	**446**	122	24.57	0.025	12.56	0.052	1.12	0.099	1.12	1.56
4	(3.3.1)	**374**	4	28.61	0.024	1.60	0.046	0.16	0.085	**0.00**	0.98
5	(3.4.1)	**487**	666	11.21	0.030	3.57	0.054	**0.00**	0.108	**0.00**	1.22
6	(4.2.1)	**458**	2	8.52	0.024	8.52	0.044	2.66	0.078	**0.00**	0.91
7	(4.3.1)	**390**	7	12.03	0.025	11.03	0.047	1.28	0.092	1.28	1.22
8	(4.4.1)	**474**	3600	8.21	0.024	3.10	0.054	0.00	0.113	**0.00**	1.58
9	(5.2.1)	**462**	4	13.66	0.025	3.92	0.053	2.38	0.088	**0.00**	1.23
10	(5.3.1)	**512**	57	4.63	0.026	1.70	0.051	**0.00**	0.099	**0.00**	1.56
Average		**434.5**	450.9	14.66	0.026	6.22	0.050	1.44	0.095	0.24	1.20

The values in RPD's column are the difference between the objective values of the solution methods against the exact solution.

Comparing the CPU times of exact solution in the fourth column of Table 7 confirms that computation time grows exponentially by increasing the dimension of the problem. According to this table, the average computational time for problems with (m = 3, n = 3, L = 1) is 4s and the average computational time for problems with (m = 3, n = 4, L = 1) is 666s. It means that by adding one level to number of jobs, the average computational time increases more than 166 times. The RPD columns depict the gap percent between makespan obtained by M-SPT, M-NEH, NP-heuristic and GA against the optimal solutions. Table 7 confirms the advantages of the NP-heuristic in comparison to the SPT and NEH methods. Thus, the NP-heuristic is used to compare performance of GA for large-size problems. As can be seen in Table 7, the proposed GA based heuristic provides solution the same or close to optimal solution obtained through LINGO. The computational times elapsed to solve the small-sized problem also are much less than LINGO.

[Downloaded from ijaor.com on 2026-11-08]

For large-sized instances, Table 8 shows the makespan and CPU time in second for M-SPT, M-NEH, NP-heuristic and GA. "Best C_{max}" and "Mean C_{max}" columns show the best and the mean makespan of five independent runs for each instance using GA, respectively.

Table 8 Makespan and run time obtained by the solution methods for the test problems

Prob.	*m*	*n*	*L*	M-SPT		M-NEH		NP-heuristic		GA		
				C_{max}	Time (s)	C_{max}	Time (s)	C_{max}	Time (s)	Best C_{max}	Mean C_{max}	Time (s)
1	5	5	1	775.3	0.048	725.5	0.024	691.0	0.146	**648**	653.6	35.01
2	5	10	2	1345.2	0.065	1262.5	0.025	1163.0	0.620	**1063**	1080.8	64.51
3	5	20	2	1958.2	0.192	1890.0	0.0	1797.0	4.664	**1649.8**	1666.8	121.04
4	10	5	1	1134.4	0.057	1061.8	0.027	895.4	0.373	**828.9**	855.7	67.59
5	10	10	2	1661.9	0.089	1541.2	0.026	1452.9	2.587	**1306**	1335.0	132.14
6	10	20	2	2768.8	0.375	2581.1	0.030	2460.0	18.524	**2385**	2416.2	259.11
7	20	5	1	2637.2	0.085	2549.6	0.030	2531.1	1.067	**2380**	2396.7	158.68
8	20	10	2	3170.4	0.136	3009.0	0.030	2752.0	8.897	**2700**	2723.7	275.83
9	20	20	2	4116.7	0.731	3810.5	0.0367	3696	74.318	**3587.5**	3610.7	592.71
Average				2174.2	0.197	2047.9	0.028	1937.6	12.355	**1838.7**	1859.9	189.62

As Table 8 shown, among of the solution methods, GA for the related problem is better than the other algorithms. Moreover, the superiority of implemented GA is concluded over other heuristics including M-SPT, M-NEH and NP-heuristic due to the computational results.

The convergence of the GA to optimal solution is studied by plotting the minimum fitness value for every generation against the generation number. One such curve for the 10.10.2 problem size is shown in Figure 9.

