Scheduling Optimization of FMS Using NSGA-II

Nidhish Mathew Nidhiry[1a], R. Saravanan[2b]

[1]Karapagam University, [2] JCT College of engineering and technology

[a]nidhishnidhiry@gmail.com
[b]saradarani@hotmail.com

Abstract: The Flexible Manufacturing Systems (FMS) belong to that class of productive systems whose main characteristic is the simultaneous execution of several processes and the sharing of a finite set of resources. Now days, FMS must attend to the demand of the market for personalized products. Consequently, the life cycle of the product tends to be shorter and a greater variety of products must be produced simultaneously. FMS considered in this work has 32 CNC Machine tools for processing 40 varieties of products. Since minimizing machine idle time and minimizing total penalty cost are contradictory objectives the problem has a multi objective nature. In this work, we have developed a multi-objective optimization procedure based on NSGA-II and software has been developed using .net programming for setting the optimum product sequence. A Global - optimal front was then obtained using the software after 3000 generations.
Keywords: Flexible manufacturing system, Multi - objective optimization, NSGA II, Scheduling Optimization, genetic algorithms

1 INTRODUCTION

FMS operational decisions consist of pre-release and post release decisions. FMS planning problems also known as pre-release decisions, take into account the pre-arrangement of parts and tools before the start of operation. FMS scheduling problems, which come under the category of post release decisions, deal with the sequencing and routing of the parts when the system is in operation. The machine loading problem in a FMS is specified so 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 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 outcome in close proximity to the real assumption of the FMS environment.

Flexible manufacturing systems (FMS) have been developed to combine the flexibility of job shops and the productivity of flow lines. Such systems consist of three sub-systems: a processing system including a number of CNC machines, an automated material-handling system to link these machines, and a computer control system controlling the operation of the whole FMS.While the first two sub-systems provide the potential to achieve high flexibility and high productivity, the computer control system determines how much of the potential can be realized. FMSs have been classified into different types according to their job flow patterns, size or type of production they use. From the point of view of scheduling and control, four types of FMS are defined: single flexible machines (SFMs); flexible manufacturing cells (FMCs); multi-machine flexible manufacturing systems (MMFMSs); and multi-cell flexible manufacturing systems (MCFMSs);

1.1 Earlier research

During the last three decades much research has been done in this area. Many heuristic algorithms have been developed to generate optimum schedule and part-releasing policies. Most of these algorithms include enumerative procedures, mathematical programming and approximation techniques, i.e., linear programming, integer programming, goal programming, dynamic programming, transportation and network analysis, branch and bound, Lagrangian

ISSN: 2222-7059(print), ISSN: 2222-7067(online), http://www.aspbs.com/aiem/
DOI: 10.7508/AIEM-V3-N1-63-72

relaxation, priority-rule-based heuristics, local search algorithms (ITS, threshold algorithm, Tabu search, SA), evolutionary algorithm (GA), etc. Of these techniques, few are specific to particular objectives, and few are specific to particular problem instances with respect to computational time needed.

Z.X guo and W.K wong [13] presented a comprehensive review of genetic algorithm based optimization model for scheduling flexible assembly lines. In this paper a scheduling problem in the flexible assembly line is investigated and a bi-level genetic algorithm is developed to solve the scheduling problem. Tiwari and Vidyarthi [9] proposed a genetic algorithm based heuristic to solve the machine loading problem of a random type FMS. The proposed GA based heuristic determines the part type sequence and the operation machine allocation that guarantee the optimal solution to the problem. Another scheduling paper [1], consider only 6 machines and 6 jobs. R Kumar, M K Tiwari and R Shankar [7], considered Ant Colony Optimization approach in FMS scheduling. Rossi and Dini [18] proposed an ACO based software system for solving scheduling problem in FMS considering routing flexibility, sequence-dependent set up, and transportation time. But ACO algorithm performs better in cases such as traveling sales problem, the vehicle rooting problem etc. Dong-Sheng Xu et al.[19], proposed An Improved Ant Colony Optimization (IACO) algorithm to the Flexible Job Shop Scheduling Problem (FJSSP). IACO algorithm provides an effective integration between Ant Colony Optimization (ACO) model and knowledge model. In the IACO algorithm, the knowledge model adopts some available knowledge for the optimization of ACO, and then employs the existing knowledge to guide the current heuristic searching.

