# Combined Objective Optimization Of FMS Scheduling With Non-Traditional Optimization Techniques

Mr. A.V.S. Sreedhar Kumar [1], Dr. V. Veeranna [2], Dr. B. Durgaprasad [3] & Dr. B. Dattatraya Sarma [4]

**Abstract**

The Flexible Manufacturing System (FMS) is a complex discrete event dynamic system (DEDS) and complete utilization of the available resources in a system is extremely important to optimize its productivity [01]. Scheduling of manufacturing systems refers to the determination of the sequence in which jobs are to be processed over the production stages, followed by the determination of the start time and finish time of processing jobs. The importance of FMS scheduling is to increase the utilization of the resources and to reduce the idle time and reduction of in-process inventory. In this work we have demonstrated the effectiveness of evolutionary approaches like Genetic Algorithm and Differential Evolution in scheduling of FMS systems. To demonstrate the validity of the proposed approach we have considered a 43 job 16 machines FMS system. The results of the proposed approach are compared with conventional scheduling rules [03]. The proposed method is coded using Matlab version 7.1

*Keywords:* Genetic Algorithm, Differential Evolution, FMS, scheduling, priority rules

*1.Professor, Mechanical engg.dept.,Narayana Engg.college.,Nellore, A.P-524004*
*2.Professor & Dean, Brundavan institute of Technology & science,Kurnool,A.P,518002*
*3.Professor, Dept. of. Mechanical Engg.,JNTUA college of Engineering, Anantapur, A.P-515002*
4. *Principal,S.V.College of Engg. Nellore-A.P*

## 1. Introduction

The increasing demand for low cost, low-to-medium volume production of modular goods with many different variations creates the need for production systems that are flexible and that allow for small product delivery times. This leads to production systems working on small batches, having low setup times and mainly characterized by many degrees of freedom in the decision making process.This type of system is known as flexible manufacturing systems (FMS). They are highly automated systems with built in redundancies that should provide the desired flexibility. Scheduling in an FMS environment is more complex and difficult than in a conventional manufacturing environment. To achieve high performance for an FMS, a good scheduling system should make a right decision at a right time according to system conditions

## 2. Proposed methodology

### 2.1 Genetic Algorithm:

Genetic Algorithms are computerized search and optimization algorithms based on mechanics of natural genetics and natural selection [02]. They operate on the principle of "The survival of the fittest", where weak individuals die before reproducing, while stronger ones live longer and bear many off spring and breed children, who often inherit the qualities that enabled their parents to survive. The reproduced children are in most cases stronger than their parents.

( Eiben, A. E. et al ,1994).

Each cycle produces a new generation of possible solutions for a given problem. At the first stage an initial population is created as a starting point for the search. Each element of the Population is called a (string) chromosome. In the next stage, a performance (or fitness) of each individual is evaluated with respect to the constraints imposed by the problem. Based on each individual's fitness, a selection mechanism chooses "mates" for the genetic manipulation process. The selection process is ultimately responsible for selecting of the best fitted individuals. The manipulation process uses genetic operators to produce a new population of individuals (offspring) by manipulation "Genetic information" referred to as Genes possessed by members (parents) of current population [05]. Some of the terms and its significance in genetic algorithm.

*Population:* ***Set of solutions.***

*Individual:* ***Solution to a problem.***

*Fitness:* ***Quality of a solution***

*Chromosome:* ***Encoding for a Solution.***

*Gene:* ***Part of the encoding of a solution.***

*Reproduction:* ***Crossover***

### 2.1.1 Outline of the basic genetic algorithm

1. **[Start]** Generate random population of *n* chromosomes (suitable solutions for the problem).

2. **[Fitness]** Evaluate the fitness *f(x)* of each chromosome *x* in the population.

3. **[New population]** Create a new population by repeating the following steps until the new population is complete.

3.1. **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the greater the chances of being selected).

3.2. **[Crossover]** With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring are an exact copy of parents.

3.3. **[Mutation]** With a mutation probability, mutate new offspring at each locus (position in chromosome).

3.4. **[Accepting]** Place new offspring in a new population.

4. **[Replace]** Use newly generated population for a further run of algorithm.

5. **[Test]** If the end condition is satisfied, **stop**, and return the best solution in current population.

6. **[Loop]** Go to step 2.

