J. Ind. Eng. Int., 7(14), 7-18, Summer 2011
ISSN: 1735-5702
© IAU, South Tehran Branch

Scheduling of flexible manufacturing systems using genetic algorithm: A heuristic approach

Vijay Kumar [1*]***; A. N. Narashima Murthy*** [2]***; Krishnappa Chandrashekara*** [3]

[1] *Professor, Dept. of Mechanical Engineering, JSS Academy of Technical Education, Bangalore, India*

[2] *Professor, Dept. of Mechanical Engineering, Sri Jayachamarajendra College of Engineering, Mysore, India*

[3] *Professor, Dept. of Mechanical Engineering, T. John College of Engineering, Bangalore, India*

Received: 7 August 2008; Revised: 2 July 2009; Accepted: 12 October 2009

Abstract: Scheduling of production in Flexible Manufacturing Systems (FMSs) has been extensively investigated over the past years and it continues to attract the interest of both academic researchers and practitioners. The generation of new and modified production schedules is becoming a necessity in today's complex manufacturing environment. Genetic algorithms are used in this paper to obtain an initial schedule. Uncertainties in the production environment and modeling limitations inevitably result in deviations from the generated schedules. This makes rescheduling or reactive scheduling essential. One of the four different types of uncertainties that normally cause discrepancies between the actual output and the planned output is considered in this paper. These include unforeseen machine break-downs, increased order priority, rush orders arrival and order cancellations. In this paper, the current status of the shop is considered while rescheduling. The proposed algorithms revise only those operations that must be rescheduled and can, therefore, be used in conjunction with the existing scheduling methods to improve the efficiency of flexible manufacturing systems.

Keywords: *Flexible manufacturing Systems; Scheduling; Genetic algorithm*

1. Introduction

At present, most industries are confronted with perpetual customer demands for a wider variety of products, faster production rates, shorter delivery time and more reliable delivery. The flexibility to manufacture a wide range of products in short time has been achieved at the expense of manufacturing efficiency. This deficiency is reduced with the introduction of flexible manufacturing systems (FMSs) which have the flexibility of job shops while approaching the efficiency of transfer lines. One aspect of such manufacturing systems which is particularly difficult in practice is scheduling. Scheduling in FMSs differs from that in a conventional job shop because of the availability of alternating manufacturing resources resulting in routing flexibility. This may potentially increase the output by eliminating the bottle-necks often present when alternate routes are not feasible. The performance of a production system depends greatly on good and proper rescheduling with the uncertainties present in the production environment. These uncertainties often result in orders following a route through the shop floor different from the one originally developed. In such cases, previously generated schedules become invalid and have to be regenerated.

Several factories related to system design, production control and inventory control can cause interruptions in an FMS. In this paper, the researchers limit the discussions to disruption caused by unforeseen machine break-downs, increased order priorities, rush order arrivals and order cancellations. The proposed rescheduling algorithms can be used along with existing scheduling systems to improve effectiveness. Efforts are focused on necessary local rescheduling only to maintain the stability of existing schedules and provide quick solutions. The main emphasis of rescheduling is to find immediate solutions to the problems resulting from disturbances in the production system. This may result in a less effective overall schedule compared with total rescheduling; however, it meets the new constraints with the least disruption of ongoing production. Rescheduling commences from the time a disturbance occurs and takes into account the current state of production on the shop floor. It is an iterative process of two steps: (1) reschedule and evaluate the existing schedule depending on the change of the conditions, demands or constraints.
If the result of the revised schedules is acceptable, then stop; otherwise: (2) determine an improved solution by performing iterations. An acceptable revised schedule is the one that overcomes few

*Corresponding Author Email: vijayjss@yahoo.com
Tel.: + 91 9844789224

constraints, for example, a schedule that allows re-routing of parts from a machine that has just broken down. In reality, defining an acceptable revision depends on the prevailing requirements.

In this paper, a new scheduling approach is presented based on genetic and rescheduling algorithms which can be used to complement the currently used computer-based scheduling systems. The proposed algorithms assume that some of the jobs in the system have alternative process plans, a common situation in FMS. In this paper genetic algorithms are used for generating an initial schedule.

2. Literature review

Genetic algorithms (GAs) have been applied to a variety of function optimization problems, and were shown to be highly effective in searching large, complex response surface even in the presence of difficulties such as high dimensionality, multimodality and discontinuity (Goldberg, 1989). Stochastic process generates an initial population of schedules and then principles of natural selection/survival of the fittest are applied to improve the schedules. The major obstacle in applying genetic algorithms in manufacturing scheduling is finding an appropriate representation of a schedule. These algorithms are called genetic because the manipulation of the possible solutions resembles the machines of natural selection. Their chief advantage lies in their ability to jump randomly from schedule to schedule, allowing them to escape from local optima in which other algorithms often get trapped.