In the last few years most research concerning the AGV scheduling has been focused on developing scheduling algorithms for a single objective such as the minimizing of setup cost loading and unloading time. Toker A, Kondakci S and Erkip N [10] proposed an approximation algorithm for the ‘n’ job ‘m’ machine resource constraint job shop problem. Hoitomt et al. [4] explored the use of the Lagrangian relaxation technique to schedule job shops characterised by multiple non-identical machine types, generic procedure constraints and simple routing considerations. W He and Kusiak [2] addressed three different industrial scheduling problems, with heuristic algorithms for each problem.

Lee and Dicesare [6] used Petri nets to model the scheduling problems in FMS. Similarly, Kim et al. [15] presented a new scheduling method for manufacturing system based on the timed Petri net model and a reactive fast graph search algoritham.Lee and Lee [16] adopted heuristic search for scheduling flexible manufacturing using lower bound reachability matrix which is based on T-timed Petri net. Liu J et al [17] proposed policies for deadlock prevention in FMS by defining a subclass of Petri nets called resource-shared net with buffers (RSNB).

Shnits and Sinreich [8] present the development of a multi-criteria control methodology for FMSs. The control methodology is based on a two-tier decision making mechanism. The first tier is designed to select a dominant decision criterion and a relevant scheduling rule set using a rule-based algorithm. In the second tier, using a look-ahead multi-pass simulation, a scheduling rule that best advances the selected criterion is determined. Yu and Greene [12] use a simulation study to examine the effects of machine selection rules and scheduling rules for a flexible multi-stage pull system. J. Jerald , P. Asokan , G. Prabaharan and R. Saravanan [5] proposed a combined objective scheduling optimization solution for FMS. Saravanan M. and Noorul haq [13] modified same problem in scatter-search approach of flexible manufacturing systems, but this work is only for 43 parts and few generations. Shanhikant Burnwal and Sankha Deb [20] use a cuckoo search based approach for the same problem.

Many authors have been trying to emphasize the utility and advantages of genetic algorithm, simulated annealing and other heuristics. In this vein, it has been proposed to use a new evolutionary computational approach called Non-dominated Sorting Genetic Algorithm-II (NSGA-II) for Multi-Objective Optimization NSGA-II of a specific manufacturing environment limiting ourselves to only two objectives. The procedure is applied to relatively large-size problems of up to 80 part varieties passing through 16 different CNC machine centers, and the results are found to be closer to the global optimum sequence.

1.2 Problem descriptions

The problem environment, assumption and aim of the present work are as follows:

1. The FMS considered in this work has a configuration as shown in Fig. 1. There are eight flexible machining cells (FMCs), each with two to six computer numerical machines (CNCs), an independent and a self-sufficient tool magazine, one automatic tool changer (ATC) and one automatic pallet changer (APC). Each cell is supported by one to

three dedicated robots for intra-cell movement of materials between operations. There is a loading station from which parts are released in batches for manufacturing in the FMS. There is an unloading station where the finished parts are collected and conveyed to the finished storage. There is one automatic storage and retrieval system (AS/RS) to store the work in progress. The eight FMCs are connected by two identical automated guided vehicles (AGVs). These AGVs perform the inter cell movements between the FMCs, the movement of finished product from any of the FMCs to the unloading station and the movement of semi-finished products between the AS/RS and the FMCs.