### 2.1.2 Genetic Algorithm Operators

(a).Reproduction

It is usually the first operator applied on a population. It selects good string in a population and forms a mating pool. In this, the above average string is picked from the current population and their multiple copies are inserted in the mating pool in a probabilistic manner. The commonly used reproduction operator is the proportionate reproduction operator where a string is selected for the mating pool with a probability proportional to its fitness. Since the population size is usually kept fixed, the sum of the probability of each string selected must be one.

(b).Crossover

In the crossover operator, new strings are created by exchanging information among string of the mating pool. Two strings are picked from the mating pool at random and some portions of the strings are exchanged between the strings. The two strings participating in the crossover operation are known as parent strings and the resulting strings are known as children strings. It is intuitive from this construction that good sub-strings from parent strings can be combined to form better child strings, if an appropriate site is chosen. Since the knowledge of an appropriate site is usually not known beforehand, a random site is often chosen. With a random site, children strings produced may or may not have a combination of good sub-strings from parent strings, depending on whether or not the crossing site falls in the appropriate place. Because of good strings are created by crossover, there will be more copies of them in the next mating pool generated by the reproduction operator. But if good strings are not created by crossover, they will not survive too long, because reproduction selects against those strings in subsequent generations [05].

(c) Mutation

The mutation operator changes 1 to 0 and vice versa with a small mutation probability. The bit-wise mutation is performed bit by bit by flipping a coin with a probability if at any bit the outcome is true then the bit is altered, otherwise the bit is kept unchanged. The need for mutation is to create a point in the neighborhood of the current point, thereof achieving a local search around the current solution. The mutation is also used to maintain diversity in the population.

**2.1.3 The GA parameters used in optimization are as mentioned below**

Population Size: 100

Scaling Function: Rank

Selection Function: Uniform

Elite Count: 2

Cross over fraction: 0.8

Mutation Function: Adaptive Feasible

Cross Over Function: Single Point.

Generations: 1000

Time limit: -Inf-

**2.2 Differential Evolution:**

DE is a simple evolutionary algorithm that encodes solutions as vectors and uses operations such as vector addition, scalar multiplication and exchange of components (crossover) to construct new solutions from the existing ones. When a new solution, also called candidate, is constructed, it is compared to its parent. If the candidate is better than its parent, it replaces the parent in the population. Otherwise, the candidate is discarded. As a steady-state algorithm, DE implicitly incorporates elitism, i.e. no solution can be deleted from the population unless a better solution is found. While being a very successful optimization method, DE's greatest limitation originates in its encoding [11,12]. As no vector representation of solution exists for combinatorial problems, DE can only be applied in numerical optimization

**Differential Evolution for Multi-objective Optimization**

1. Evaluate the initial population P of random individuals.

2. While stopping criterion not met, do:

2.1. For each individual Pi(i=1,...,popSize) from P repeat:

(a) Create candidate C from parent Pi

(b) Calculate the objectives of the candidate.

(c) If the candidate dominates the parent, the candidate replaces the parent.

(d) If the parent dominates the candidate, the candidate is discarded. Otherwise, the candidate is added in the population.

2.2. If the population has more than popSize individuals, apply environmental selection to get the best pop Size individuals.

2.3. Randomly enumerate the individuals in P.

3. Return nondominated individuals from P

**2.2.1 The parameter settings for DE is as follows**

Population Size: 100;

Maximum Iterations: 1000

Mutation Factor: 0.5

Crossover Rate: 0.9

**2.3 Priority rules:**

These rules refer only to some particular aspects of the scheduling problem, that is, to the ones of interest for the present study [3,8,9]. These rules are briefly presented here, for more precise descriptions the work of Young-On&Yao, (1994) and Joshi & Smith, (1994) can be reffered.

• Sequencing: that is deciding the ordering of orders to be inserted into the system.

• Routing: that is deciding where to send a job for an operation in case of multiple choices.

• Priority: setting for a job in a machine buffer: that is deciding which will be the next job to be served by a machine.