Cheng *et al.* (1999) was instructive rather than realistic in that they showed how genetic algorithms can be used to solve a simple job shop scheduling problem. Cheng and Sin (1990) investingated the simplest scheduling problem, i.e. a static queue of jobs with specified due dates and run time without precedence constraints, with a single server, and which used minimal lateness as a criterion. The problem representation does not allow using a conventional crossover operator as it would create illegal schedules. The authors investigated three different crossover operators for the job shop scheduling: PMX, a weak greedy crossover, and a powerful greedy crossover. They concluded that a choice of crossover operators would make a difference in scheduling the results. Gordon *et al.* (2002) applied genetic algorithms to small, medium and large job shop problems. The two crossover operators they used were PMS (partially matched crossover and LOX (linear order crossover). They considered that each operation could be done only on one machine. They compared the performance of their genetic algorithms with the two most widely used scheduling heuristics, namely LST (least slack time) and SPT (shortest processing time).Their experiments prevailed the superiority of the genetic algorithms approach over the common heuristics and a slight superiority of the LOX operator over the PMX operator. Graves (1981) addressed an n-job, a single machine problem with an objective to minimize the flow time variance. They proposed heuristic procedure based on genetic algorithm. With the potential to address a more generalized objective function such as weighted flow time variants. They also used a PMX (partially Matched crossover) operator for their experiments and mention investigating the performance of other crossover operator.

The above applications of genetic algorithms suffer from major drawback since the experiments did not consider some important scheduling constraints such as precedence among manufacturing tasks, alternative process plans, and dispatching rules which are major factors contributing to the difficulty of the scheduling problem. Recently, Lauff and Werner (2004) addressed the problem of alternative process plans and developed a new two chromosomes representation where each chromosome represents a complete schedule. They did not consider the due dates and the system was tested using only one performance measure, viz, machine utilization. Yen and Wan (2003) applied genetic algorithms in FMS scheduling where other important scheduling aspects were considered such as precedence constraints, due dates, alternate routings, variable batch size and variable task processing time.

The results obtained by all of the above authors indicate that genetic algorithms present a good scheduling alternative: they are reasonably fast, gradual (i.e. some solution is available immediately) and provide better results than the heuristics (based on the objective function of the problem). This paper used steady state genetic algorithms for solving the multiple routing scheduling problem.

3. Problem description

The FMS scheduling problem may be stated as: “Given alternative process a plan (routing flexibilities) for each part, the objective is to find a feasible schedule for a given set of part types such that some given performance criterion is optimized". The prerequisite of this stage is the process plan for each part, and other data including the no of type of jobs, the number of task in each job, number and types of machines available, processing and set up time of task on machines, order due dates, release time of jobs into the shop floor and performance criterion to be chosen. Typical measures chosen to evaluate the schedules in this research are as follows:

3.1. Notations

The following notations are used in this research:

I	job number i (i=1,2,3….n)
K	machine k (k=1,2,3…..m)
j_i	number of operations of job i
Oij	operation number j of job i($Oi1,Oi2....Oij$)
Cij	completion time of operation Oij
Ti	tardiness of job i
U_k	utilization of machine k

- Minimize mean flow time

$$\text{Mean flow time} = -/n \sum_{i=1}^{n} \sum_{j=j_i}^{n} Ci \tag{1}$$

- Minimize mean tardiness-tardiness is the positive difference between the job's completion time and its due date.

$$\text{Tardiness} = Ti = \sum_{J=Ji} Ci - di, I \tag{2}$$

$$\text{Mean tardiness} = 1/n \sum_{i=1}^{n} [Ti] \tag{3}$$

- Maximize average resource utilization-the %utilization of an individual machine is calculated based on the maximum flow time, i.e. for a schedule, the individual and average resource utilization is calculated as follows:

- $$\text{Utilization} = Uk = \frac{total\, busy\, time}{\max(Ci)} \tag{4}$$

$$\text{Average utilization} = 1/m \sum_{k=1}^{m} Uk$$

All the criteria are evaluated in the evaluation function of the Gas. The evaluation function gives a complete schedule and an associated performance criterion value. The earliness of the job can be handled in the same way as tardiness, where a penalty is assigned for completing the jobs before its due date.

3.2. Genetic algorithms formulation

The achievement of the optimum schedule that confirms to the stated objective is a naturally attractive aim of the scheduler. It was seen from the literature review that substantial effort has been devoted to the associated mathematic computational background and the methods explored have been wide ranging and often mathematically complex. Various issues in the design of genetic algorithms for combinatorial-type scheduling problems such as schedule representation, population initialization, evaluation function, recombination operators, and parameter values are discussed by Qi *et al.* (2000) and are described here briefly.

Evaluation functions determine the quality of solutions in the genetic population. In each generation, the value of the fitness function or the objective function for a particular schedule is calculated by assigning values onto the machines. The evaluation function treats operations in an order consistent with the precedence relations of the problem. Once, all the predecessors of the problem have been scheduled, the operation is said to be schedulable, regardless of the actual time at which the next scheduling decision is required. This procedure is repeated until all the operations have been scheduled. The genetic algorithms proceed from generation to generation, saving the best schedule in each generation and getting rid of those schedules with objective function values lower than those of the saved ones

Two genetic operators used in this paper are reduced surrogate crossover, for crossing two parent schedules; and adaptive mutation, for random introduction of new genes in the chromosomes (schedule). The other recombination operators may result in an invalid machine assignment and therefore an infeasible schedule. Crossover is usually implemented by choosing one crossover point at random, then exchanging segments between the two parent strings, thus forming new children which contain information from each of the two parent chromosomes. The problem of using such a crossover is that it can easily create duplicate chromosomes (schedules).

Reisman *et al.* (1997) used adaptive mutation in this research as opposed to normal mutation. Adaptive mutations base the amount of disruption to a given string on two factors: the relative similarity of its two parent strings, and a mutation rate.

