Int J Syst Assur Eng Manag (November 2017) 8(Suppl. 2):S1596–S1607
DOI 10.1007/s13198-017-0633-5

ORIGINAL ARTICLE

Integrated scheduling of part and tool in a flexible manufacturing system using modified genetic algorithm

Naveen Kumar[1] · Pankaj Chandna[2] · Dheeraj Joshi[3]

Received: 30 January 2017 / Revised: 3 April 2017 / Published online: 27 May 2017

Abstract Scheduling problems in an FMS have been considered as complex optimization problems whose solution by conventional techniques requires a great deal of efforts and time. In this paper, a simultaneous loading and scheduling of part and tool has been proposed for a flexible manufacturing system which has identical machines and a common tool magazine. All the tools are stored in the common tool magazine, and shared among the different machines through a material handling system. Each tool type is single in number. A modified genetic algorithm (MGA) with three parent crossover and a mutation operator is used to find the optimal solution of the loading and scheduling problem. The MGA uses an algorithm which is based on Giffler and Thompson procedure with a heuristic approach to resolve the job conflict and generate an active feasible schedule. The performance of the proposed algorithm is analyzed by comparing the makespan results with the results existing in literature. It is observed that the MGA yields better results than the algorithms reported so far. Furthermore, efficiency of MGA improves as the problem size increases.

Keywords Loading · Scheduling · Flexible manufacturing system · Modified genetic algorithm

Abbreviations

FMS	Flexible manufacturing system
AGVs	Automated guided vehicles
ASRS	Automated storage and retrieval system
PDRA	Priority dispatching rules algorithm
CTM	Central tool magazine
GA	Genetic algorithm
GADG	Genetic algorithm with dominant genes
ACO	Ant colony optimization
PNs	Petri nets
ASMEA	Symbiotic evolutionary asymmetric multileveled algorithm
WIP	Work in process
SAA	Simulated annealing algorithm
FMC	Flexible manufacturing cell

List of symbols

est_{ik}	Earliest start time of kth operation of ith job
eft_{ik}	Earliest finishing time kth operation of ith job
DT	Datum time
N	Number of jobs
K	Number of operations
t_{ik}	Processing time of kth operation of ith job
IP	Initial population size
s	Population size of the selection pool A
J_i	Job number ($i = 1$ to n)
J_{ik}	kth operation of ith job
COJ	Conflict of jobs
M_j	Machine number ($j = 1$ to m)
m	Number of machines
MT	Makespan time
p	Mutation probability
MGA	Modified genetic algorithm
$MAXGEN$	Maximum number of generations for MGA
$P.O.J$	Processed operation of job
JA	Job assigned

✉ Naveen Kumar
agrawal.naveen1@gmail.com

[1] Sir Padampat Singhania University, Udaipur, India

[2] NIT Kurukshetra, Kurukshetra, India

[3] Delhi Technological University, Delhi, India

1 Introduction

To meet the challenge of fast changing demands, the manufacturer must have to produce a large number of product types with minimum lead time. The advent of computer numerical controlled machines has resulted in the development of flexible manufacturing systems (FMS). A Flexible manufacturing system facilitates the production of variety of part types in smaller batch size with minimum setup time. Such system consists of number of general purpose numerically controlled machines, interconnected by automated material handling system and the control is being done by a central computer. Using FMS for production, flexibility of job shops and efficiency of mass production can be enhanced; however, setting up an FMS requires a lot of investments, hence, the efficient solutions to the various decision problems are key to avoid its underutilization (Turkcan et al. 2007).

There are four stages of decision problems for successful installation and implementation of an FMS: designing, planning, scheduling and control (Stecke 1983). Loading and scheduling problem is to allocate parts (or operations) and required tools to machines with exact time span (Roh and Kim 1997). To fully exploit the benefits of an FMS, careful attention must be paid to operational issues like loading, scheduling and control (Gnanavel et al. 2010). Scheduling of part and tool without considering other resources in the system, e.g. material-handling system like automated guided vehicles (AGVs) and automated storage and retrieval system (ASRS), lowers the efficiency and flexibility of FMS production activity. Many FMSs employ automated guided vehicle in material-handling system in order to improve the flexibility and efficiency of its production activity. The operation and control of AGVs is done with the help of well-designed vehicle management system.

FMS loading and scheduling problem has drawn considerable attention of researchers both from academics and industries in past four decades. FMS scheduling is much more complicated than job shop scheduling due to inherent flexibility of FMSs. Literature is replete with variety of scheduling problems concerning machining environment, job description, and objective function (French 1982; Brucker 1995). Part and tool flows in a FMS are two dynamic entities and their management is important for its efficient operation (Prabaharan et al. 2006). Scheduling of parts without considering the tool flow may lead to an inefficient working of FMSs, thereby preventing an FMS reaching to its fullest potential and make it 'inflexible' in practice (Gray et al. 1993; Veeramani et al. 1992; Selim and Ozkan 2001). In most of the existing works on loading and scheduling problems, the part and tool flow are considered as separate issues owing to computation complexity, and often, the effect of one of the pair on the other is neglected (Prabaharan et al. 2006). This might lead to suboptimal solution to the loading and scheduling problem, as solution to one issue affect the solution of other.