**2.3.1 Some sequencing rules are:**

- EDD (Earliest Due Date) : the first order that enters the system is the one with the earliest due date
- HP(Highest Penalty): the first order that enters the system is the one with the highest penalty
- LPT( Longest Processing Time) : the first order

that enters the system is the one with the longest processing time

- SPT( Shortest Processing Time) : the first order that enters the system is the one with the shortest processing time.
- LBS (largest batch size): the first order that enters the system is the one with the largest batch size
- SBS(shortest batch size): the first order that enters the system is the one with the shortest batch size

**2.3.2 Assumptions**

(1) All machines are available at time 0;
(2) All jobs are released at time 0;
(3) Each machine can process only one operation at a time;
(4) Each operation can be processed without interruption on one of a set of available machines;
(5) Recirculation occurs when a job could visit a machine more than once;
(6) The order of operations for each job is predefined and cannot be modified.

**2.4 Matlab Programming Approach**

Matlab software is used for programming the proposed approach. The use of matlab enables us to solve complex scheduling problems involving different job types and multiple machines. The program is coded in such a way that the user can have flexibility in terms of varying the number of machines and number of jobs. The input to the program is given in the form of data stored in Excel sheet table-1&2. The data includes machine timings for each operations, operation sequence, batch size, and penalty and due date. The user if required to change, can alter the job size or number of machines by deleting the appropriate values in the Excel sheet. Upon running the program it prompts the user to enter file name in which the data is stored.

**3. Operating Environment and Problem Definition FMS Problem description**

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 four flexible machining cells (FMCs), with two to six computer numerical machines (CNCs), each is provided with independent tool magazines, part program controller, automatic tool changer (ATC) and buffer storage, part-carrying conveyors (input and output), a robot, and an automated storage and retrieval system (AS&RS). All the above are linked by means of host computer[04]. The four FMCs are connected by automated guided vehicles (AGV). This AGV perform the intercell 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. There is a dedicated robot for loading and unloading AGV.

2. The assumptions made in this work are as follows:
There are 40 to 50 varieties of products for a particular combination of tools in the tool magazines. Each 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.

3. The objective of the schedule is the combination of minimizing the machine ideal time and minimising the total penaltycost.

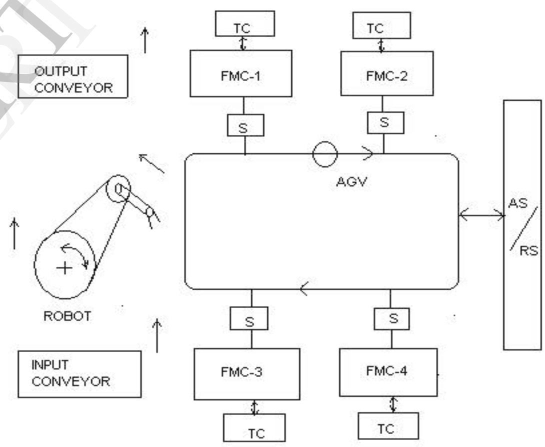


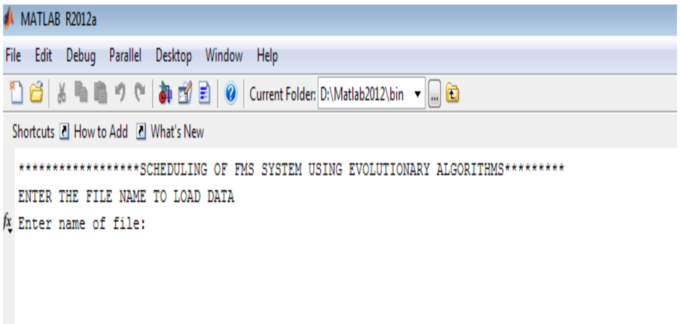

*Fig-1 :Configuration of the FMS , FMC(Flexible manufacturing cell),TC(Tool changer),AGV(Automated guided vehicle),AS/RS(Automated storage and retrieval system),S-shuttles*

| FMC number | Machine number |
|---|---|
| FMC-1 | 4,7,10 |
| FMC-2 | 3,14,15 |
| FMC-3 | 1,11,12,13 |
| FMC-4 | 2,5,6,8,9,16 |