The more similar the two parent strings are, the more likely mutation is to occur. The actual mutation which occurs is the product of this similarity and a fractional mutational rate. Thus, if the mutation rate is 0.20 and the parents of a particular string were identical, approximately 20% of the strings will be mutated. Or, if , if the parents contained 50% unique information, only about 10% of the string will be mutated. A variety of parameters such as selection bias (SBIAS), adaptive mutation (AMUT), population size (PSIZE), and a number of generations (NGEN) plays a crucial role in the successful implementation of genetic algorithms. The settings of these parameter values significantly affect the performance of genetic algorithms. The problem of setting the parameter values has been extensively studied for bit string representation, Hejaji and Saghafian (2005). Guo *et al.* (2006) conducted extensive experiments to select parameters for list representations, and their findings are given in 3.5.

3.3. Problem review using steady state genetic algorithms

The problem considers a flexible manufacturing system with five machines which are capable of processing four different jobs is considered is shown in Table 1. The problem size is enhanced limiting number of machines to nine and number of jobs to ten, Table 7. Each job has three tasks that should be capable of processing four different jobs, is considered. The processing times for each job is depicted in Table 5. Each job has three tasks that should be performed in a strict sequential manner. Priorities among the jobs are assigned random. The search process begins with the proper representation of a schedule, generation of schedule population and evaluation of each schedule in the population, and application of genetic operators to this population for improving the schedules. The search process continue for a specified number of generations yielding an optimal, or a new optimal solution. All data sets for this problem are given in Appendix. The task sequences and their quantity are listed in Table 6. There are five machines in a machining work cell, two in the review and one in the inspection and the wash work cells. All the machines in the machining work cell are capable of performing operations such as turning, drilling, facing, milling, boring and screwing. Task processing and setup time information are given in Table 7. This example considers a planning horizon of 10000 units.

The genetic algorithms for the scheduling problem is developed incorporating the design issues discussed above. These steps are now discussed in detail.

(1) (Chromosome representation) Represent schedule as a list of machines.

(2) (Initialization) Select the initial parameters and create an initial diversified population of schedules.

(a) Set the values for PSIZE, SBIAS, AMUT and NGEN. The length of the chromosome is set to equal the total number of operations to be scheduled.

(b) Read stage process times, setup times, due dates and ready times for all the jobs.

(c) Create an initial population of schedules of size PSIZE and call it 'oldpop'.

(d) Calculate the objective function values for all the schedules on the population using evaluation function described earlier.

(e) Sort the population in an increasing order of objective function value.

(f) Set NGEN=1(i.e current generation=1)

(3) (Recombination) Apply recombination operator to the óld pop' to form a new population.

(a) Select two parents from the old pop based on the specified schedule selection bias. Index=PSIZE x (SBIAS-sqrt (BIAS^2-4.0 x (SBIAS-1) x random()))/2.0/ (SBIAS-1) Where index is the schedule number to be selected from the sorted population.

(b) Apply the reduced surrogate crossover operator to the two selected parents to form a new child.

(c) Apply the adaptive mutation operator to this child.

(d) Calculate the objective functional value for this child using the evaluation function.

(e) If the objective function value of the child is better than any of the schedules in the population, then insert the child in the population at the appropriate place according to the value of its objective function and remove worst schedule from the population.

(4) (New Generation) Evaluate the current generation number (GEN) to determine the next step.

(a)If GEN<NGEN, then GEN is incremented by one and the current population becomes oldpop. Go to Step (3).

(b) If GEN=NGEN, then Stop.

The best schedule would be the schedule in the current population with the highest objective function value.

3.4. Selection of genetic algorithms parameters

The most difficult and time-consuming issue in the successful implementation of the genetic algorithms is finding good parameter sittings. A number of approaches have been suggested to derive robust parameter setting for traditional GAs, including carrying out brute force searches using an adaptive operator fitness technique. In one of the most extensive studies for determining the optimal parameter values, Subramanian *et al.* (2000) concluded that the optimal parameter settings vary according to the problem. However, very little work has been reported regarding setting the parameters for steady state genetic algorithms used for the combinatorial-type scheduling problems. The main parameters required for the steady state genetic algorithms are population size (PSIZE), selection bias (SBIAS) and adaptive mutation rate (AMUT).

In this paper, a different approach, based on selection probability, is used for crossover. The reduced surrogate crossover employed in this paper returns the difference between the two schedules. Instead of crossover probability, selection probability is used and is defined as a selection bias which can take a value between 1.0 and 2.0. Selection bias is a floating-point number used in the selection of two schedules for genetic reproduction (crossover).This number specifies the amount of preference to be given to the superior individuals in a genetic population. For example, a bias of 2.0 indicates that the best schedule has twice the chance of being chosen compared to the average schedule. Chan and Chan (2004) conducted that a medium population size, high selection bias, and low mutation rate gives the best performance. They recommend the following sets of parameters for steady state genetic algorithms:

(1) Population size: 80

(2) Selection bias: 1.9

(3) Adaptive mutation rate: 0.1

3.5. Result validation

In this section, the performance of steady state genetic algorithms is evaluated against two other existing approaches. These problems are chosen from published literature and are given in Tables 5, 6, 7 and 8. The first problem was taken from Nasr and Elsayed (1990) in which they solved the problem of multiple routeing scheduling using a bound algorithm where the best mean flow time was reported to be 12.25 units. The best found solution using genetic algorithms was 11.50 units.