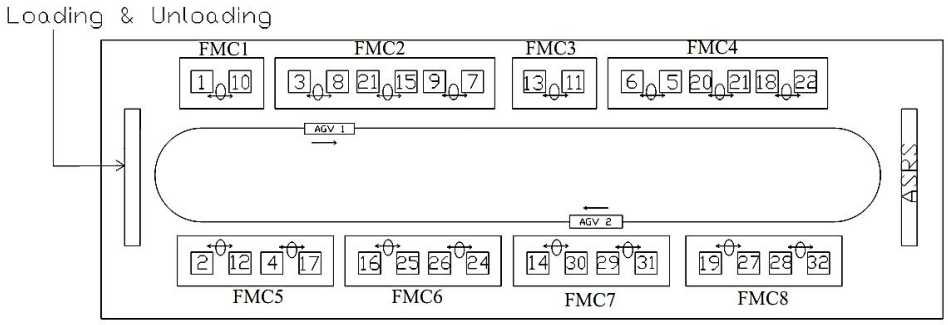

Figure 1.FMS structure

2. The assumptions made in this work are as follows:

- There are 40 varieties of products for a particular combination of tools in the tool magazines using 32 Machines in 5 FMCs.
- The type/variety has a particular processing sequence batch size, deadline and penalty cost for not meeting the deadline.
- Each processing step has a processing time with a specific machine.
- There is no constraint on the availability of pallets, fixtures, AGVs, robots, automated storage and retrieval system, cutting tools, and part programs as and when they are needed at the required places.
- A random product-mix generated as shown in the Table 1 reflects the current market demand.

3. The objective of the schedule: -
1. Minimizing the machine ideal time (TDi).

2. Minimizing the total penalty cost (TPi).
Where
TDi = Total Machine Idle Time.
$TDi = \Sigma_j MI_j$ (j= machine number)
$MI_j = TI-\Sigma_i PT_{ji}$ (i= job number)
TI = Total elapsed time.
PT_{ji}= Processing time of i^{th} job on the j^{th} machine.
TPi = Total penalty cost
$TPi= \Sigma_i PT_i - DD_i \times P_i \times BS_i$
PT_i =Processing time of i^{th} job.
DD_i= Due date for i^{th} job.

2PROPOSED METHODOLOGY

2.1 NSGA II algorithm

The Methodology used to find the optimal solution to this problem is NSGA. It is based on a ranking procedure, consisting in extracting the non-dominated solutions for a population and giving them a rank of 1.These solutions are removed from this population; the next group of non-dominated solution has a rank of 2 and so on. The algorithm has a current population that is used to create an auxiliary one (the offspring population); after that, both populations are combined to obtain the new current population. The procedure is as follows; the two populations are sorted according to their rank, and the best solutions are chosen to create the new population. In the cause of having to select some individuals with same rank, a density estimation based on measuring the crowding distance (see figure 2) to the surrounding individuals belonging to the same rank is used to get the most promising solutions. Typically, both the current and auxiliary population has equal size. The procedure of NSGA – II is shown in figure 3 and flow chart shown in figure 4. The basic steps involved in NSGA II algorithm are:

1) Generation t = 0;
(2) Generate a uniformly distributed parent population of size P;
(3) Evaluate the individuals and sort the population based on the non-domination;
(4) Assign each solution a rank equal to its non-domination level (minimization of fitness is assumed);
(5) Use the usual Fino style tournament selection;

(6) Use the Simulated single point Crossover operator and mutation to create an Offspring population of size P;
(7) Combine the offspring and parent population to form extended population of size 2P;
(8) Sort the extended population based on non-domination;
(9) Fill new population of size P with the individuals from the sorting fronts starting from the best;
(10) Invoke the crowding-distance method to ensure diversity if a front can only partially fill the next generation. The crowding-distance method maintains diversity in the population and prevents convergence in one direction;
(11) Update the number of generations, t = t + 1;
(12) Repeat the steps (3) to (11) until a stopping criterion is met.

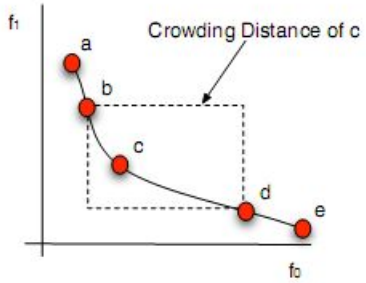

Figure 2. Measuring the crowding distance of a non-dominated point.