Scheduling problem in an FMS is generally solved using any of the two approaches: part movement approach, or tool movement approach (Roh and Kim 1997). In part movement approach, the tools are loaded on the machines and part moves to different machines depending on the availability of the required cutting tool on the machine. In tool movement policy, all the operations of a part are performed on the same machine and the required cutting tools are moved to the machine. The first approach is generally used when the cost of the cutting tools is considered to be insignificant in comparison to the cost of the parts. Most of the research work related to scheduling in an FMS has used the part movement approach (Mukhopadhyay and Nandi 1999). But when the tool cost contributes significantly to the cost of the part; sometimes as high as 25–30% (Selim and Siraceddin 1999); the second approach is being used. Due to limited tool budget, it is essential to find out the tool copy configuration for the best system performance (Jun et al. 1999). Therefore, by adopting appropriate tooling strategies and economizing on the tooling cost, large reduction on the tooling cost are feasible (Gray et al. 1993). In tool flow approach, automated transport of tools to the machines has been considered. The added cost of tool automation improves the economic efficiency of an FMS (Gray et al. 1993). Although the scheduling problems have been studied under the part movement policy extensively in many papers, there are relatively few articles on the problems under the tool movement policy (Roh and Kim 1997). A few researchers in their surveys on the tool management issues of automated manufacturing systems established that the lack of tooling considerations has resulted in the poor performance of these systems and stressed for tool scheduling (Gray et al. 1993; Veeramani et al. 1992). Most of the existing work on loading and scheduling in an FMS solve the problem sequentially, i.e. first the loading problem is solved, and based on the results obtained from the solution of loading problem, scheduling problems are solved. This approach does not give the optimal solution to the loading and scheduling problem, as the solution of loading problem becomes a limitation for the scheduling problem (Kim et al. 2007). Considering these aspects, in this paper the part scheduling and tool allocation problem has been solved simultaneously. The concept of common tool magazine (CTM) that shares with and serves for several machines, reduce the cost of duplicating tools in each and every machining centre, is of particular interest in each FMS.

Scheduling of parts with identical machines in an FMS even without additional resources has been considered to be NP-hard and requires heuristic approach for solution (Agnetis et al. 1997). An FMS scheduling problem is generally solved using optimization and heuristic based methods, however, Optimization methods, e.g. branch and bound, integer programming, dynamic programming etc., become cumbersome to solve as the problem size increases. Therefore, efficient heuristic methods need be developed for large sized problems. Priority dispatching rule algorithms (PDRA) are the simple heuristics which can be easily applied to any FMS scheduling problem. Giffler and Thompson (GT) algorithm (Giffler and Thompson 1960) developed for the job shop manufacturing environment can also provide quality results for an FMS scheduling problem (Nasciment 1993). Prabaharan et al. (2006) used combined PDRA and simulated annealing algorithm for the scheduling and sequencing of parts and tools in a flexible manufacturing cell (FMC). Udhayakumar and Kumanan (2010) used Ant colony optimization (ACO) algorithm to schedule the tools and parts with the objective of minimizing the makespan in an FMS having identical machines. Fathi and Barnette (2002) solved the scheduling problem of part and tool in an FMS with identical machine environment in order to minimize makespan using three heuristic methods.

Artificial Intelligence techniques have drawn considerable attention of researchers for solving scheduling problem in FMS. One of such intelligent probabilistic search technique is genetic algorithm (GA) (Holland 1975). According to Goldberg (1989), GAs can be applied to treat the complexity levels required to provide adaptive search at the requisite robustness. Genetic algorithm has been successfully used to solve verities of optimization problems including problems related to manufacturing, i.e. scheduling, process planning and system design. Ponnambalam et al. (2001) stated that GA is the most popular type of evolutionary algorithm to solve FMS scheduling problems. Keung et al. (2003) in their work proposed a GA to solve the sequencing problem in an FMS having one material handling device, with an objective to minimize the penalty cost. Shankar et al. (2005) designed multi-objective evolutionary algorithm equipped with a mechanism to generate parallel diverse optimal solutions, for scheduling of an FMS. Chan et al. (2008) used Genetic Algorithm with Dominant Genes (GADG) to solve the scheduling problem in an FMS. The results obtained for the minimization of makespan, were compared with results obtained by other techniques like Ant Colony Optimization and Petri Nets (PNs). The results obtained by the proposed algorithm were found to be better. Kim et al. (2007) considered an FMS with four types of flexibility: machine, tools, processing and sequencing. They used Symbiotic Evolutionary Asymmetric Multileveled Algorithm (ASMEA) to solve the scheduling problem in the said FMS environment. They reported that the results obtained were of high quality and the speed of convergence of the algorithm was also fast. Hsu et al. (2008) solved sequencing problem of cyclic tasks in an FMS by using GAs and PNs. They selected work in process (WIP) as performance parameter. They concluded that the results obtained by GA in 75% of the cases were equivalent to that of obtained by best heuristics available in the literature. Gang and Wu (2004) used hybrid approach (GA and PN) to solve sequencing problems in FMS. Reddy and Rao (2006) used GA and a heuristic for simultaneous scheduling of machines and AGVs (automated guided vehicles). Balin (2011) used GA to solve scheduling problem with non-identical parallel machines. The author used new crossover operator and optimality criteria for minimization of makespan. Godinho et al. (2014) in their work presented a comprehensive review of the available literature on GA applied to FMS scheduling. The analysis of the literature is done based on the proposed six classifications. Kaplanoglu (2016) in his work proposed an object oriented approach along with simulated annealing optimization algorithm for multi-objective flexible job-shop scheduling problem. Wu et al. (2017) in his work proposed genetic algorithm with a new chromosome representation scheme to solve the distributed flexible job-shop scheduling problem.

In GA evolutionary process; crossover and mutation are the important search operators. Most of the work reported on application of GAs in FMS scheduling problem, have considered two parent crossover method. Elsayed et al. (2014) proposed a new GA with multi-parent crossover with a diversity operator. They tested this algorithm on number of optimization problems and the results show its better performance.