The second example was extracted from Dutta (1990) who used an artificial intelligence approach to solve the problem. Dutta reported a value of 84.0 for the make span whereas by using a GAs approach, a make span of 76.0 was achieved. For both examples, the time taken to achieve the solution was not reported. Table 2 gives the comparison of results in terms of performance gain in the quality of solutions.

4. Rescheduling in manufacturing

Rescheduling is needed as a result of significant changes in the operating conditions. Rescheduling is never planned in advance but is brought on as a result of certain unavoidable circumstances. The proposed rescheduling algorithms are simple, and hence can be used for locally revising schedules in real time, as opposed to rescheduling the overall production which is normally considered over a period of weeks or days.

Rescheduling can be carried out either manually or through application software. The manual method involves editing the existing schedules which are normally in the form of Gantt chart. This method is tedious, time consuming and can potentially compromise the efficiencies of the small schedule. Researchers have often used simulation for rescheduling purposes. A good survey of articles involving dynamic job shops scheduling is presented in Hasija and Rajendran (2004). Zhang *et al.* (2000) used a 'regeneration' method in developing their scheduling systems. This method involves rescheduling the entire set of operations of jobs including those unaffected by the change in conditions, demands, and/or constraints. This is time consuming, and often results in response times unacceptable to the user. They compared three scheduling procedures to deal with machine breakdowns. However they did not address the problems of other uncertainties such as rush orders, increased priority and order cancellations. Lauff and Werner (2004) used simulation to investigate rescheduling, and approached the problem by selecting between sequencing and dispatching in case of uncertainties. A schedule is first determined by using a branch and bound technique. This approach is switched to dispatching, which uses either the FCFS or SPT rule, when there is a change in production conditions. Stoop and Wiers, (1996) proposed a rescheduling algorithm based on the construction of a scheduling binary tree and a net change concept adopted from MRP systems. One limitation of the algorithm is that it can deal only with rescheduling situations that assume no change in the existing operation sequence for each machine. They did not consider the alternate routings for rescheduling. Wang and Wu (2000) proposed a knowledge-based system to help automate the control activity at the scheduling level in FMS environments. The author assumed a batch size of one which is rarely the case in actual production systems.

4.1. Rescheduling algorithms

Several types of random variables (uncertainties) that affect the actual shop output should be taken into account if scheduling is to be realistic. This paper considers one of the different types of uncertainties:

- machine breakdown algorithm
- the arrival of rush orders,
- increased order priority (i.e. the changes in due dates), and
- order cancellation

In each of the above cases, tasks are performed in accordance with the predetermined task sequence, even after the disturbance occurs. If a task is unable to continue as scheduled, then based on its priority an attempt is made first to find an Alternative free machine, failing which pre-emption is attempted. The system state such as machines and tasks status and ready and completion time for each tack are updated whenever a task either is ready to start or is completed, or when an interruption.

When a machine breakdown occurs, the remaining operations of the job may have to be performed using other machines. This affects the scheduling decisions previously made. The rescheduling becomes even more difficult if certain tasks can be completed on only one machine. At the time of interruption, if a task is being performed on the broken machine, the system is checked for availability of alternate machines. If an alternative machine is free, the pre-empted task is assigned to it. If an alternative machine is performing another task, the priority of the two tasks is compared, and the task with higher priority is assigned to that machine. Also, task setup time on a machine is an important factor in rescheduling. In the case of a breakdown, a comparison is made between the task setup time of the pre-empted task on an alternate machine and the expected down time of the failed machine. If the task setup time is lower, only then is the task switched to the alternate machine; otherwise, it waits until the broken machine becomes available for production.

In this research, an initial schedule is obtained using GAs. An example problem with the machine and task parameters shown in Figure 1 was generated for illustrating the proposed rescheduling algorithms. A manufacturing system with five machines, which are capable of processing four different jobs, is considered.

Table 1: Setup/processing time requirements of task.

Job (i)	Task (Oij)	Machines					Priority
		M1	M2	M3	M4	M5	
1	11	10/20	-	12/21	-	-	1
	12	-	-	-	8/22	12/20	
	13	5/15	-	6/12	-	-	
2	21	8/14	-	-	-	-	2
	22	-	8/22	10/18	-	-	
	23	7/29	-	10/30	-	-	
3	31	12/38	12/40	-	-	14/36	3
	32	-	-	-	9/21	7/23	
	33	23/41	-	-	-	21/39	
4	41	20/53	-	18/52	22/50	-	4
	42	-	9/33	-	12/28	-	
	43	6/22	8/23	-	-	-	

Table 2: Comparison of GA's performance with respect to two other approaches.