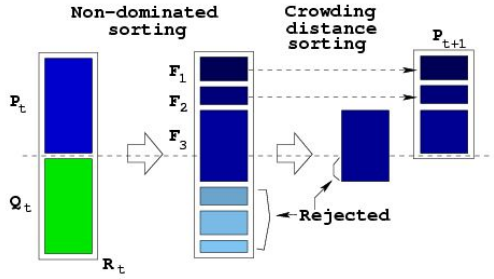

Figure 3. The procedure of NSGA – II

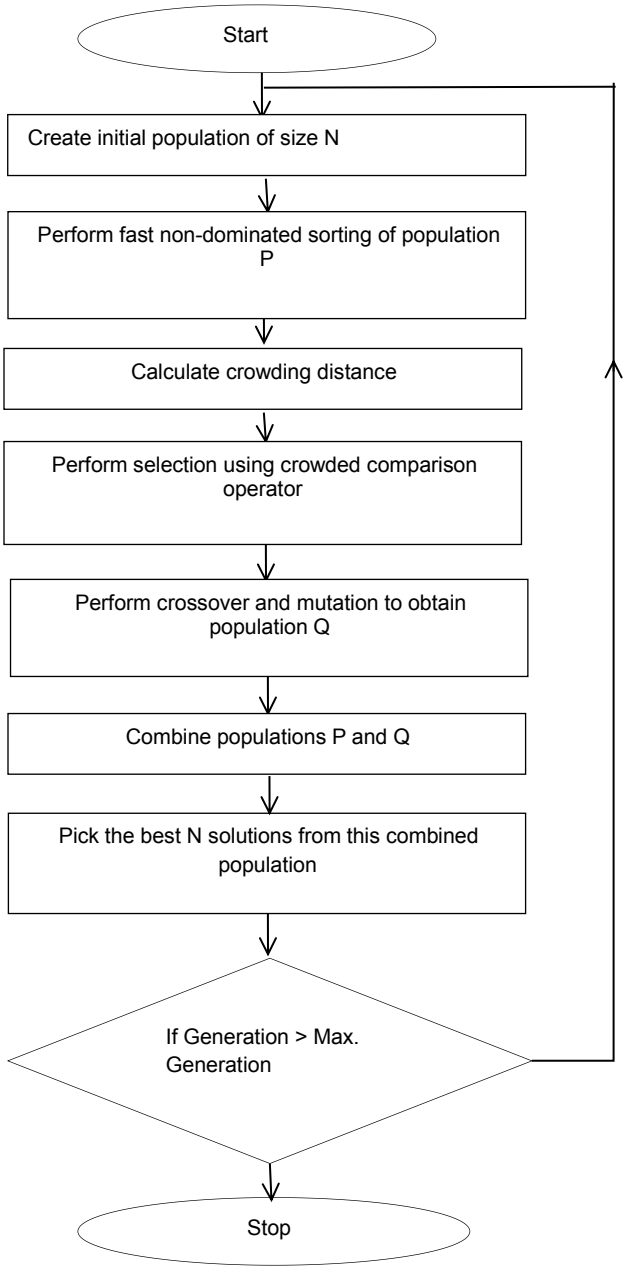

Figure 4. Flow chart of NSGA II algorithm

Optimization procedure

The objective of the schedule: -
1. Minimizing the machine ideal time (TDi).
2. Minimizing the total penalty cost (TPi).

Table 1. Machining sequence, time, deadline, batch size and penalty details.