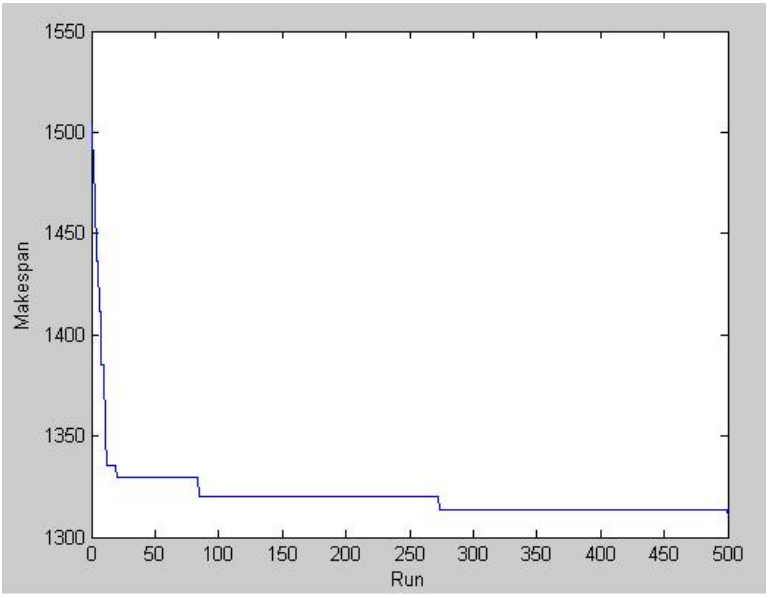


Fig. 9 GA convergence for the problem size *m*=10, *n* =10 and *L*=2

6 Conclusion and future work

This paper presents a mixed integer linear programming formulation for flow shop scheduling problem with availability constraints which is often occurring in shop of real world. The schedules generated by this model are not restricted by permutation constraint. The MILP

[Downloaded from ijaor.com on 2026-11-08]

Model could be used to compute optimal solutions for small-sized problems by optimization solvers. However, as the size of problems increased, it is difficult or impossible to reach optimal solution using these solvers. Therefore, a genetic algorithm is implemented to solve FSSP with availability constraints. To verify the effectiveness of the presented approach, computational experiments are performed. Comparing the results of the proposed GA with LINGO for the small-sized problems shows that the GA can reach optimal and in some cases near to optimal solutions in short computational time. Also for large size instances, results of the GA are compared with some modified classical heuristics such as M-SPT and M-NEH and a developed heuristic that can generate non-permutation schedules. The superiority of implemented GA is concluded over these heuristics due to the computational results.
It may be interesting to extend the MILP model with considering other assumptions such as sequence-dependent setup times, transportation constraints, and reentrant operations. Further research also can be used other meta-heuristic algorithms to solve the problem such as tabu search (TS), simulated annealing (SA), and particle swarm optimization (PSO). Hybrid algorithms should be developed by using a local search algorithm within a GA that is after generating an offspring, the solution should be improved by applying for instance TS or SA before applying the selection criterion of GA.

References

1. Yang, W. H., Liao, C.J., (1999), Survey of scheduling research involving setup times, International Journal of Systems Science, 30 (2), 143-155.
2. Aggoune, R., (2004), Minimizing the makespan for the flow shop scheduling problem with availability constraints, European Journal of Operational Research, 153, 534–543.
3. Allaoui, H., Artiba, A., (2006), Scheduling two-stage hybrid flow shop with availability constraints, Computers & Operations Research, 33(5), 1399–1419.
4. Cassady, C. R., Kutanoglu, E., (2005), Integrating preventive maintenance planning and production scheduling for a single machine, IEEE Transactions on Reliability. 54(2), 304–309.
5. Ma, Y., Chu, C., Zuo, C., (2010), A survey of scheduling with deterministic machine availability constraints, Computers and Industrial Engineering, 58 (2), 199-211.
6. Albers, S., Schmidt, G., (2001), Scheduling with unexpected machine breakdowns, Discrete Applied Mathematics, 110(2–3), 85–99.
7. Cassady, C. R., Kutanoglu, E., (2003), Minimizing job tardiness using integrated preventive maintenance planning and production scheduling, IIE Transactions, 35(6), 503–513.
8. Yulan, J., Zuhua, J., Wenrui, H., (2008), Multi-objective integrated optimization research on preventive maintenance planning and production scheduling for a single machine, International Journal of Advanced Manufacturing Technology, 39 (9-10), 954-964.
9. Zhao, C., Ji, M., Tang, H., (2011), Parallel-machine scheduling with an availability constraint, Computers & Industrial Engineering, 61(3), 778-781.
10. Lee, C. Y., (1997), Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint, Operation Research Letters, 20, 129–139.
11. Kubzin, M. A., Strusevich, V. A., (2005), Two-machine flow shop no-wait scheduling with machine maintenance, 4OR, 3(4), 303-313.
12. Liao, L. M., Tsai, C. H., (2009), Heuristic algorithms for two-machine flowshop with availability constraints, Computers and Industrial Engineering, 56(1), 306-311.
13. Hadda, H., (2011), A polynomial-time approximation scheme for the two machine flow shop problem with several availability constraints, Optimization Letters, 1-11, Article in Press.
14. Zhao, C., Tang, H., (2011), A note on two-machine no-wait flow shop scheduling with deteriorating jobs and machine availability constraints, Optimization Letters, 5, 183–190.
15. Ben Chihaoui, F., Kacem, I., Hadj-Alouane, A. B., Dridi, N., Rezg, N., (2011). No-wait scheduling of a two-machine flow-shop to minimize the makespan under non-availability constraints and different release dates. International Journal of Production Research, 49 (21):6273-6286.

