SCHEDULING OPTIMISATION IN FMS USING GENETIC ALGORITHM WITH MAKESPAN CRITERION

Syed Mustafa Kazim[1], Amit Kr. Singh[2], Dr. S.C. Jayswal[3]
[1,2,3]Department of Mechanical Engineering,
Madan Mohan Malviya University of Technology, Gorakhpur, U.P. (India)

Abstract: Flexible Manufacturing System is able to produce a large variety of products while maintaining a large volume of production. This is the need of the hour as the contemporary market is largely consumer driven and they are demanding variety of products to choose from. This work is considering objective of minimizing the Makespan time in scheduling of the FMS system. Codes are developed in C++ for getting optimum sequence of operation to reduce the machine idle time and maximize the machine utilization. The FMS system considered in this work have 3 CNC Machine tools for processing 8 varieties of products. After 100 generations of Genetic Algorithm (GA) the global optimum schedule is obtained.

Keywords: FMS: Flexible Manufacturing System, GA: Genetic Algorithm, AGV: Automatic Guided Vehicle, Scheduling Optimization, C++ codes for GA, Pheno type GA.

I. INTRODUCTION

Flexible Manufacturing System is a system of machines that is totally in contrast to hard automation system, they provide a flexible enough system to cater to the changing needs of the industry whereby giving nearly the same advantages as that of the fixed or hard automation system. The machine scheduling problem in a FMS is specified as to assign the machine, operations of selected jobs, and the tools necessary to perform these operations by satisfying the technological constraints (available machine time and tool slots constraint)in order to ensure the minimum Makespan time, system unbalance and maximum throughput when the system is in operation. An attempt has been made to solve the objective function simultaneously to bring the outcomes in close proximity to the real assumption of the FMS environment.

II. LITERATURE REVIEW

N. Jawahar (1997) discussed about Genetic Algorithm based scheduler for setup constrained FMC in which Makespan time criterion has been considered as a measure of performance [1]. Machine loading problem for a random FMS with multiple machines and fixed number of tool slots was carried out by A. Kumar (2005). They proposed a constraint based GA which avoided getting trapped at local minima [2]. S.G. Ponnambalam (2008) addressed the problem of solving machine loading in FMS using Particle Swarm Algorithm; the objective function of the scholarly work was to minimize system unbalance [3]. Scheduling of AGVs and machines in FMS using SFHA-Sheep Flock Heredity Algorithm was addressed by K.V. Subbaiah (2009). They were able to minimize the Makespan and mean tardiness of the FMS system [4]. Multiobjective task scheduling of AGV in an FMS was taken up by P. Udhayakumar (2010). They implemented GA and Ant Colony Optimization technique to minimize travelling time of the AGV [5]. N.M. Nidhiry and Dr. R. Saravanan (2012) carried out research on scheduling optimization of FMS, 43 varieties of products were considered software in .net language was developed for getting the optimum solution [6]. A research on Evolutionary Apporaches for scheduling a FMS system with AGV and robot was done by C. Anandaraman (2012). They used SFHA and Artificial Immune System Algorithms to derive optimum solution for scheduling of machine, the result of the two algorithms is also compared [7]. N.M. Nidhiry addressed the problem of FMS scheduling optimization using modified NSGA-II. They used 80 part sets to optimize Penalty cost and minimize machine Idle Time [8].

III. PROBLEM DESCRIPTION

The Loop Layout considered is consisting of three machines and one AGV. The layout is taken as input for the scheduling of machines. The loading unloading station is connected to ASRS (Automatic Storage and Retrieval System). The objective is to minimize Makespan.

A. FMS Environment

The environment within which the FMS under consideration can be described as follows [7]:

- The types and number of machines are known. Operations are non-preemptive. There is sufficient input / output buffer space at each machine.
- Processing, set-up, loading and unloading times are available and are deterministic.
- Number of AGVs is given and all are identical in the sense that they have characteristics.
- Flow path layout is given and travel times on each segment of the path are known.
- A Load /Unload (L/U) station serves as a distribution centre for parts not yet processed and as a collection centre for parts finished. All vehicles start from L/U station initially and return to thereafter accomplishing all their assignments. There is sufficient input/output buffer space at the (L/U) station.
- AGVs carry a single unit-load at a time. They move along predetermined shortest paths. Pre-emption of

trips is not allowed. The trips are called loaded or deadheading (empty) trips depending on whether a part is carded or no part is carded during that trip, respectively. The durations for the deadheading trips are sequence dependent and are not known until the vehicle route is specified.