Part no.	Processing sequence – M/c No. & process time in min.	Deadline (days)	Batch size (Nos.)	Penalty cost (Rs./unit/day)
1	{1,4},{8,1},{9,2},{11,2}	17	300	2.00
2	{10,1},{3,1},{15,3},{7,2},{6,4},{21,2}	17	500	1.00
3	{5,1},{17,3},{18,4}	14	800	1.00
4	{22,4},{30,2}	16	800	3.00
5	{2,5},{4,3},{25,4}	11	250	2.00
6	{31,5},{19,1}	16	700	1.00
7	{2,5},{26,3},{31,5}	26	550	3.00
8	{17,4},{29,5},{28,1}	25	950	2.00
9	{14,1},{25,5},{24,1},{31,1}	1	200	0.00
10	{3,2},{11,4},{22,2}	14	250	3.00
11	{28,4},{22,4}	1	250	1.00
12	{26,2},{18,4},{31,1},{30,1}	19	1000	3.00
13	{26,1},{27,5},{18,4}	20	800	4.00
14	{24,3},{1,3},{6,2},{15,2}	22	1000	4.00
16	{30,3}	17	450	3.00
15	{15,4},{18,3}	15	500	5.00
17	{13,1},{26,4},{24,1}	16	550	3.00
18	{19,2},{6,3}	24	250	5.00
19	{14,1},{25,5},{6,2},{28,2},{25,5}	5	650	1.00
20	{7,2},{13,4}	11	250	5.00
21	{24,5},{5,5},{26,2},{18,2},{5,5}	11	950	3.00
22	{12,5}	24	200	5.00
23	{14,2},{5,1},{6,5},{18,4}	14	150	4.00
24	{8,4},{19,4},{12,5},{13,4}	7	200	5.00
25	{17,3},{10,2}	24	350	1.00
26	{30,2}	24	450	0.00
27	{18,5},{21,5},{12,4}	21	400	1.00
28	{12,1},{18,1},{19,2}	4	950	5.00
29	{4,1},{15,5}	7	700	1.00
30	{12,3},{13,5}	14	1000	1.00
31	{18,2},{12,2}	2	800	2.00
32	{12,3},{16,4},{19,3}	13	800	1.00
33	{15,4},{16,5},{5,3}	17	600	4.00
34	{13,2},{16,2}	12	300	4.00
35	{3,4},{14,1}	9	700	2.00
36	{13,2}	20	700	2.00
37	{11,5},{21,2},{16,3},{8,3},{9,2},{16,4}	22	250	4.00
38	{21,4},{29,3},{19,2},{16,5}	8	50	1.00
39	{6,5},{10,5}	9	500	1.00
40	{21,2},{16,4},{19,4}	7	150	5.00

GA coding scheme

As the GA work on coding of parameters, the feasible job sequences (the parameters of the considered

(1) Fino style coding
(2) Binary coding
In this work, Fino style coding is considered.

Fino style coding:
In this coding each sequence is coded as 40 sets of two-digit numbers ranging from 01 to 40.

Ex:
12,18,27,32,11,03,31,15,26,01,19,08,14,16,37,17,25, 39,06,23,13,34,10,30,40,22,35,09,38,05,20,02,36,29, 28,07,21,04,24,33.

GA parameters
Population size = 100
Reproduction: Tournament selection (Target value – 0.75)
Crossover probability= 0.6
Mutation probability = 0.01
Termination criteria = 3000 number of generations or a satisfactory value for objectives, whichever occurs first.

Consider the complexity of one iteration algorithm. The basic operations and their worst-case complexities are as follows:
1.Non dominated sorting is O (MN^2)
2.Crowding – distance assignment is O (M (N) log (N))
Overall complexity is O (MN^2), which is governed by non-dominated sorting. Where M is the number of objectives and N is the population size (third N is for the maximum number of fronts).

3 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 otherwise, 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 Fino-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 to check whether or not a crossover is desirable. If the outcome of the coin flipping is positive, 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 7 8 9) + (**4 5 3 6 8** 9 7 2 1) = (**1 2 3 4 5** 6 8 9 7) (**4 5 3 6 8** 1 2 7 9)
Parent1 Parent2 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. 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.