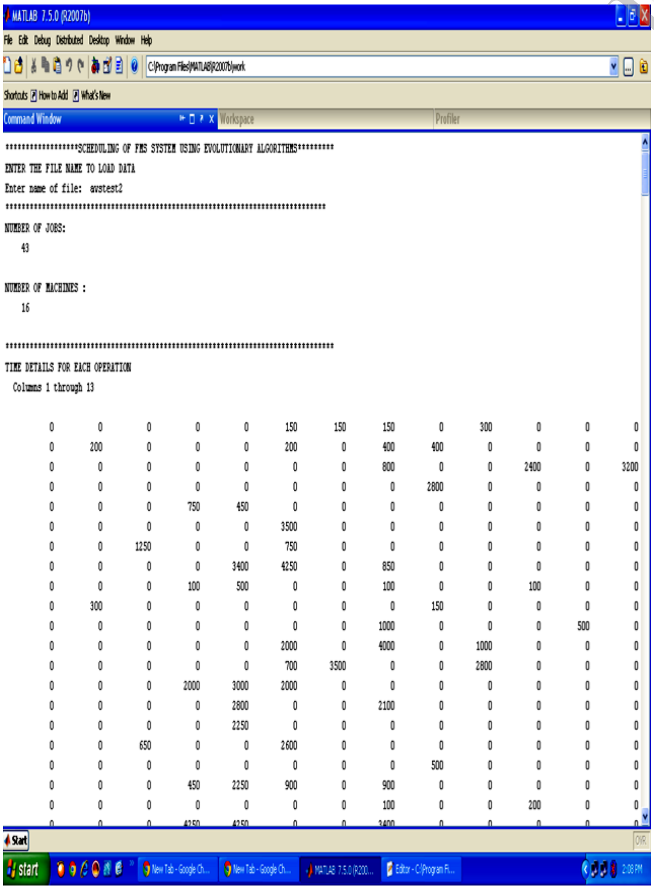
***Table 1.*** *Machining sequence, P.T- process time (in min),D.D (due date in days),B.S( batch size in No's) and P.C ( penalty cost in Rs/units/day)(43jobs-16machines)*

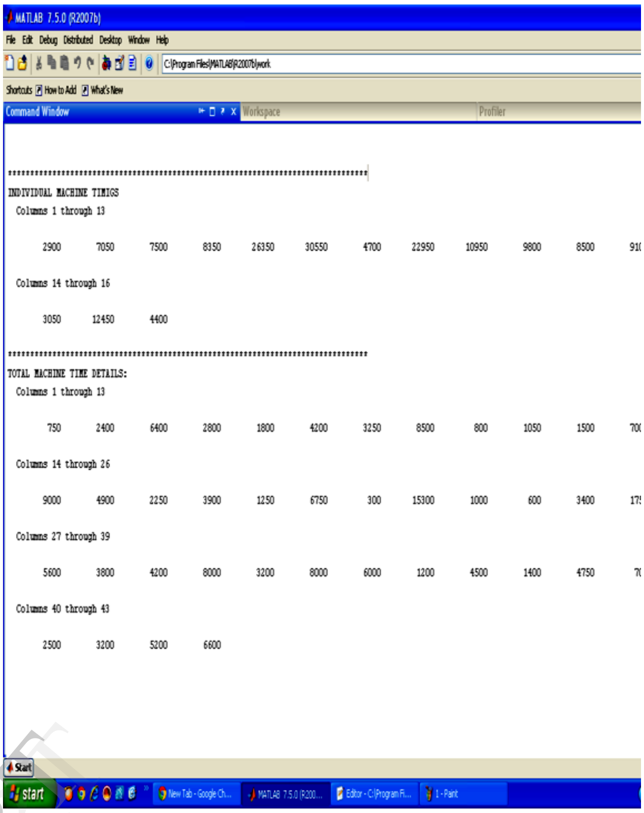
| | M1 | M2 | M3 | M4 | M5 | M6 | M7 | M8 | M9 | M10 | M11 | M12 | M13 | M14 | M15 | M16 | B.S | P.C | D.D |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 150 | 1 | 17 |
| P2 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 4 | 0 | 2 | 200 | 1 | 17 |
| P3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 3 | 0 | 4 | 0 | 0 | 0 | 800 | 1 | 14 |
| P4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 700 | 2 | 26 |
| P5 | 0 | 0 | 0 | 5 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 150 | 1 | 11 |
| P6 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 700 | 1 | 16 |
| P7 | 0 | 0 | 5 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 250 | 2 | 26 |
| P8 | 0 | 0 | 0 | 0 | 4 | 5 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 850 | 2 | 26 |
| P9 | 0 | 0 | 0 | 1 | 5 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 100 | 0 | 1 |
| P10 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 150 | 2 | 20 |
| P11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 250 | 1 | 1 |
| P12 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 4 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1000 | 3 | 19 |
| P13 | 0 | 0 | 0 | 0 | 0 | 1 | 5 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 700 | 4 | 25 |
| P14 | 0 | 0 | 0 | 2 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 1000 | 4 | 22 |
| P15 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 700 | 5 | 15 |
| P16 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 750 | 3 | 27 |
| P17 | 0 | 0 | 1 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 650 | 4 | 20 |
| P18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 250 | 5 | 24 |
| P19 | 0 | 0 | 0 | 1 | 5 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 450 | 1 | 5 |