P	N	M	Method	Measure	Value			CPU Seconds	
					Reported	GA	D	Reported	GA
1	4	6	Mixed integer programming	Mean flow time	12.25	11.50	6.1%	N/A	0.20
2	6	7	Artificial Intelligence	Makespan	84.0	76.0	9.5%	N/A	0.32

P = Problem size, N = Number of jobs, M = Number of Workcentres/machinesD=Gain in performance

Each job has three tasks that should be performed in a strict sequential manner. Priorities among the jobs are assigned at random. The search process begins with the proper representation of a schedule, generation of schedule population and evaluation of each schedule in the population, and application of genetic operators to this population for improving the schedules. The search process continues for a specified number of generations yielding an optimal, or a near optimal, solution. Utilization of machines is an important issue to be considered while rescheduling manufacturing resources. For example, in the case of a breakdown if the affected task has a choice of more than one machine, it is re- routed to the least utilized machine and the task status are updated continuously while running the schedule. Sabuncuoglu and Gurgun (1996) identified two phases before rescheduling is invoked. These are: the planning phase, where an initial schedule is to be generated, and the control phase, where schedule progress is monitored and abnormal states are identified. Once an abnormality is identified by the control phase, a reschedule procedure is invoked. Based on the type of abnormality, certain decisions are made before the rescheduling algorithm is applied to the schedule. For example, in the case of machine breakdowns, the expected duration of breakdown has to be considered. In the case of new orders, it must be determined whether it is a normal order or a rush order. If it is a normal order, it is merged into an existing schedule, and all tasks of the new order are given the same treatment as other tasks. If it is a rush order, the highest priority is assigned to it and machine assignment us made accordingly. The framework of the control phase employed in this research is depicted in Figure 1.

As discussed earlier, the control phase of a manufacturing system includes monitoring and adjusting the system's schedule to ensure smooth order progress on the shop floor. The basic research issue addresses in this paper is that of determining how real-time control in the FMS can be achieved in the presence of uncertainties. After an initial schedule is loaded, the basic execution loop is as follows:

```
    While all tasks are not complete Do;
Step 1: Monitor schedule progress
Step 2: Examine system disturbances
Step 3: Classify the control problems (such as
        breakdown, rush order etc.)
Step 4: Select alternative actions if available
Step 5: Update status of machines and uncom-
        pleted tasks
Step 6: Reschedule remaining tasks
End;
```

The algorithms used for different types of disturbances are illustrated with examples in the following section.

4.1.1. Machine breakdowns

The following loop is executed after the initial schedule is loaded, for time T= 0, 1, 2, 3, until the schedule is complete.

At any time t, IF there is a machine breakdown, then find the broken machine and interrupted task.

Assign expected downtime randomly for this machine

```
    IF there is any operation currently on this ma-
    chine, then
    Split the task and revise the task status (time
remaining)
    IF there are alternative machines available,
    find setup and processing time required for all
    alternative machines. If more than one choice
    is available choose the least utilized machine
ELSE broken task will start on the machine when-
ever it becomes operational
    ENDIF
    IF there is a choice of alternate machines
    IF alternate machine is free, assign broken task
    to alternate machine and update the system
    status
ELSE (if alternate machine is not free)
    IF broken task priority is higher than the task
    currently performed on alternate machine, then
    pre-empt the alternate machine and update the
    system status
ELSE (if priority is lower)
    IF the difference between the ready time of the
    broken machine and release time of an alter-
    nate machine is more than the setup time on al-
    ternate machine, then assign broken task to al-
    ternate machine when it becomes free
ELSE broken task will start on the same machine
whenever it becomes
Ready
    ENDIF
    ENDIF
    ENDIF
ELSE broken task will start on the same machine
whenever it becomes operational
    ENDIF
ELSE update the machine status list
    ENDIF
```