1 2 **3** 4 5 **6** 7 8 9 → 1 2 **6** 4 5 **3** 7 8 9

Parent Child

4 RESULT AND DISCUSSIONS

The optimization procedures developed in this work are based on the non-dominated sorting Genetic Algorithm (NSGA-II). The FMS configuration considered in this work is taken from existing literature [7]. In literature [5], procedure is developed for 43 jobs using combined objective optimization method. A Comparison between the proposed NSGA II and other algorithms namely SPT, PSO, CS (found in literature) and NSGA II after 40 generations has been presented in table2 and fig.3. Table 2 shows the results obtained by the proposed NSGA – II and it performs better in terms of objective functions and computational effort, i.e.50% less than CS. The figure 3 shows the comparison of both the approaches in the study. But in our work we have taken the scheduling problem with 40 parts and multi objective optimization approach is implemented. Results are obtained for 43 jobs problem using combined objective method and NSGA-II, and the performance is compared and analyzed. Table 4 shows the results obtained by both approaches and it is found that NSGA – II performs better.

Table 2 Comparison between various approaches

Algorithm	SPT[20]	PSO[20]	CS[21]	NSGA - II
Machine idle time	180100	315650	163800	109850
Penalty Cost	101930	298196	138025	16298
Sequence	20,23,38,1,9,26,22,10,34,18,36,11,25,5,16,2,40,4,41,31,7,24,28,17,6,29,35,37,15,39,42,27,33,3,43,19,13,12,32,30, 8,14,21	27,30,38,10,18,15,34,42,5,33,8,37,23,25,9,23,5,43,20,6,4,36,19,17,24,39,31,12,8,32,26,6,14,22,3,1,11,41,9,40,21,13,7	8,14,28,31,3,42,26, 33, 22,20,5,24, 12, 41,18,7,10,19,23,38,4,35,40,37,15,15,17,39,6,2,34,1,29,27,16,36,30,25,32,13,3,11,10,9.	5,30,34,28,16,24,25,10,11,27,36,2,18,1,4,29,20,13,37,17,3,9,41,12,15,6,22,7,42,38,19,23,43,21,32,14,33,8,26,35,40,31,39.

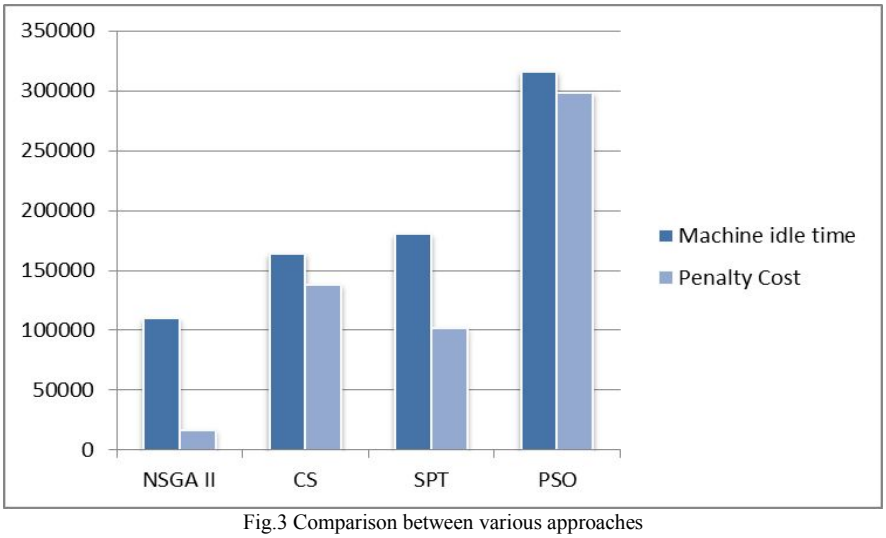

Fig.3 Comparison between various approaches

Table 4 Result after 100 generations

Methodology	Machine idle time in minutes	Minimum total penalty cost in Rs.
Combined objective function	206850	122555.9
NSGA-II	195085	121185.3

4.1 Results obtained for 80 jobs scheduling problem by NSGA-II

Global Pareto optimal front is obtained after executing 3000 generations and the details are shown in table 5. Results are shown in figure 4 to 10. The table 6 shows the same problem considering average 10 minutes for AGV movement. Table 6 shows the results of similar problems with 16 machines producing 80 jobs and 20 machines producing 50 jobs. The software is executed on an Intel Core 2 Duo based PC with 4 GB RAM using .net software. It took 20 minutes to complete the computation.