This work proposes an FMS environment consisting of 'm' identical machines with a CTM. In case, any two or more machines require same tool at the same time, one of the machine is served with the tool in order to resolve the job conflict, while the other machines have to wait, till the tool completes the job on the assigned machine. The makespan has been used as performance measure in this study, because, it represents a long term or steady state performance measure for many manufacturing systems. In this work, we have used the GA having three-parent crossover with a mutation operator.

The paper is structured as follows: the formulated problem is described in Sect. 2. In Sect. 3 a new algorithm Active Schedule (ACT_SCH) and MGA are discussed. Discussion of results and sensitivity analysis is presented in Sect. 4. Finally, the conclusions are given in Sect. 5.

2 Problem structure

In many manufacturing systems tools are stored at a common place called common tool magazine. These tools are shared with machines in the system, through a material handling system in order to reduce the tool inventory. In this paper, problem of simultaneous scheduling of tools and part has been attempted for a flexible manufacturing system with a common tool magazine with an objective to minimize the makespan. Description of the problem environment, assumptions and performance measure are discussed in the section below.

2.1 FMS environment

The FMS in the problem has identical machines. The machines are capable of processing a group of parts and part verity. None of machines has dedicated tool magazine. All the tools are stored in the CTM and shared among all the machines through single tool grip. There is a limitation on the quantity of each tool type in CTM and it is kept as one in this problem. The FMS environment is shown in Fig. 1.

2.2 Assumptions

- Machines are capable of doing all operations of any of the jobs.
- Each job has a number of operations and individual operation requires a specific tool.
- Sequence of operations and respective tools vary from job to job.
- The operations sequence and type of tool required for each operation with processing time of operations are pre-specified.
- One job can be processed on one machine/tool at a time.
- All the operations of a job are processed during a single machine visit.

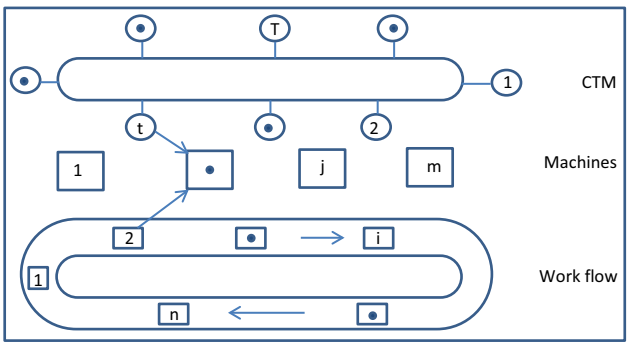

Fig. 1 Working environment of FMS

- Operation cannot be interrupted, i.e., each operation once started must be completed.
- A job does not visit the same machine twice.
- Availability of tools in each verity is considered as one and can be used more than once for any job.
- The operation time of a job includes the loading, unloading, tool changeover and setup times (both tool and job) along with the processing time.
- Once the operation is over the tool returns to the common tool magazine with negligible transfer time and is available for the next operation.

2.3 Performance measure

Considering the high initial cost of investment in FMSs, it is important for the manufacturer to maximize the machine utilization. In this context, minimum makespan that increases the utilization of machines is considered as the appropriate measure of system performance.

The formulae used for makespan are given as below:

Job completion time

$$C_i = \sum_{k=1}^{K} t_{ik} \tag{1}$$

$$\text{Makespan} = \max(C_1, C_2, \ldots, C_n) \tag{2}$$

where k = operation, i = job, t_{ik} = operation processing time

2.4 Problem statement

Determination of integrated schedule of tool and job which gives optimal or near optimal makespan, considering that n jobs are processed on m identical machines with T number of tool types are shared for many operations, in a flexible manufacturing system.

3 Proposed heuristics

MGA heuristic is used to find the optimal or near optimal solution for the proposed problem. The above algorithm uses a new algorithm ACT_SCH which is developed by modifying GT algorithm, for generating active feasible joint operation-tool schedule. The new algorithm ACT_SCH is described as below:

3.1 ACT_SCH algorithm for generating active feasible schedule

GT algorithm is basically used for generating active feasible schedule for a job shop scheduling. GT algorithm is a mathematical version of procedure for drawing Gantt

chart to arrive at active feasible schedule. The proposed new algorithm ACT_SCH is developed by using the similar procedure to GT algorithm. Whenever two or more jobs require the same tool at the same time (job conflict), the tool is assigned to a job that has minimum datum time (DT) for the waiting operation. Through this job conflict resolving policy, the waiting time of other jobs for the same tool reduces and hence an active schedule with lower makespan is obtained. The procedural step of ACT_SCH is explained in following Sect. 3.1.1; the flowchart is shown in Fig. 2 and illustrated in Sect. 3.1.2.