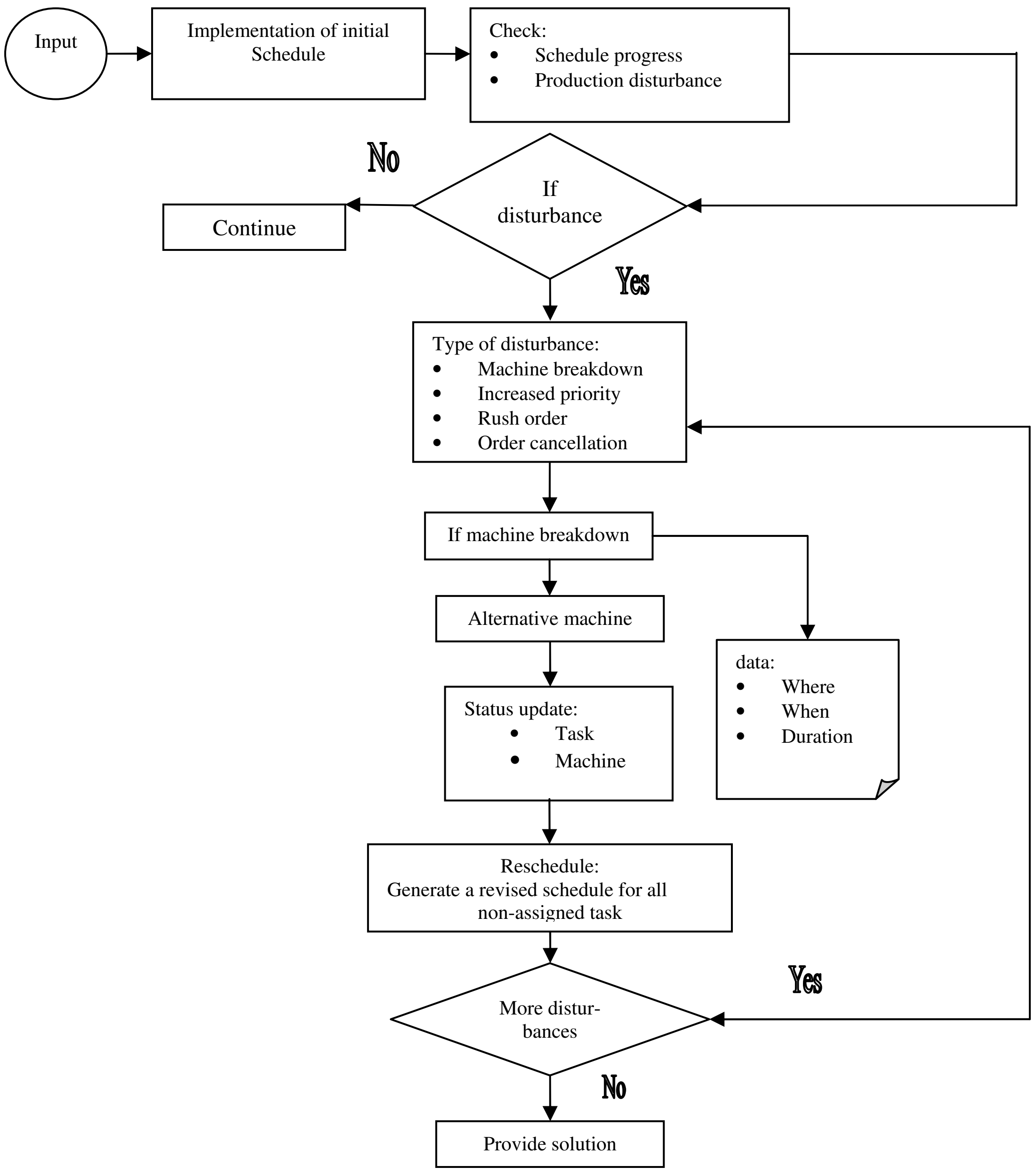

Figure 1: Control framework.

5. Results and discussions

Dynamic reaction to developments on the shop floor is essential for realizing a truly flexible control of the manufacturing system. In order for a controlling mechanism to perform in a dynamic production environment, it must consider scheduling or dispatching rules, as well as the system performance. We examined three performance measures: mean flow time, mean tardiness and average machine utilization with respect to the dispatching rules, namely the first-come-first-served rule, and studied their degradation with respect to machine failure rates.

The problem size is enhanced considering six work centres and ten jobs to illustrate and validate the algorithm. The example considers a planning horizon of 10000 units. At any discrete point in time, any machine could fail randomly and independently with the same probability of failure. This failure probability is dependent on the random number cut off value. At the beginning of the scheduling period, 10000 numbers between 0 and 1 are randomly generated to represent discrete points of time in the planning horizon, and cut off values are used in determining the machine failure probability. For example, if a cut off value is 0.0004, then at each discrete point of time in planning horizon, the value of the corresponding random number is compared with the cut off value. A value less than the cut off value will result in a machine failure. For our example, a cut off value of 0.0004 resulted in a failure probability of 0.06% or a failure frequency of 6, i. e., machines will fail 6 times during the planning horizon. The duration of failure was machine-dependent and randomly generated. Table 3 shows the random number cut off value, probability of machine failures and frequency of machine failures.

Table 4 shows the effect of machine breakdown on the mean flow time, mean tardiness and ave-rage machine utilization with respect to dispatching rules. The mean completion time is computed using individual completion time. Machine utilization is equal to the time a machine was occupied divided by the maximum flow time. For mean tardiness, due dates for each order are assigned based on the total work content of orders and are set equivalent to (DDT *total processing time), where DDT is the due date tightness. In this experiment, a setup due date tightness factor (DDT) was set to 1.5.

These performance criteria have been charted in Figures 2, 3 and 4. Because alternate routings are available in the system, the graph shows that in most cases, the system performance degradeation is initially gradual up to a machine failure probability of 0-12% for FCFS rule. As the probability of failure increased, the system performance deteriorated drastically. This is due to the larger number of jobs competing for the machines available for production.

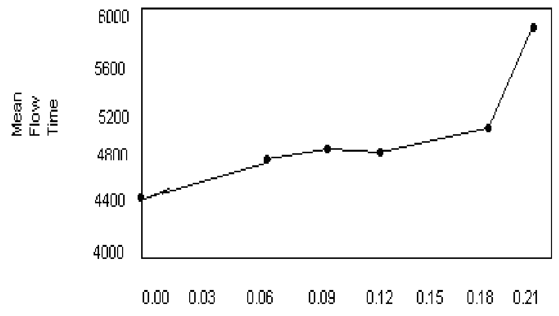

Figure 2: Machine failure probability.

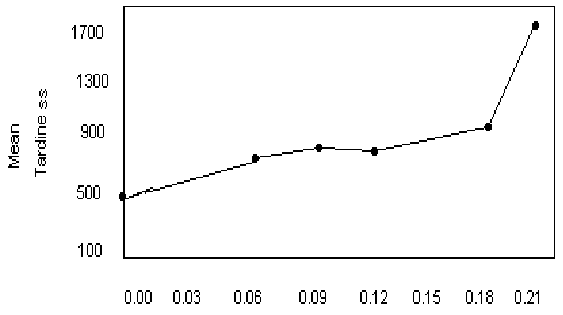

Figure 3: Machine failure probability.

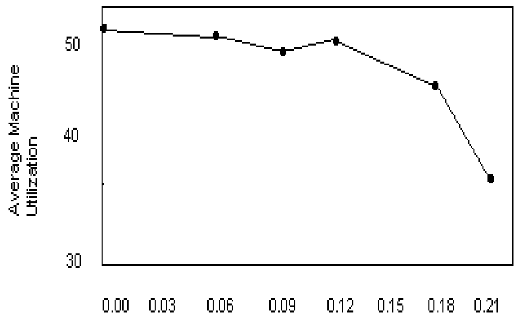

Figure 4: Machine failure probability.