| P20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 50 | 5 | 11 |
| P21 | 0 | 0 | 0 | 5 | 5 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 850 | 3 | 16 |
| P22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 200 | 5 | 24 |
| P23 | 0 | 0 | 0 | 2 | 1 | 5 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 50 | 4 | 14 |
| P24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 4 | 5 | 4 | 0 | 0 | 0 | 200 | 5 | 7 |
| P25 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 350 | 1 | 24 |
| P26 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 450 | 0 | 27 |
| P27 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 5 | 4 | 0 | 0 | 0 | 0 | 400 | 1 | 22 |
| P28 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 950 | 5 | 3 |
| P29 | 0 | 0 | 0 | 1 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 700 | 1 | 7 |
| P30 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 5 | 0 | 0 | 0 | 0 | 1000 | 1 | 18 |
| P31 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 800 | 2 | 2 |
| P32 | 0 | 3 | 0 | 0 | 0 | 4 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 800 | 1 | 15 |
| P33 | 0 | 0 | 0 | 0 | 4 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 500 | 4 | 27 |
| P34 | 0 | 0 | 2 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 300 | 4 | 12 |
| P35 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 900 | 2 | 9 |
| P36 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 700 | 2 | 20 |
| P37 | 5 | 2 | 0 | 0 | 0 | 3 | 0 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 250 | 4 | 22 |
| P38 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 50 | 1 | 8 |
| P39 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 500 | 1 | 9 |
| P40 | 0 | 2 | 0 | 0 | 0 | 4 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 250 | 5 | 7 |
| P41 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 800 | 4 | 22 |
| P42 | 0 | 5 | 0 | 0 | 0 | 4 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 400 | 2 | 19 |
| P43 | 3 | 0 | 0 | 0 | 2 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 550 | 3 | 15 |