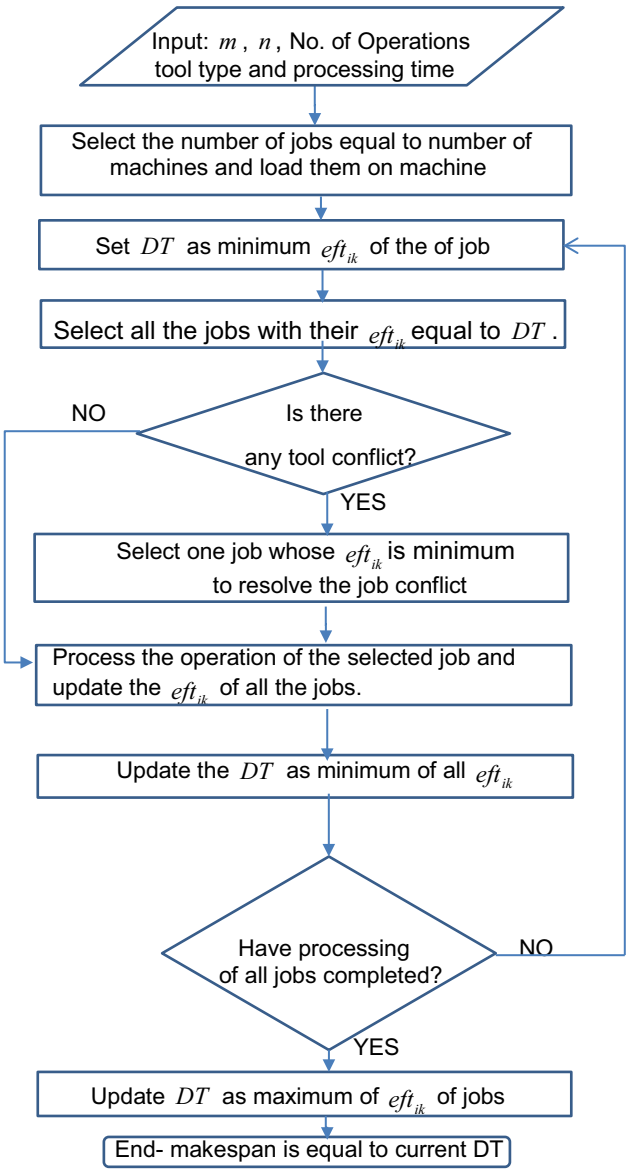

Fig. 2 Flow chart for ACT_SCH algorithm

3.1.1 Procedural steps of ACT_SCH

Step 1 Construct a table having columns one for each machine, job assigned (JA), datum time and processed operation of a job (POJ). The column for each machine is further subdivided for each tool type

Step 2 Select randomly the number of jobs equal to the number of machines and assign one job to one machine arbitrarily. Enter a value equal to sum of earliest start time (est_{ik}) and processing time (t_{ik}) in the first line of the table in appropriate block of machine under appropriate tool. The total of the two values gives earliest finish time (eft_{ik}) of the immediately waiting operation of the assigned job. For the first operation of first job assigned on a machine the value of est_{ik} is taken to be zero

Step 3 Enter the value of DT equal to the smallest of the eft_{ik} entries, in the appropriate column

Step 4 Select the jobs whose eft_{ik} matches with the current datum time. In case there are more than one jobs whose eft_{ik} matches with current datum time; select one job arbitrarily

Step 5 Check for the job conflict? If yes, go tostep 6 else go to step 7

Step 6 List the operations of all the jobs contending for the same tool. Select a job whose eft_{ik} is minimum and earmark its operation. Assign the tool to the earmarked operation of the job

Step7 Process the earmarked operation along with its machine and tool. Enter the job number and operation completed in the format of 'i_k' in the column "*P.O.J*"

Step 8 Update the value of eft_{ik} of next waiting operation as sum of earliest finishing time of immediate processed operation and processing time of the waiting operation. Also, update the eft_{ik} of the contending operation of other jobs (who are not assigned with the required tool during conflict resolution) as sum of earliest finish time of the operation of the job assigned with the tool plus its processing time

Step 9 Update the datum time as minimum of earliest finishing time. Check, if all the operations of all the jobs are scheduled. If yes, go to step 10, else go to step 4

Step 10 Update the DT as largest entry of earliest finishing time. The table gives the active feasible schedule and DT represents the makespan time

From the table the est_{ik} and eft_{ik} of each job can be read and subsequently the Gantt chart can be drawn.

3.1.2 Numerical example

A numerical example of scheduling problem of three jobs on two machines with three tool types is taken to explain the detailed procedure of the proposed algorithm (ACT_SCH). The sequence of jobs is taken as 2–1–3. The problem is given in the Table 1 and the procedure for generation of active feasible schedule is shown in the Table 2.

In row number 4 in Table 2, the operation number 3 of job number 2 is waiting on machine number 1 and operation number 2 of job number 1 is waiting on machine 2 to be processed and both require same tool type 1. This is a condition of conflict of job (COJ) and is resolved by selecting job 1 to be supplied with the tool as the earliest finish time of its operation is the least among all the contending jobs. Now, the operation 3 of job 2 has to wait till the tool type 1 gets free from the assigned job.

Table 1 Job-operation-tool matrix

Job no.	Operation no. (i)		
J	1	2	3
1	4 (2)[a]	10 (1)	8 (3)
2	6 (3)	5 (2)	6 (1)
3	7 (1)	6 (3)	8 (2)

[a] The value in parenthesis represents tool type for the processing of an operation

3.2 Modified genetic algorithm (MGA)

Genetic algorithms are stochastic search techniques that rely on the process of natural selection (Goldberg 1989). To start with simple GA, A set of random solutions called initial population, is generated. Each chromosome in this population contains information about a possible solution. The fitness of each chromosome is calculated based on the objective function. The chromosomes with higher fitness value are pooled in a group and are used for crossover. Mutation operator is applied to the offspring obtained through crossover, in order to maintain diversity of solution. Select the number of parents and offspring with higher fitness value, equal to the population size for next round of breeding and mutation. The process is repeated until optimal or near optimal solution is achieved.

It is important to control the distribution of offspring in comparison to their parents in order to improve the performance of GAs. The narrow distributions of offspring lead to premature converge of the algorithm, whereas wider distributions of offspring result in the algorithm taking a

Table 2 Active schedule generation procedure

Machine number						Job Processed		DT	COJ	P.O.J
1			2			Machine 1	Machine 2			
Tool type			Tool type							
1	2	3	1	2	3					
		0 + 6 6		0 + 4 4*		2	1	4	1	1_1
		0 + 6 6*	4 + 10 14			2	1	6	3	2_1
	6 + 5 11*		4 + 10 14			2	1	11	3	2_2
11 + 6 19			4 + 10 14*			2	1	14	2.1	1_2
14 + 6 20*					14 + 8 22	2	1	20	2	2_3
20 + 7 27					14 + 8 22*	3	1	22	1	1_3
20 + 7 27*						3	–	27	3	3_1
		27 + 6 33*				3	–	33	3	3_2
	33 + 8 41*					3	–	41	3	3_3

*Jobs selected for processing

long time to converge to the optimal solution. In this context, the research paper proposes a modified GA with three-parent crossover and a diversity operator. Three parents crossover produces three offspring by linear combination of three parents to improve the fitness and diversity of the offspring. The mutation operator further diversifies the generated offspring.