Table 5 Result after 3000 generations (80 Jobs scheduling problem)

Methodology	Trial No.	Machine idle time in minutes	Minimum total penalty cost in Rs.
NSGA-II	1	181200	55636.45833
	2	181250	55271.875
	3	182650	54398.61111
	4	185950	53190.625
	5	182300	54763.19444
	6	184750	53699.30556

Table 6 Result of different problems after 3000 generations

Problems	Trial No.	Machine idle time in minutes	Minimum total penalty cost in Rs.
16 machines & 80 jobs	1	169835	98968
	2	167335	99770
	3	166415	100289
20 machines & 50 jobs	1	147850	55070
	2	135100	59132
	3	135900	56899

5 CONCLUSIONS

Optimization procedure has been developed in this work based on the Multi-objective Non-Dominated genetic algorithm (NSGA-II). This method is implemented successfully for solving the scheduling optimization problem of FMS. Software has been written in .net Language. FMS schedule is obtained for 40 jobs and 32 machines. The result obtained by NSGA-II is analyzed for two objectives, minimizing total penalty cost and minimizing total machine idle time. For the given set of objectives, the results obtained by the NSGA-II-based algorithm have been compared with a number of existing algorithms including CS, SPT, and PSO (found in literature), and the machine idle time and the total penalty cost by NSGA-II-based algorithm is found to be better than or as good as the best results obtained by the aforementioned methods. After 3000 generation the best solution is obtained. The problem of computational effort of FMS scheduling increases in proportion to the number of components. In the case of 40 components 8.159152832478977343 4561126959612e+47combinations are possible. Due to the very high computational effort exhaustive search is not possible. Similarly random search also requires a lot of computational effort. By implementing genetic algorithm for 3000 generations, only 1.5 lakh computations are needed for getting the optimal solution. The research work leads to the conclusion that NSGA II is superior in terms of computational effort and pareto optimal front. The procedure developed in this work can be suitably

modified to any kind of FMS with a large number of components and machines. Future work will include the availability and handling time of load unloading stations, robots and AGV's.

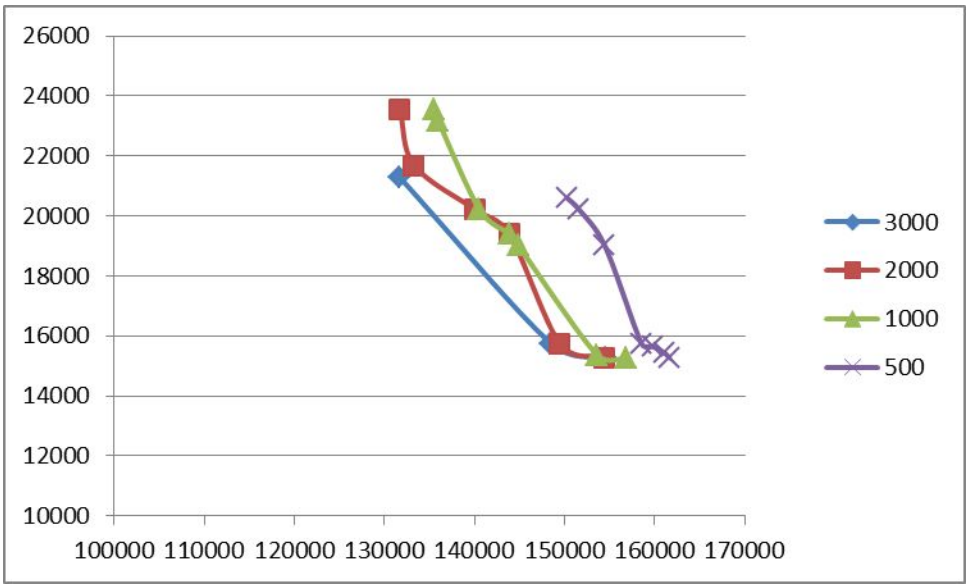

Figure 11.Progression of Pareto-optimal fronts.