Snapshot of Matlab program requesting the user to enter the file name




Upon loading the file name the program returns the schedule sequence as well as the value of Combined Objective Function , Penalty , Idleness for that particular Sequence






## 4.Optimisation procedure

### 4.1 Objective function

In this work, the combined objective function (COF) of minimizing the machine idle time and minimizing the total penalty cost is considered [6,10]. However, for computational convenience, the machine setup times are assumed to be same for all the machines. Feasible schedule is evaluated using the COF for minimizing the total penalty cost and maximizing machine utilization. Therefore the objective becomes,

Minimize COF = $(W_{CS})$ X (TPC/MPP) + $(W_{MU})$ X (TMD / TE ),

where, $W_{CS}$= weight factor for customer satisfaction and $W_{MU}$ = weight factor for machine utilization.

TPC=total penalty cost incurred, and

$$TPC = \sum_{i} (CTi - DDi)\ X\ UPCi\ X\ BSi$$

where, i = job number, CT*i* = completion time of job i, DD*i* = due date for job I, UPC*i* = unit penalty cost for job I,

$BS_i$ = batch size of job I.

MPP = maximum permissible penalty

TMD = total machine down time, and

$$TMD = \sum_{j} \mathrm{MDj}$$

TE= total elapsed time,

PTji= processing time of i[th] job with j[th] machine, where

j= machine number, and

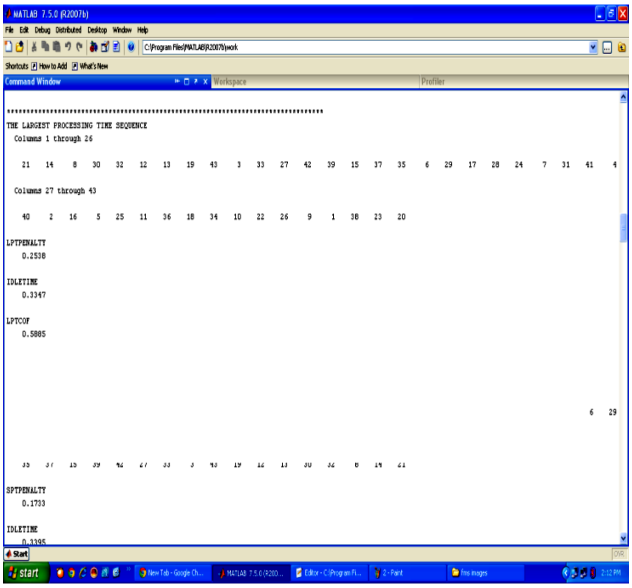
$$\mathrm{MDj} = TE - \sum_{j} \mathrm{PTji}$$

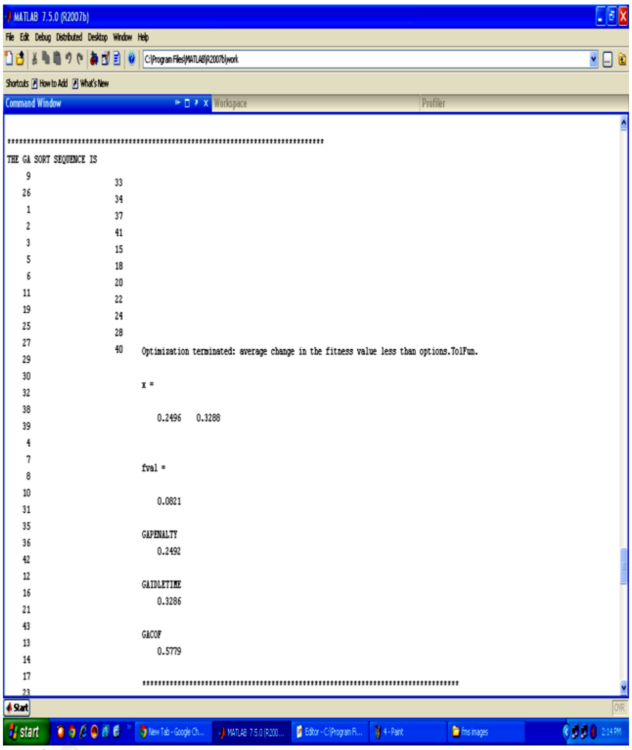
In the computation the weight factors $W_{CS}$ and $W_{MU}$ are assumed to be equal and hence, $W_{CS} = 0.5$ and $W_{MU} = 0.5$. However, different ratios can be applied to them according to the demand of business situation

## 5. Results and comparison

The optimisation procedures developed in this work are based on the various non-traditional approaches that have been implemented using MATLAB 7.1. Different optimal schedules are obtained for the FMS using the above approaches, and the performances are compared and analysed. Among the Eight approaches used in this work, the schedule obtained by the D.E algorithm gives the optimum COF value, i.e., minimum total penalty cost and minimum machine idleness, as shown in the tables 2&3

*Snapshot of program execution after the data is loaded*



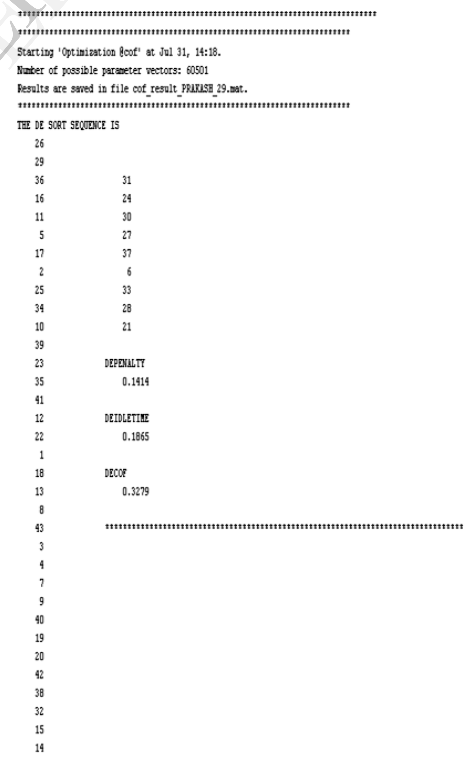



Table- 2 Summary of results by different techniques, 43 jobs- 16 machines problem