Initial population of chromosomes of size IP equal to $(n^2 - n)/2$ is generated. Each chromosome in this population contains numbers from 1 to j which are randomly generated by permutation. The structure of a chromosome containing five jobs is shown in the Fig. 3.

Each number in the chromosome signifies a job number and the placement of numbers in the chromosome from left to right represents their sequence of assignment on *m* machines. For instance, if there are 2 machines then first two jobs; 5 and 3 are assigned to machine 1 and 2 respectively. The next job in the sequence is assigned to a machine which becomes free after processing the previous job.

From the generated initial population *s* best solutions are selected and stored in an archive pool A. Now using tournament selection (with size 2), the chromosomes are selected and stored in the selection pool B with a pool size of 3**IP*. Three consecutive parents from the selection pool are selected and are arranged in increasing value of fitness i.e. $f(P_1) \leq f(P_2) \leq f(P_3)$. Any duplicity of the chromosomes is removed by replacing the unwanted chromosomes with a random chromosome from the selection pool. The crossover among these three chromosomes P_1, P_2 and P_3 is performed to generate three offspring. The procedure to generate offspring O_1 is explained in Fig. 4a, b. The whole process of crossover operation is divided into two steps. In the first step, as illustrated in Fig. 4a, parent chromosome P_2 crossover with P_3 using the randomly generated crossover vector x_1, to produce intermediate offspring IM_1. In this step, firstly the resultant parent P_2x_1 is obtained by replacing all non-zero elements of x_1 by the corresponding elements of parent P_2. Then all the zero elements in the resultant chromosome are replaced by the corresponding elements of second parent P_3 in order to generate intermediate offspring IM_1. In case, if any of the replacement of zero elements results in repeat of numbers in IM_1, then it has to be replaced by a non-repeating element of the first chromosome i.e. P_2. As shown in Fig. 4a the element 3 of chromosome P_3 cannot replace 0 element in P_2x_1, as number 3 already exist in the resultant chromosome. Hence, the first element of chromosome P_2 corresponding to zero element i.e. 4 is considered for the replacement. But this number would also cause duplicity, and therefore the next element in P_2 corresponding to 0 i.e. 5, is considered. Since it does not create duplicate entry in the resultant chromosome, therefore number 5 is used instead of number 3 to replace the 0 element in the resultant chromosome P_2x_1. The proposed method of crossover results into a feasible intermediate offspring IM_1. In the second step, third parent P_1 crossover with IM_1 as illustrated in Fig. 4b, to generate offspring O_1.

To obtain offspring O_2 and O_3 first of all, intermediate offspring IM_2 and IM_3 are generated by crossover of P_2

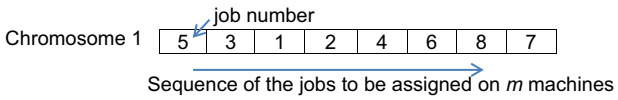

Fig. 3 Structure of a chromosome

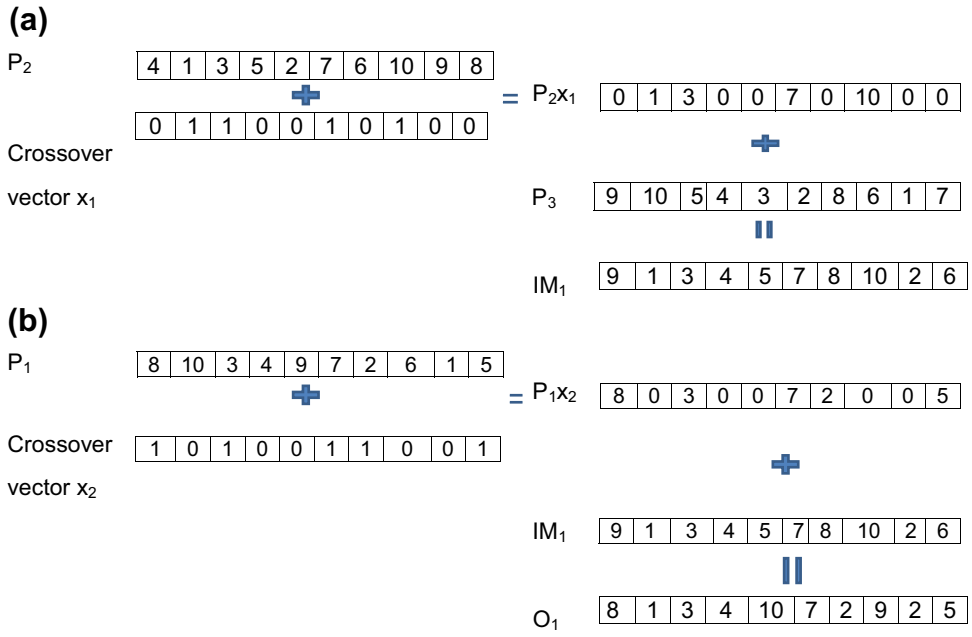

Fig. 4 **a** Step-I: parent chromosomes crossover using crossover vector x_1 to generate intermediate offspring IM_1. **b** Step-II: IM_1 and third parent chromosome P_1 crossover using crossover vector x_2 to generate final offspring O_1

with P_3 and P_3 with P_1 respectively, subsequently P_1 crossover with IM_2 and P_2 crossover with IM_3 to generate offspring O_2 and O_3 respectively.