Table 3: Cut off value, failure probability and frequency.

Cut off Value	0.0000	0.0004	0.0008	0.0012	0.0016	0.0020
Failure Probability	0.00	0.06	0.09	0.12	0.18	0.21
Frequency	0	6	9	12	18	21

Table 4: Performance measures versus failure probability for FCFS dispatching rule.

Performance Measure	0.00	0.06	0.09	0.12	0.18	0.21
Mean Flow Time	4420	4561	4611	4615	4766	5736
Mean Tardiness	470	554	591	594	759	1639
Average Machine Utilization	50.93	50.91	49.78	49.73	46.14	36.20

Table 5: Processing times for FMS problem (Nasr and Elsayed, 1990).

Job (i)	Operations (j)	Machines (M_K)					
P_i	O_{ij}	M_1	M_2	M_3	M_4	M_5	M_6
	O_{11}	2	3	4	-	-	-
P_1	O_{12}	-	3	-	2	4	-
	O_{13}	1	4	5	-	-	-
	O_{21}	3	-	5	-	2	-
P_2	O_{22}	4	3	-	-	6	-
	O_{23}	-	-	4	-	7	11
P_3	O_{31}	5	6	-	-	-	-
	O_{32}	-	4	-	3	5	-
	O_{33}	-	-	13	-	9	12
	O_{41}	9	-	7	9	-	-
P_4	O_{42}	-	6	-	4	-	5
	O_{43}	1	-	3	-	-	3

Table 6: Task precedence and order quantity.

	Task1	Task2	Task2	Task2	Task2	Quantity
Job Type1	Turn	Review	Drill	Inspection	Wash	20
Job Type2	Bore	Wash	Screw	Inspection		20
Job Type3	Turn					40
Job Type4	Mill	Wash	Inspection	Drill	Review	20
Job Type5	Turn	Wash				40
Job Type6	Mill	Wash	Inspection	Review		40
Job Type7	Face	Wash	Inspection	Drill	Wash	20
Job Type8	Mill	Review				40
Job Type9	Drill	Review	Wash	Turn		20
Job Type10	Mill	Wash	Inspection			40

Table 7: Machines available in work cell.

	Machining work cell		Review work cell	Inspection work cell	Wash work cell
Machines	M1,M2,M3,M4	M4,M5	M6,M7	M8	M9
Tasks	Turning, Facing, Driliing Milling, Boring, Screwing		Review	Inspection	Wash

Table 8: Processing time (Setup time) information.

Jobtask	Resources								
No	M1	M2	M3	M4	M5	M6	M7	M8	M9
1—1	97(196)	88(176)	95(190)	86(172)	90(180)				
1—2						6(12)	5(10)		
1—3	34(68)	34(68)	33(66)	31(62)	32(64)				
1—4								11(22)	
1—5									7(14)
2—1	35(70)	36(74)	32(68)	32(64)	36(72)				
2—2									10(20)
2—3	65(130)	59(118)	62(124)	60(120)	62(124)				
2—4								15(30)	
3—1	54(108)	48(96)	48(98)	50(100)	51(102)				
4—1	41(82)	43(86)	39(78)	43(80)	38(76)				
4—2									9(18)
4—3								8(16)	
4—4	86(172)	93(186)	84(168)	84(180)	88(176)				
4—5						610(612)			
5—1	85(170)	82(164)	89(178)	87(174)	89(178)				
5—2									6(12)
6—1	60(120)	66(132)	60(134)	65(130)	66(128)				
6—2									7(14)
6—3								20(40)	
6—4						6(12)	6(10)		
7—1	78(156)	84(168)	80(160)	86(172)	86(164)				
7—2									9(18)
7—3								19(38)	
7—4	66(132)	63(124)	59(118)	63(126)	63(122)				
7—5									10(20)
8—1	23(46)	24(44)	24(48)	24(50)	23(47)				
8—2						10(18)	10(20)		
9—1	62(124)	62(128)	71(142)	67(134)	69(138)				
9—2						8(16)	8(16)		
9—3									10(20)
9—4	48(96)	44(88)	45(90)	48(94)	44(86)				
10—1	61(132)	65(130)	60(120)	59(118)	63(126)				
10—2									8(16)
10—3								12(24)	

6. Conclusion

In this paper, the researchers have investigated the application of GA-based techniques to the flexible manufacturing scheduling problem considering four jobs and five machines initially and there after enhanced to ten jobs and six machines. This work, and other current investigations, demonstrates that the genetic algorithm method is broad, approximate search procedure with applications in diverse problem areas. The method does not depend upon an underlying continuity of the search space and requires no information other than fitness or payoff values. Since genetic algorithms always work with a fixed population size, i.e., the number of strings (chromosomes) is fixed in a population and does not change with the generation; there is tremendous reduction in the space in which a GA looks for the optimal solution.

In the second part of the research reported a rescheduling algorithms is developed that generate a new schedule without re-evaluating all tasks in the old schedule. These algorithms use the system status as input and reschedule the tasks when disturbances occur. This is particularly in important in real-time and dynamic scheduling environments in modern manufacturing systems. It is expected that the proposed algorithms can be used together with existing computerized scheduling systems whether or not they use GAs. Since alternate machine choices area available for the tasks in flexible manufacturing systems, the effect of disruptions on the system's performance is minimized.