- Robots can handle only single unit-load at a time. They move along specially constructed tracks between the machines. Two kinds of motion are possible for the robots. They can move along the tracks to carry a job from one machine to another. Once the robot reaches a machine, it can transfer the job from the machine to AGV or vice versa or to itself using a combination of translatory and rotary motions. The duration of robot travel from one machine to other and the time taken for robot to transfer a job from/to machine and AGV are known.
- It is assumed that all the design and set-up issues within the hierarchy of OR/MS problems in an FMS as suggested by Stecke and Solberg (1981) have already been resolved. Machine loading is made. Pallets and other necessary equipment are allocated to parts.
- Ready times for all jobs are known.
- Issues of machine failure or downtime, scraps, rework, robot, AGV repair and maintenance are ignored and left as issues to be considered during real time control.

IV. OBJECTIVE FUNCTION

A. Minimizing Makespan

Makespan

Job completion time (Ci) = $\sum_{j=1}^{n} O_{ij}$ (1)

Makespan = Max (C1, C2, C3,…..……Cn) (2)

Operation completion time = Oij = Tij + Pij (3)

Mean Flow Time

Mean Flow Time (Fi) = $^{1}/_{n} \sum_{i=1}^{n} C_i$ (4)

where

i : job number

j : operation number

n : number of jobs

O_{ij} : Time taken for jth operation of ith job

T_{ij} : Total travelling time for ith job before jth operation

P_{ij} : Total processing time for ith job and jth operation

V. INPUT DATA

The input data i.e. travelling time matrix and job sets for the problem is taken from Bilge and Ulusoy (1995). The time taken for a job to be transferred from one station to another through AGV is expressed in the transfer time matrices given in Table I. Data for the job sets used in the example problems are given in Table II. The number after the letter 'M' denotes the machine number and the corresponding processing time of the job on that machine is given in the table. The distance from load/ unload station to machines and distances between a pair of machines are given in meters for the given layout.

A. VEHICLE SCHEDULING METHODOLOGY

Machines are scheduled based on the operation sequence derived by the proposed Algorithms. Initially AGVs carry jobs from the load/ unload station to the respective workstations where the first operations are scheduled. AGVs perform two types of trips, a loaded trip where it carries a load and a deadheading trip where the vehicle moves to pick up a load. Deadheading trip can start immediately after the delivery and vehicle demand at different workstations are considered and the subsequent assignments are made. This type of vehicle scheduling methodology helps in reducing the waiting times and thus helps in improving the resource utilization.

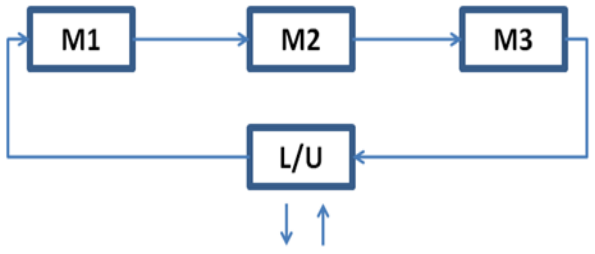

Fig 1: Loop Layout

Table 1

	L/U	M1	M2	M3
L/U	0	2	4	10
M1	12	0	2	8
M2	10	12	0	6
M3	4	6	8	0

Transfer time matrix for given layout

Table 2

Job	M1	M2	M3
1	8	16	12
2	20	10	18
3	12	8	15

Data for job set

VI. GENETIC ALGORITHM

As the GA work on coding of parameters, the feasible job sequences (the parameters of the considered problems) are coded in two different ways and separately experimented for the same problem.

- Pheno style coding
- Binary coding

In this work, Pheno style coding is considered.

For Pheno style coding GA parameters are as follows:

Initial Population size = 4

Reproduction: Tournament selection

Crossover probability= 0.6

Mutation probability = 0.01

Termination criteria = 100 number of generations or a satisfactory value for objectives, whichever occurs first.