References

[1] AVS Sreedhar kumar and Dr.V. Ve- eranna 2010, Optimization of FMS scheduling using non-traditional techniques. nternational Journal of Engineering Science and Technology. pp:7289-7296.

[2] W He, Kusiak A., 1992, Scheduling of manufacturing systems. Int. J Comput Ind., pp:163–175.

[3] Hoitomt DJ, Luh PB, Pattipati KR 1993, A practical approach to job-shop scheduling problems. IEEE Trans Robot Automat, pp:1–13.

[4] Deb, K., 2001. Multi-Objective Optim- ization using Evolutionary Algorithms, John Wiley & Sons, Ltd.

[5] J. Jerald, P. Asokan , G. Prabaharan, R. Saravanan. 2004, Scheduling optimisati- on of flexible manufacturing systems using particle swarm optimisation algorithm, Int. Journ. Of Advanced Manufacturing Technology, pp:964 – 971.

[6] Lee DY, Dicesare F, 1994, Scheduling of flexible manufacturing systems: using Petri nets and heuristic search. IEEE Trans Robot Automat, pp:23–132.

[7] R Kumar, M K Tiwari and R Shankar 2003, Scheduling of flexible manufacturing systems: an ant colony optimization approach. Proc. Instn Mech. Engrs. Vol. 217 Part B: J. Engineering Manufacture.

[8] Shnits B, Sinreich D. 2006. Controlling flexible manufacturing systems based on a dynamic selection of the appropriate operational criteria and scheduling policy. Int J Flex Manuf Syst. pp:1–27.

[9] Tiwari M.K, Vidiyarthi N.K. 2000, Solving Machine loading problem in flexible manufacturing system using genetic algorithm based heuristic approach. International journal of production research, pp:3357-87.

[10] Toker A, Kondakci S, Erkip N., 1994, Job shop Scheduling under a nonrenewable resource constraint. J Oper Res Soc., pp:942–947.

[11] Yu MC, Greene TJ., 2006. A simulation analysis of controlling rules for flexible pull systems. Int J Manuf Res. pp:314–331.

[12] Z. X. Guo , W. K. Wong, S. Y. S. Leung, J. T. FanS, F. Chan., 2008, A genetic-algorithm-based optimization model for scheduling flexible assembly lines. Int J Adv Manuf Technol . pp:156–168.

[13] Saravanan M. and Noorul haq A., 2008, 'Evaluation of Scatter Search Approach for Scheduling Optimization of Flexible Manufacturing Systems'. International journal of Advanced manufacturing Technology, pp. 978-986.

[14] Kalyanmoy, Amrit Pratap, Sameer Agarwal, and T. Meyarivan., 2002, A Fast and Elitist Multiobjective Genetic Algorithm:NSGA-II, ieee transactions on evolutionary computation.

[15] Kim, Suzuki, Narikiyo., 2007, FMS scheduling based on time petro net model and reactive graph search. Appl Math Model. pp:955 – 970.

[16] Lee J, Lee SJ., 2010, Heuristic search for scheduling flexible manufacturing systems using lower bound reachability matrix. Comput Ind Eng. pp:799-806.

[17] Liu J et al., 2009. A live subclass of Petri nets and their application in modeling flexible manufacturing system. Int J Adv Manuf Technol. pp:66-74.

[18] Rossi A, Dini G., 2007. Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimization method. Robot Comput Integrated Manuf. pp:503–516.

[19] Dong-Sheng Xu et al., 2009, An Improved Ant Colony Optimization for Flexible Job Shop Scheduling Problems. International Joint Conference on Computational Sciences and Optimization, pp.517-519.

[20] Sankar S et al., 2003. A multiobjective genetic algorithm for scheduling a flexible manufacturing system. Int J Adv Manuf Technol. pp:229–236.

[21] Shashikant Burnwal , Sankha Deb, 2013. Scheduling optimization of flexible manufacturing system using cuckoo search-based approach. Int J Adv Manuf Technol. pp:951–959.