| S.no | Scheduling Technique | Sequence | Penalty | Idleness | COF |
|---|---|---|---|---|---|
| 1 | LPT (Largest processing time) | 21 14 8 30 32 12 13 19 43 3 33 27 42 39 15 37 35 6 29 17 28 24 7 31 47 4 40 2 16 5 25 11 36 18 34 10 22 26 9 1 38 23 20 | 0.2538 | 0.3347 | 0.5885 |
| 2 | SPT( Shortest processing time) | 20 23 38 1 9 26 22 10 34 18 36 11 25 5 16 2 40 4 31 41 7 24 28 17 6 29 35 37 15 39 42 27 33 3 43 19 12 13 30 32 8 14 21 | 0.1733 | 0.2830 | 0.4563 |
| 3 | LBQ(Largest batch quantity) | 12 14 30 28 35 8 21 3 31 32 41 16 4 6 13 15 29 36 17 43 33 39 19 26 27 42 25 34 7 11 18 37 40 2 22 24 1 5 10 9 20 23 38 | 0.2287 | 0.3016 | 0.5302 |
| 4 | SBQ(Shortest batch quantity) | 20 23 38 9 1 5 10 2 22 24 7 11 18 37 40 34 25 27 42 19 26 33 39 43 17 4 6 13 15 29 36 16 3 31 32 41 8 21 35 28 12 14 30 | 0.2243 | 0.2957 | 0.5200 |
| 5 | HP(Highest penalty) | 15 18 20 22 24 28 40 13 14 17 23 33 34 37 41 12 16 21 43 4 7 8 10 31 35 36 42 1 2 3 5 6 11 19 25 27 29 30 32 38 39 9 26 | 0.3658 | 0.1206 | 0.4864 |
| 6 | EDD(Earliest due date) | 9 11 31 28 19 24 29 40 38 35 39 5 20 34 3 23 15 32 43 6 21 1 2 30 12 42 10 17 36 14 27 37 41 18 22 25 13 4 7 8 16 26 33 | 0.3385 | 0.2232 | 0.5616 |
| 7 | GA(Genetic algorithm) | 9 26 1 2 3 5 6 11 19 25 27 29 30 32 38 39 4 7 8 10 31 35 36 42 12 16 21 43 13 14 17 23 33 34 37 41 15 18 20 22 24 28 40 | 0.2492 | 0.3286 | 0.5779 |
| 8 | DE(Differential Evolution) | 15 4 26 25 11 33 8 1 40 34 27 2 13 7 29 14 39 10 9 20 36 41 18 31 23 43 28 37 6 3 24 38 12 42 32 5 35 30 21 22 16 19 17 | 0.1515 | 0.1998 | 0.3513 |



Table-3 Summary of results by different techniques, 43 jobs- 10 machines problem

| S.no | Scheduling Technique | Sequence | Penalty | Idleness | COF |
|---|---|---|---|---|---|
| 1 | LPT (Largest processing time) | 21 8 32 12 13 14 39 43 15 42 19 33 29 28 37 35 6 17 31 4 40 41 16 7 27 25 36 2 5 34 11 26 3 24 1 9 23 18 10 38 20 22 30 | 0.1873 | 0.3252 | 0.5125 |
| 2 | SPT( Shortest processing time) | 22 30 20 10 38 18 23 9 1 3 24 26 11 2 5 34 36 25 7 27 16 41 40 4 31 17 6 35 37 28 29 19 33 42 15 43 39 12 13 14 32 8 21 | 0.1040 | 0.2234 | 0.3274 |
| 3 | LBQ(Largest batch quantity) | 12 14 30 28 35 8 21 3 31 32 41 16 4 6 13 15 29 36 17 43 33 39 19 26 27 42 25 34 7 11 18 37 40 2 22 24 1 5 10 9 20 23 38 | 0.1663 | 0.2887 | 0.4549 |
| 4 | SBQ(Shortest batch quantity) | 20 23 38 9 1 5 10 2 22 24 7 11 18 37 40 34 25 27 42 19 26 33 39 43 17 4 6 13 15 29 36 16 3 31 32 41 8 21 35 28 12 14 30 | 0.1459 | 0.2533 | 0.3993 |
| 5 | HP(Highest penalty) | 15 18 20 22 24 28 40 13 14 17 23 33 34 37 41 12 16 21 43 4 7 8 10 31 35 36 42 1 2 3 5 6 11 19 25 27 29 30 32 38 39 9 26 | 0.2600 | 0.1128 | 0.3729 |
| 6 | EDD(Earliest due date) | 9 11 31 28 19 24 29 40 38 35 39 5 20 34 3 23 15 32 43 6 21 1 2 30 12 42 10 17 36 14 27 37 41 18 22 25 13 4 7 8 16 26 33 | 0.2320 | 0.2014 | 0.4335 |
| 7 | GA(Genetic algorithm) | 9 26 1 2 3 5 6 11 19 25 27 29 30 32 38 39 4 7 8 10 31 35 36 42 12 16 21 43 13 14 17 23 33 34 37 41 15 18 20 22 24 28 40 | 0.1749 | 0.3036 | 0.4785 |
| 8 | DE(Differential Evolution) | 15 4 26 25 11 33 8 1 40 34 27 2 13 7 29 14 39 10 9 20 36 41 18 31 23 43 28 37 6 3 24 38 12 42 32 5 35 30 21 22 16 19 17 | 0.1148 | 0.1993 | 0.3140 |