Genetic Operations

A. Reproduction
The tournament selection method is used for reproduction. Tournament selection is one of many methods of selection in genetic algorithms. Tournament selection involves running several "tournaments" among a few individuals chosen at random from the population. The winner of each tournament (the one with the best fitness) is selected for crossover. Selection pressure is easily adjusted by changing the tournament size. If the tournament size is larger, weak individuals have a smaller chance to be selected. Reproduction procedure as follows:
Selection method: tournament selection. (Assume the parameters for comparison as 0.75)
Step 1: select two samples from the population.
Step2: evaluate the population.
Step3: generate random no. in the range (0 to 1)
Step4: if the random number is <= 0.75, select the best one else, select the inferior one.

B. Crossover
The strings in the mating pool formed after reproductions are used in the crossover operation. Single-point crossover is used in this work. With a Pheno-type coding scheme, two strings are selected at random and crossed at a random site. Since the mating pool contains strings at random, we pick pairs of strings from the top of the list. When two strings are chosen for crossover, first a coin is flipped with a probability Pc = 0.6 check whether or not a crossover is desired. If the outcome of the coin flipping is true, the crossover is performed, otherwise the strings are directly placed in the intermediate population for subsequent genetic operation. Flipping a coin with a probability 0.6 is simulated using the Monte Carlo method. The next step is to find a cross site at random. Once crossover point is selected, till this point the permutation is copied from the first parent, then the second parent is scanned and if the number is not yet in the offspring it is added.

(1 2 | 3 4 5 6)	+	(4 5 | 3 6 2 1)
Parent1		Parent2
= (1 2 4 5 3 6)	&	(4 5 1 2 3 6)
Child1		Child2

C. Mutation
The classic example of a mutation operator involves a probability that an arbitrary bit in a genetic sequence will be changed from its original state. A common method of implementing the mutation operator involves generating a random variable for each bit in a sequence. This random variable tells whether or not a particular bit will be modified. The purpose of mutation in GAs is to allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution. This reasoning also explains the fact that most GA systems avoid only taking the fittest of the population in generating the next but rather a random (or semi-random) selection with a weighting toward those that are fitter. In this problem mutation probability is 0.01 (i.e.) 8 bits will be mutated. First generate random number 0 to 1, with 0.01 accuracy. If random number is <= 0.01, perform mutation. The next step is to find a cross site at random, the two sites are selected by generating two random numbers between the numbers of jobs. For example, if the random numbers generated are 3 and 6, then the corresponding job numbers in these positions are exchanged.

$$\begin{array}{ccc} 1\ 2\ 3\ 4\ 5\ 6 & & 1\ 2\ 6\ 4\ 5\ 3 \\ \text{Parent} & \longrightarrow & \text{Child} \end{array}$$

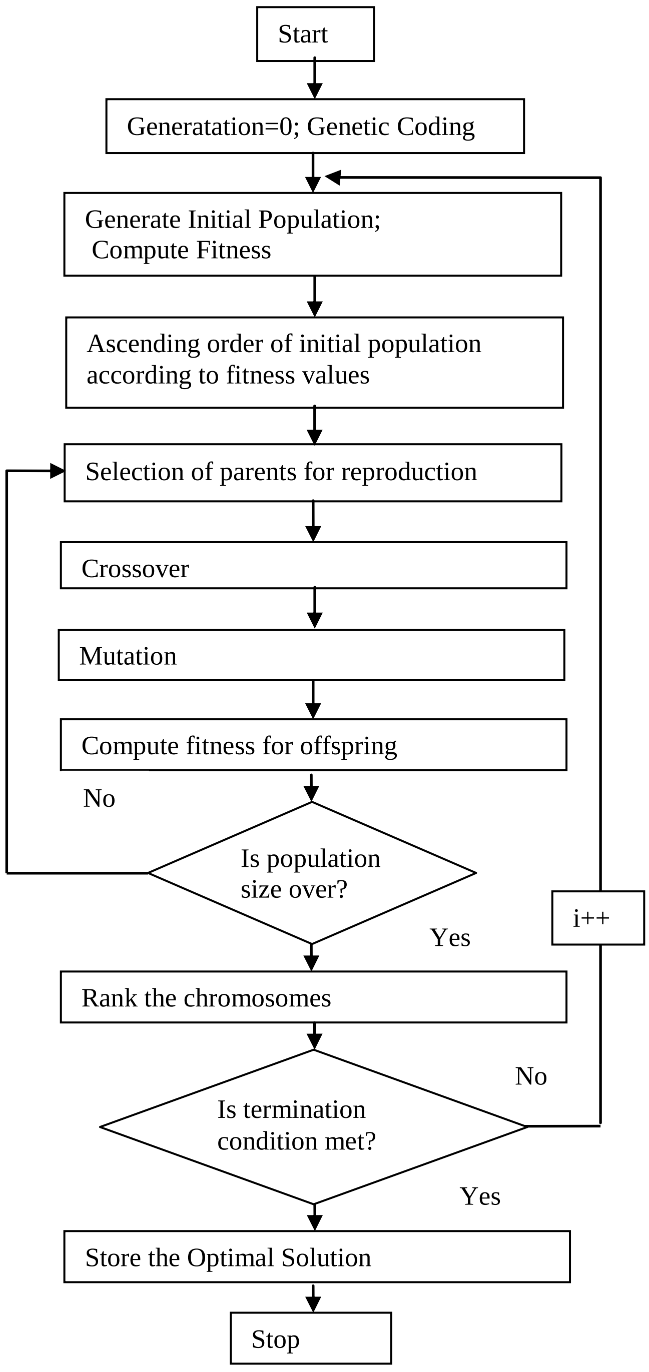
Start
Generatation=0; Genetic Coding
Generate Initial Population; Compute Fitness
Ascending order of initial population according to fitness values
Selection of parents for reproduction
Crossover
Mutation
Compute fitness for offspring
No
Is population size over?
Yes
i++
Rank the chromosomes
No
Is termination condition met?
Yes
Store the Optimal Solution
Stop