[Downloaded from ijaor.com on 2026-11-08]

16. Benbouzid-Sitayeb, F., Varnier, C., Zerhouni, N., (2007), Proposition of new genetic operator for solving joint production and maintenance scheduling: Application to the flow shop problem, Proceedings - ICSSSM'06: 2006 International Conference on Service Systems and Service Management 1, art. no. 4114502, 607-613.
17. Benbouzid-Sitayeb, F., Ammi, I., Varnier, C., Zerhouni, N., (2008), An integrated ACO approach for the joint production and preventive maintenance scheduling problem in the flowshop sequencing problem, IEEE International Symposium on Industrial Electronics, art. no. 4677142, 2532-2537.
18. Ruiz, R., García-Díaz, J. C., Maroto, C., (2007), Considering scheduling and preventive maintenance in the flowshop sequencing problem, Computers and Operations Research, 34, 3314 – 3330.
19. Aggoune, R., Mahdi, A.H., Portman, M. C., (2001), Genetic algorithm for the flow shop scheduling problem with availability constraints, IEEE International Conference on Systems, Man, and Cybernetics, 4, 2546–2551.
20. Aggoune, R., Portmann, M. C. (2006), Flow shop scheduling problem with limited machine availability: A heuristic approach, International Journal of Production Economics, 99, 4–15.
21. Perez-Gonzalez, P. Framinan, J., (2009), Scheduling permutation flowshops with initial availability constraint: Analysis of solutions and constructive heuristics, Computers and Operations Research, 36, 2866-2876.
22. Safari, E., Sadjadi, S., (2011), A hybrid method for flowshops scheduling with condition-based maintenance constraint and machines breakdown, Expert Systems with Applications, 38, 2020-2029.
23. Shoaardebilia, N., Fattahi, P., (2015), Multi-objective meta-heuristics to solve three-stage assembly flow shop scheduling problem with machine availability constraints, International Journal of Production Research, 53(3), 944-968.
24. Vahedi-Nouri, B., Fattahi, P., Ramezanian, R., (2013), Minimizing total flow time for the non-permutation flow shop scheduling problem with learning effects and availability constraints, Journal of Manufacturing Systems, 32(1), 167–173.
25. Besbes, W., Teghem, J., Loukil, T., (2010), Scheduling hybrid flow shop problem with non-fixed availability constraints, European Journal of Industrial Engineering, 4(4), 413-433.
26. Allaoui, H., Artiba, A., (2014), Hybrid Flow Shop Scheduling with Availability Constraints, Essays in Production, Project Planning and Scheduling, International Series in Operations Research & Management Science, 200, 277-299.
27. Wang, S., Liu, M., (2014), Two-stage hybrid flow shop scheduling with preventive maintenance using multi-objective tabu search method, International Journal of Production Research, 52(5), 1495-1508.
28. Graham, R. L. Lawler, E. L. Lenstra, J. K., Rinnooy Kan, A. H. G., (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics, 5, 287–362.
29. Lee, C. Y., Chen, Z. L., (2000), Scheduling jobs and maintenance activities on parallel machines, Naval Research Logistics, 47, 145-165.
30. Nawaz, M., Enscore Jr, E., Ham, I., (1983), A heuristic algorithm for the m-machine, n job flowshop sequencing problem, OMEGA International Journal Management Science, 11, 91-95.
31. Chen, Y. W., Lu, Y. Z. Yang, G. K., (2008), Hybrid evolutionary algorithm with marriage of genetic algorithm and extremal optimization for production scheduling, International Journal of Advanced Manufacturing Technology, 36, 959–968.
32. Colorni, A., Dorigo, M., Maffioli, F., Maniezzo, V., Righini, G., Trubian, M., (1996), Heuristics from Nature for Hard Combinatorial Problems, International Transactions in Operational Research, 3(1), 1-21.
33. Haupt, R. L., Haupt, S. E., (2004), Practical genetic algorithms, John Wiley & Sons, 2th Edition.
34. Taguchi, G., Chowdhury, S., Taguchi, S., (2000), Robust engineering, McGraw-Hill, New York.
35. Jeff Wu, C.F., and Hamada, M. (2002), Experiments: Planning analysis and parameter design optimization, John Wiley & Sons.
36. Phadke, M.S., (1989), Quality engineering using robust design, USA: Prentice-Hall.

[Downloaded from ijaor.com on 2026-11-08]