In three parent crossover mechanism, it is observed that offspring O_1 and O_3 move towards better fitness while offspring O_3 diversify the population.

A mutation operator is used to diversify the population to help the algorithm to escape from local minima. In this, randomly selected two elements of a selected offspring are exchanged. The probability of mutation operation is taken as p. The offspring and archive pool individuals are then merged and the best s individuals are selected as new population for the next generation.

Various parameters for the MGA, like $s = 0.5 * IP$, $p = 0.05$, $MAXGEN = 100$ and crossover probability $= 1$ are decided based on the trial runs of modified Genetic Algorithm on five different scheduling problems from the literature (Prabaharan et al. 2006) for each case of $m = 2$, $m = 3$ and $m = 4$. The results are compared with the optimum value obtained by complete enumeration of the problems. It is observed that optimum results are obtained for 43.33% of the problems and 50% of the problem shows deviation <8% from optimum value.

The procedural step of Modified GA is explained in following Sect. 3.2.1; the flowchart is shown in Fig. 5.

3.2.1 Steps of modified GA

Step 1 Set $p = 0.05$, $IP = (n^2 - n)/2$, $MAXGEN = 100$ and $s = 0.5 * IP$

Step 2 An initial population of size IP is generated by random permutation of number of jobs j

Step 3 An archive A is filled with s best solutions

Step 4 Using the tournament selection process with size $TC = 2$, a selection pool B is filled with selected chromosomes with size $3IP$

Step 5 Select three consecutive chromosomes from the selection pool and arrange them so that $f(P_1) \leq f(P_2) \leq f(P_3)$. Replace any duplicate chromosome with randomly selected chromosome from the selection pool

Step 6 Crossover of three parents is performed to obtain three offspring

Step 7 Mutation operation of the selected offspring is performed

Step 8 Generate the active feasible schedule from the sequence of jobs in chromosomes by using ACT_SCH algorithm and calculate the makespan time

Step 9 Combine chromosomes from archive pool and offspring. Select the best individuals to make a new population of size IP

Step 10 Check, if termination criterion is met? If yes go to step 12 Otherwise go to step 2

Step 11 Stop and note down the optimal sequence and corresponding optimal makespan.

4 Results and discussions

To obtain the test results for makespan time for the proposed joint scheduling problem a set of 20 problems, each for $m = 2$, $m = 3$ and $m = 4$, are taken from the literature (Prabaharan et al. 2006). The results obtained for makespan

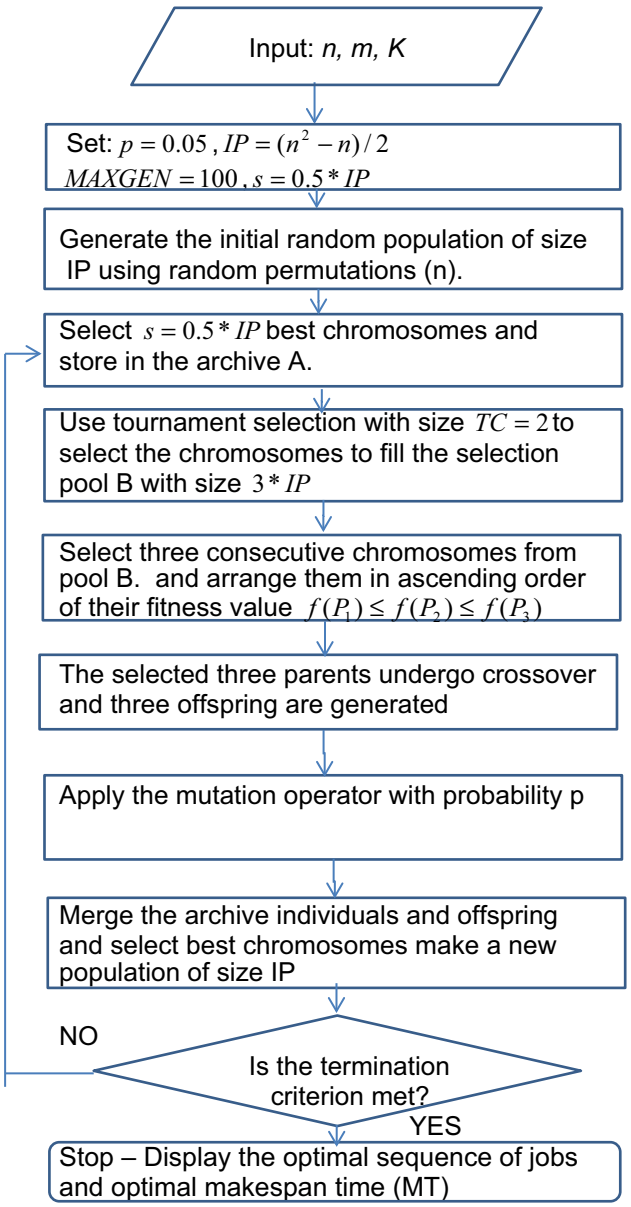

Fig. 5 Flowchart for MGA algorithm

time using proposed Modified GA are compared with the results reported in the literature in Tables 3, 4 and 5.

The results shown in fourth column of the tables represents the best value of makespan obtained by two techniques, namely PDRA and Simulated Annealing Algorithm (SAA), in the literature (Prabaharan et al. 2006), whereas the results reported by Udhayakumar and Kumanan (2010) are shown in the fifth column. The results obtained by the proposed MGA are shown in the last column. The best value of makespan time is marked with boldface. The computational time taken by the proposed MGA is shown in the Fig. 6.