Fig 2: Flowchart for representing the task scheduling of AGV using GA

VII. IMPLEMENTATION OF GENETIC ALGORITHM

For scheduling the FMS and calculating the Makespan and Mean Flow Time continuous numbers are given for each of the operations as shown in Table III. These numbers are used to generate the initial random sequences, while obeying the precedence relation, the operation of a particular job must be in increasing order.

Table 3

Job No.	Job1		Job2		Job3	
Machine	M1	M3	M1	M2	M3	M2
Operation No.	1	2	3	4	5	6

Operation numbering system

In GA random sequence of operation is generated which serves as the initial population. Let the initial population of four sequence a,b,c,d is given as input:

Table 4

a	1	2	3	4	5	6
b	3	4	2	5	6	1
c	6	5	4	3	2	1
d	4	3	6	2	1	5

Set of initial population

Calculation of M_S: Makespan
Calculating M_S of chromosome b for illustration:
3 4 2 5 6 1

3rd Job
Time to transfer the job from L/U to M/c 1 + Processing Time (Job 2 on Machine 1) = 2 + 20 = 22

4th Job
Time to transfer the job from M/c 1 to M/c 2 + Processing Time (J2) = 2 + 10 = 12

2nd Job
Time to transfer the job from M/c 2 to M/c 3 + Processing Time (J1) = 6 + 12 = 18

5th Job
Time to transfer the job from M/c 3 to M/c 3 + Processing Time =
0 + 15 = 15

6th Job
Time to transfer the job from M/c 3 to M/c 2 + Processing Time =
8 + 8 = 16

1st Job
Time to transfer the job from M/c 2 to M/c 1 + Processing Time =
12+ 8 = 20

M_S = Make span = 22 + 12 + 18 + 15 + 16 + 20 = 103

A. Some C++ Codes used:

- For Random Number generation

```
//random number generator rn1
{
srand(time(0));
rn1=(rand()% MAX)+1;
cout<<"  rn1 = "<<rn1;
system("pause");
}
```

VIII. RESULTS AND DISCUSSIONS

A Non Traditional optimization approach using Genetic Algorithm is applied to the given layout and set of input data. The Codes for implementation of the algorithm is developed in C++ language. The stopping criterion for the running of loop is either 100 generations or meeting of the convergence criterion whichever occurs first. After meeting the earlier mentioned GA parameters the codes are executed to give the best routing sequence for the given FMS layout.

Table 5

S.No	No. Iterations	M_S	Best Chromosome
1	20	130	b
2	40	108	c
3	60	95	a
4	80	92	c
5	100	91	c

Set of initial population

The best chromosome after certain number of iterations is shown in Table V. In Fig. 2 graph is plotted between number of iterations and Make span M_S and a continues improvement in the value of Objective function is found which is seen to converge after 80 iterations thus a global minimum is found at a M_S value of 91.