The Scheduling/Rescheduling algorithm considers a scheduling period of 10000 units. As the cut off value increases 0 to 0.002, then at each discrete point of time in the planning horizon, a value less than a cut off value will result in a machine failure. The machine failure frequency varies from 0 to 21 for the given cutoff value considered. From the graph charted for measure of performance, the system performance degradation is initially gradual up to a machine failure probability of 0.12% for FCFS dispatching rule. It is concluded from the results that FCFS dispatching rule performed well for the three performance measure. An important property of genetic algorithms observed here was that since GAs performs a multi-point search as opposed to a single point, the time required to reach an optimal or near optimal solution is small compared to other approaches. Since GAs are faster, they can be used to reschedule the manufacturing tasks dynamically in the case of interruptions. When an interruption occurs, the system status is updated and the genetic algorithms are rerun at that point of time to schedule the remaining manufacturing tasks. Together, these qualities permit the use of GA's in complex areas such as manufacturing scheduling.

References

Chan, F.; Chan, H., (2004), A comprehensive survey and future trend of simulation study on FMS scheduling. *Journal of Intelligent Manufacturing*, 15(1), 87-102.

Cheng, R.; Gen, M.; Tsujimura, Y., (1999), A tutorial survey of job-shop scheduling problems using genetic algorithms: Hybrid genetic search strategies. *Computers & Industrial Engineering*, 36(2), 343-364.

Cheng, T.; Sin, C., (1990), A state-of-art review of parallel-machine scheduling research. *European Journal of Operation Research*, 47, 271-292.

Dutta, A., (1990), Reacting to scheduling exception in FMS environments. *IIE transactions*, 22(4), 300-314.

Goldberg, D. E., (1989), *Genetic algorithms in search optimization and machine learning*. Addison-Wesley, Reading, MA.

Gordon, V.; Proth, J.; Chu, C., (2002), A survey of the state-of-art of common due date assignment and scheduling research. *European Journal Operational Research*, 139(1), 1-25.

Graves, S., (1981), A review of production scheduling. *Operations Research*, 29(4), 646-675.

Guo, Z. X.; Wong, W. K.; Leung, S. Y. S.; Fan, J. T.; Chan, S. F., (2006), Mathematical model and genetic optimization for the job shop scheduling problem in a mixed- and multi-product assembly environment: A case study based on the apparel industry. *Computers & Industrial Engineering*, 50(3), 202-219.

Hasija, S.; Rajendran, C., (2004), Scheduling in flowshops to minimize total tardiness of jobs. *International Journal of Production Research*, 42(11), 2289-2301.

Hejazi, S,; Saghafian, S., (2005), Flowshop-scheduling problems with makespan criterion: a review. *International Journal of Production Research*, 43(14), 2895-2929.

Lauff, V.; Werner, F., (2004), On the complexity and some properties of multi-stage scheduling problems with earliness and tardiness penalties. *Computers & Operations Research,* 31(3), 317-345.

Lauff, V.; Werner, F., (2004), Scheduling with common due date, earliness and tardiness penalties for multi-machine problems: a survey. *Mathematical & Computer Modelling*, 40(5-6), 637-655.

Michalewicz, Z., (1992), *Genetic algorithm+data structures=evolution programs*. Springer, Berlin Heidelberg New York.

Nasr, N.; Elsayed, E. A., (1990), Job shop scheduling with alternative machines. *International Journal of Production Research*, 28(9):1595-1609

Qi, X.; Yin, G.; Birge, J., (2000), Single-machine scheduling with random machine breakdowns and randomly compressible processing times. *Stochastic Analysis & Applications,* 18(4), 635-653.

Reisman, A.; Kumar, A.; Motwani, J., (1997), Flowshop-scheduling/sequencing research: a statistical review of the literature. 1952-1994. *IEEE Transactions on Engineering Management,* 44(3), 316-329.

Sabuncuoglu, I.; Gurgun, B., (1996), A neural network model for scheduling problems. *European Journal of Operational Research*, 93, 288-299.

Stoop, P.; Wiers, V., (1996), The complexity of scheduling in practice. *International Journal of Operations & Production Management*, 16(8), 37-53.

Subramaniam, V.; Lee, G.; Ramesh, T.; Hong, G.; Wong, Y., (2000), Machine selection rules in a dynamic job shop. *International Journal of Advanced Manufacturing Technology*, 16(12), 902-908.

Syswerda, G., (1991), Schedule optimization using genetic algorithms. In: Doris L (eds) Handbook of genetic algorithms. Van Nostrend Reinhold, New York, pp 332-349.

Wang, T.; Wu, K., (2000), A revised simulated annealing algorithm for obtaining the minimum total tardiness in job shop scheduling problems. *International Journal of Systems Science*, 31(4), 537-542.

Yen, B.; Wan, G., (2003), Single-machine bicriteria scheduling: A survey. *International Journal of Industrial Engineering: Theory, Applications and Practice*, 10(3), 222-231.

Zhang, Y.; Luh,; Yoneda, K.; Kano, T.; Kyoya, Y., (2000), Mixed-model assembly line scheduling using the Lagrangian relaxation technique. *IIE Transactions*, 32(2), 125-134.