The percentage difference from best (PDB) is calculated to measure the performance of the proposed algorithm in giving better results comparison to best results reported in the literature.

$$PDB = (MT_{best} - MT_{MGA}) * 100/MT_{best}$$

where, MT_{best} is the best makespan value by the other reported algorithms and MT_{MGA} is the value obtained by MGA.

The results show that the MGA algorithm gives better results for all the problems except for the problem number 3 and 14 for $m = 2$, and problem number 19 for $m = 4$. The average percentage difference calculated are 1.95% for $m = 2$, 5.25% for $m = 3$ and 7.29% for $m = 4$. The results also show that the average percentage difference increases with increase in number of machines which represents that the performance of proposed MGA increases with increase in problem size. The programming for ACT_SCH algorithm and MGA was developed using "MATLAB". The computational time taken by MGA is reasonable. The computational time increases exponentially with higher number of jobs. But the higher time does not delay the implementation of solution as the scheduling problems can be solved offline prior to the actual loading of the batch of jobs.

5 Conclusions

The application of MGA to the FMS loading and scheduling problem is explored in this paper. The FMS environment consists of several identical machines without tool magazines. A CTM stores all types of tools and shares the tools among the machines through the tool transport system. The Modified GA uses the principle of evolution to find the optimal solution of the scheduling problem. The three parent crossover generates better fitness and diversified offspring. Use of diversity operator helps the algorithm to avoid local minima and premature convergence.

In the proposed MGA, initial population is generated randomly and the best *s* individuals based on fitness values are stored in archive pool. Tournament selection with size 2 was used to select the individuals of size 3 * *IP* for the reproduction. Three consecutive individuals are selected

Table 3 Comparison of results obtained for m = 2

Problem no.	No. of jobs	No. of operations	Best solution PDRA and SAA	Solution by ACO	Proposed MGA algorithm
1	3	3	**38**	**38**	**38**
2	4	3	**99**	**99**	**99**
3	5	4	111	**103**	107
4	6	4	131	128	**126**
5	7	5	212	210	**208**
6	8	7	300	300	**293**
7	9	5	242	243	**208**
8	10	6	402	394	**385**
9	11	5	297	294	**282**
10	12	4	257	265	**248**
11	13	5	703	693	**669**
12	14	6	542	540	**525**
13	15	7	339	340	**334**
14	16	8	332	**328**	332
15	17	8	349	350	**344**
16	18	8	399	419	**397**
17	19	9	482	476	**468**
18	20	9	532	527	**524**
19	25	9	643	643	**635**
20	30	9	797	813	**795**

Table 4 Comparison of results obtained for m = 3

Problem no.	No. of operations	PDRA and SAA	Best solution	Solution by ACO	Proposed MGA algorithm
1	3	3	**25**	**25**	**25**
2	4	3	**90**	91	**90**
3	5	4	83	83	**78**
4	6	4	102	100	**95**
5	7	5	168	160	**153**
6	8	7	227	221	**216**
7	9	5	200	193	**168**
8	10	6	299	290	**280**
9	11	5	220	219	**202**
10	12	4	203	204	**182**
11	13	5	540	524	**475**
12	14	6	402	403	**367**
13	15	7	247	239	**233**
14	16	8	238	256	**232**
15	17	8	253	246	**242**
16	18	8	291	320	**272**
17	19	9	344	338	**324**
18	20	9	392	387	**364**
19	25	9	450	**441**	445
20	30	9	583	614	**556**

Table 5 Comparison of results obtained for m = 4

Problem no.	No of jobs	No. of operations	Best solution PDRA and SAA	Solution by ACO	Proposed MGA algorithm
1	3	3	–	–	–
2	4	3	87	87	**70**
3	5	4	78	79	**73**
4	6	4	90	87	**85**
5	7	5	142	136	**135**
6	8	7	192	181	**168**
7	9	5	158	149	**146**
8	10	6	270	258	**232**
9	11	5	196	182	**164**
10	12	4	187	187	**157**
11	13	5	457	448	**407**
12	14	6	328	318	**308**
13	15	7	210	198	**191**
14	16	8	200	189	**183**
15	17	8	217	203	**189**
16	18	8	247	238	**217**
17	19	9	271	259	**255**
18	20	9	322	316	**288**
19	25	9	366	**351**	355
20	30	9	464	452	**446**

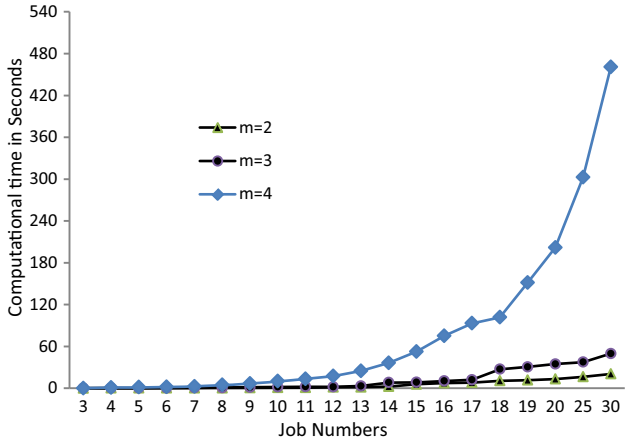

Fig. 6 Computational time taken by MGA

for reproduction by using the crossover operator. The mutation operator is then used to diversify the offspring. The offspring and archive pool individuals are then merged and the best s individuals are selected as new population for the next generation.