## 7. Conclusion

In this work, Optimisation procedures have been developed based on the two non-traditional approaches, i.e., genetic algorithm and D.E. These are implemented successfully for solving the optimisation problem of FMS scheduling. Code has been written in MATLAB 7.1. Results are obtained for two types of problems: 43 jobs- 10 machines and 43 jobs- 16 machines. Results obtained by the different approaches are compared and the performances are analysed for the combined objective function of minimising total penalty cost and minimizing total machine idleness. D.E algorithm is found to be superior and gives the minimum combined objective function.

## 8. References

1. Giffler B, Thomson GL (1960) Algorithms for solving production scheduling problems. Int J Oper pg:487–503

2. Guohui Zhang a, Liang Gao b, Yang Shi b"An effective genetic algorithm for the flexible job-shop scheduling problem"
*Expert Systems with Applications 38 (2011) 3563–3573*

3. Durgesh Sharma,Suresh Garg,Chitra Sharma".Effect of Scheduling rules on performance of Semi Automated Flexible Manufacturing System" Global Journal of Enterprise Information System January-June 2012,Volume-4 Issue-I

4. J. Jerald • P. Asokan • G. Prabaharan • R. Saravanan "Scheduling optimisation of flexible manufacturing systems using particle swarm optimisation algorithm" Int J Adv Manuf Technol (2005) 25: 964–971

5. N. Jawahar t, P. Aravindan* and S. G. Ponnambalam*A Genetic Algorithm for Scheduling Flexible Manufacturing Systems" Int J Adv Manuf Technol (1998) Springer

6. B.B. Choudhury,B.B. Biswal,R.N.Mahapatra" Appropriate Evolutionary Algorithm for Scheduling in FMS" 2009 IEEE

7. Guohui Zhang ,Liang Gao,Yang Shi."A Genetic Algorithm and Tabu Search for Multi Objective Flexible Job Shop Scheduling Problem", 2010 International Conference on Computing,Control and Industrial Engineering, 251-254

8.Some Studies on optimum design of flexible manufacturing system layout ,P Hd. thesis Dr.V.Veeranna

9.Dr.V.Veeranna,Dr.B.Dattatreya sarma,Dr.G.Chakraverti "Optimization of FMS Layout by Heuristic Procedure with Scheduling as a Constraint"Industrial Engg.Journal , Jan 2006,25-28

10. B.B. Choudhury,B.B. Biswal,D.Mishra,"Task assignment and scheduling in a constrained manufacturing system using GA"International J Agile System & Management

11 .Dervi_s Karabo_GA ,Sel_cuk ¨Okdem ,"A Simple and Global Optimization Algorithm forEngineering Problems: Differential Evolution Algorithm" Turk J Elec Engin, VOL.12, NO.1 2004,

12. M.V. Satish Kumar, C.S.P Rao, G.Rangajanardhan," A Hybrid Differential Evolution Approach for Simultaneous Scheduling Problems in a Flexible Manufacturing Environment" The International Journal of Applied Management and Technology, Vol 6, Num 3