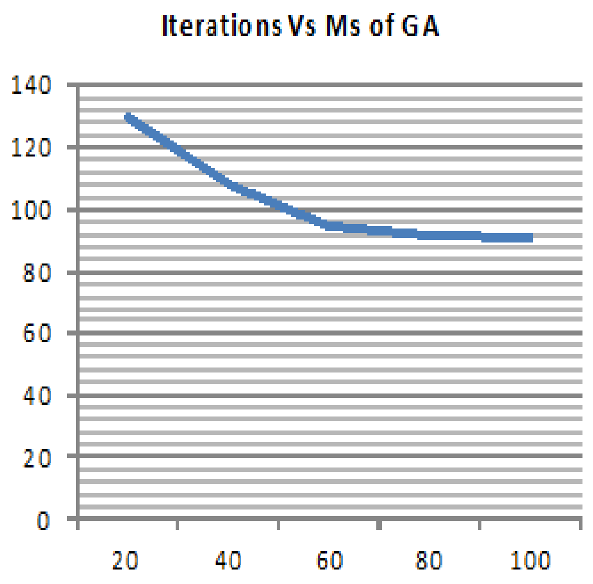

Fig 3: FMS scheduling result observation using GA

Table 6

Method	Optimum Sequence	M_S
GA	324165	91

Set of optimum sequence by GA

Pheno style GA coding is carried out which gives an optimum sequence of 324165 the result is shown in Table VI.

IX. CONCLUSION

Optimization procedure has been developed in this work which is based on genetic algorithm and is implemented successfully for solving the scheduling optimization problem of FMS. Codes are written in C++ language. Results are obtained for the jobs and 3 machines FMS system. With less computational effort it is possible to obtain the solution for such a large number of jobs and machines .This work leads to the conclusion that the procedures developed in this work can be suitably modified to any kind of FMS with a large number of components and machines subject to multi objective functions. Future work will include availability and handling times of loading/unloading stations, robots and AGVs.

X. ACKNOWLEDGEMENT

The transfer time matrix of input data and other relevant inferences for FMS are drawn out from MTAB FMS System procured under TEQIP-II, installed at Madan Mohan Malviya University of Technology, Gorakhpur. I would like to express my heartfelt gratitude to Mr. Jagdeesh of MTAB Engineers Pvt. Ltd. and Mr. S.K. Choudhary Lab assistant at Advance Machining Lab MMMUT, for helping me whenever needed.

REFERENCES

[1] N. Jawahar, P. Aravindan, “A genetic algorithm-based scheduler for setup-constrained FMC”, Elsevier Science B.V. Computers in industry 35 (1988) 291-310.

[2] Akhilesh Kumar, Prakash, M.K. Tiwari, “Solving machine-loading problem of a flexible manufacturing system with constraint-based genetic algorithm” Sciencedirect, European Journal of Operational Research 175 (2006) 043–1069.

[3] S. G. Ponnambalam, Low Seng Kiat, “Solving Machine Loading Problem in Flexible Manufacturing Systems Using Particle Swarm Optimization”, World Academy of Science, Engineering and Technology Vol:2 2008-03-25.

[4] K. V. Subbaiah, M. Nageswara Rao, “Scheduling of AGVs and machines in FMS with makespan criteria using sheep flock heredity algorithm”, International Journal of Physical Sciences Vol. 4(2), pp. 139-148, March, 2009.

[5] P. Udhaykumar, S. Kumanan, “Task Scheduling of AGV In FMS using Non-Traditional Optimisation Techniques”, Int J Simul Model 9 (2010) 1, 28-39, ISSN 1726-4529.

[6] Nidhish Mathew Nidhiry, Dr. R. Saravanan, “Evaluation of Genetic Algorithm Approach for Scheduling Optimization of Flexible Manufacturing Systems”, IJERA, Vol. 2, Issue 4, July-August 2012, pp.437-446.

[7] C. Anandaraman, ArunVikram, “Evolutionary approaches for scheduling a flexible manufacturing system with automated guided vehicles and robots”, International Journal of Industrial Engineering Computations 3 (2012) 627–648.

[8] Nidhish Mathew Nidhiry, Dr. Saravanan, “Fms Scheduling Optimization Using Modified NSGA-II”, Proceedings of 11th IRAJ International Conference, 2014, Chennai, India. ISBN: 978-93-82702-53-5.