The ACT_SCH algorithm is used to generate the active feasible schedule. The results obtained for the makespan by the MGA are compared with the results reported in the literature and it was found that the proposed MGA algorithm gives optimal or near optimal results in almost all the test problems. The computation time taken by the MGA for $m = 2$ and $m = 3$ cases is comparable to that reported for ACO and SAA in the literature, but the same is higher for $m = 4$ case. Since the scheduling problems can be solved offline before the actual production starts, higher computational time does not delay the implementation of solution.

References

Agnetis A, Alfieri A, Brandimarte P, Prinsecchi P (1997) Joint job/tool scheduling in FMC with no on-board tool magazine. Comput Integr Manuf Syst 10(1):61–68

Balin S (2011) Non-identical parallel machine scheduling using genetic algorithm. Expert Syst Appl 38(6):6814–6821

Brucker P (1995) Scheduling algorithm, 1st edn. Springer, Berlin

Chan FTS, Chung SH, Chan LY (2008) An introduction of dominant genes in genetic algorithm for FMS. Int J Prod Res 46(16):4369–4389

Elsayed SM, Sarkar SA, Essam DL (2014) A new genetic algorithm for solving optimization problems. Eng Appl Artif Intell 27:57–69

Fathi Y, Barnette KW (2002) Heuristic procedure for parallel machine problem with tool switches. Int J Prod Res 40(1):151–164

French S (1982) Sequencing and scheduling, 5th edn. Hardwood, London

Gang X, Wu Z (2004) Deadlock-free scheduling strategy for automated production cell. IEEE Trans Syst Man Cybern Part A Syst Hum 34(1):113–122

Giffler B, Thompson GL (1960) Algorithms for solving production scheduling problems. Int J Oper Res 8:487–503

Gnanavel BA, Jerald J, NoorulHaq A, Muthu Luxmi M, Vigneswaralu TP (2010) Scheduling of machines and automated guided vehicles in FMS using differential evolution. Int J Prod Res 48(16):4683–4699

Godinho FM, Barco CF, Neto RFT (2014) Using genetic algorithms to solve scheduling problems on flexible manufacturing systems (FMS): a literature survey, classification and analysis. Flex Serv Manuf J 26(3):408–431

Goldberg DE (1989) Genetic algorithm in search, optimization, and machine learning. Addition-Wesley, Boston, MA

Gray AE, Seidmann A, Stecke KE (1993) A synthesis of decision models for tool management in automated manufacturing. Manag Sci 39:549–567

Holland HH (1975) Adaption in natural and artificial systems. University of Michigan Press, Detroit, MI

Hsu T, Korbas O, Dupas R, Gonclaves G (2008) Cyclic scheduling of FMS: modeling and evolutionary solving approach. Eur J Oper Res 191:464–484

Jun HB, Kim YD, Suh HW (1999) Heuristics for a tool provisioning problem in a flexible manufacturing system with an automatic tool transporter. IEEE Trans Robot Autom 15(3):488–496

Kaplanoğlu V (2016) An object-oriented approach for multi-objective flexible job-shop scheduling problem. Expert Syst Appl 45:71–84

Keung KW, Ip WH, Yuen D (2003) An intelligent hierarchical workstation control model for FMS. J Mater Process Technol 139:134–139

Kim YK, Kim JY, Shin KS (2007) An asymmetric multileveled symbiotic evolutionary algorithm for integrated FMS scheduling. J Intell Manuf 18(6):631–645

Mukhopadhyay SK, Nandi PK (1999) Solving tool allocation problem in flexible manufacturing system. J Ind Eng (India) 80:16–20

Nasciment MA (1993) Giffler and Thompson algorithm for job shop scheduling is still good for flexible manufacturing systems. J Oper Res Soc 44(5):521–524

Ponnambalam SG, Aravindan P, Srinivasa Rao P (2001) Comparative evaluation of genetic algorithms for job-shop scheduling. Prod Plan Control 12(6):560–574

Prabaharan T, Nakkeeran PR, Jawahar N (2006) Sequencing and scheduling of job and tool in a flexible manufacturing cell. Int J Adv Manuf Technol 29(7–8):729–745

Reddy BSP, Rao CSP (2006) A hybrid multi-objective GA for simultaneous scheduling of machines and AGVs in FMS. Int J Adv Manuf Technol 31:602–613

Roh HK, Kim YD (1997) Due date based loading and scheduling methods for a flexible manufacturing system with an automatic tool transporter. Int J Prod Res 35(11):2989–3003

Selim AM, Ozkan S (2001) Integrated scheduling and tool management in flexible manufacturing systems. Int J Prod Res 39(12):2697–2722

Selim AM, Siraceddin O (1999) Joint lot sizing and tool management in a CNC environment. Comput Ind 40:61–75

Shankar S, Ponnamabalam SG, Gurumarinuthu M (2005) Scheduling flexible manufacturing systems using parallelization of multi-objective evolutionary algorithms. Int J Adv Manuf Technol 30:279–285

Stecke KE (1983) Formulation and solution of non-linear integer production planning problem for flexible manufacturing system. Manag Sci 29(3):273–288

Turkcan A, SelimAkturk M, Storer RH (2007) Due date and cost-based FMS loading, scheduling and tool management. Int J Prod Res 45(5):1183–1213

Udhayakumar P, Kumanan S (2010) Sequencing and scheduling of job and tool in a flexible manufacturing system using ant colony optimization algorithm. Int J Adv Manuf Technol 50:1075–1084

Veeramani D, Upton DM, Barash MM (1992) Cutting tool management in computer integrated manufacturing. Int J Flex Manuf Syst 4(3–4):237–265

Wu MC, Lin CS, Lin CH, Chen CF (2017) Effects of different chromosome representations in developing genetic algorithms to solve DFJS scheduling problems. Comput Oper Res 80:101–112