OPTIMIZATION METHODOLOGIES ADOPTED TO SOLVE FMS SCHEDULING

4.1 INTRODUCTION TO OPTIMIZATION

Decision making is a very crucial process in any operation planning and management. Any optimization problem needs to be defined explicitly for which an optimum solution has to be found. This is usually done by defining the decision variables, constraint functions and objective functions. Decision variables are the independent variables for which the optimum values are to be found based on which, the other variable can be designed. Constraint functions are used to define some limitations to the solution space. An objective function is defined in terms of decision variables. Most of the decisions are defined by either maximizing the function or minimizing it to the least possible value [1]. Figure 4.1 shows an outline of the procedure usually involved in an optimal design formulation process.

Once the decision variables, constraints and objective functions are defined, the problem has to be modeled in a proper form in order to be solved. The modeling is usually influenced by the available tools, algorithms, accuracy etc. Once the problem is modeled, it can be solved using different optimization approaches such as the mathematical modeling, heuristics, Metaheuristic approaches etc.

4.2 Priority Rules

Priority rules and heuristic search techniques are capable of finding a good solution in relatively short period of time [9]. Even though, they cannot guarantee to find an optimal solution, they can be used to choose the job with the highest priority, as defined by the rule when there is more than one job waiting to be processed by the same machine [14]. In this research work, the following priority rules are employed.

4.2.1 Largest Processing Time (LPT)

According to this rule, the priority is given to the job which has maximum processing time. The job which has the maximum processing time is executed first.
4.2.2 Shortest Processing Time (SPT)

According to this rule, the priority is given to the job which has the least processing time and these jobs are executed in the order of increasing the values of the processing time.

![Flow chart of the optimal design procedure](image)

4.2.3 Earliest Due Date (EDD)

This priority rule prioritizes the execution of the job in accordance with their delivery schedule. The job which has to be delivered at an earlier due date is given preference to the job over others which can be delivered at a later date.
4.2.4 Highest penalty (HP)

This priority rule first executes the job which has highest penalty at first and the other jobs are sequenced in the order of decreasing penalty values.

4.2.5 Largest batch size (LBS)

This rule fixes the priority by considering the size of the batch of each job and the job having the largest batch size is executed first.

4.2.6 Smallest batch size (SBS)

According to this rule, the preference is given to the job which has the least batch size and the other jobs are sequenced in the increasing order of the batch sizes.

4.3 METAHEURISTIC APPROACHES ADOPTED TO SOLVE FMS SCHEDULING

Metaheuristic algorithms are approximate, non deterministic and have checks and balances, to avoid getting trapped to confined areas of search space. The nature of the Metaheuristics search, orchestrates the interaction of local improvement procedures and higher level strategies. This results in the creation of a process that helps performing robust search of feasible region that makes it capable of escaping from local optima. Escaping from the local optima is one key feature of the Metaheuristics. The primary advantage of Metaheuristic is, that it has tendency to move quickly towards very good solutions. This presents a very efficient way of solving large and complicated problems. The primary disadvantage of Metaheuristics is that there is no guarantee that the best solution will be an optimal solution or even a near optimal solution [20].

The primary role of Metaheuristics is to deal with the problems that are too complex and large in size to be solved by heuristic and other approaches. Metaheuristic approaches apply the domain specific knowledge and use the search experience to guide the search for optimal solutions. Some of the prominently used Metaheuristic for FMS scheduling problems are Genetic Algorithm, Fuzzy logic, Ant colony algorithm, Particle swarm optimization algorithm, Tabu search, Simulated annealing etc. In recent times, algorithms inspired by biological phenomenon or
natural phenomenon are being widely used to optimize a given problem. Owing to the simplicity and fast computational time coupled with their ability to be applied for multiple problems, these algorithms have attracted the attention of the researchers worldwide. In this research work, three Metaheuristics approach such as DE, GE & BFOA are used to optimize the research problem.

![Flow Chart of Evolutionary Algorithm](image)

**Fig. 4.2: Flow Chart of Evolutionary Algorithm**

With the increasing use of Metaheuristics for solving optimization problems, biologically inspired evolutionary algorithms have become an indispensable tool in Metaheuristic based approach. The use of evolutionary algorithms for solving NP hard optimization problem is progressively gaining the grounds. These three evolutionary algorithms consist of three main processes. These processes include the generation of initial solution, evaluation of fitness of the solution and generation of new population. In the first process, the initial population is randomly generated according to a particular representation of solution. In the next process, each solution is assessed and evaluated for fitness value. In the third process, the new population is
perturbing solutions in the existing population. The basic flow of an Evolutionary algorithm is summarized in figure 4.2.

4.3.1 Genetic Algorithm

Even though the concept of GA was known much earlier than 1975, John Holland of the University of Michigan and Ann Arbor have introduced the concept of GA in a broad way to the research audience in the mid sixties [1]. The primary idea of GA is to mimic the process of natural selection and survival of the fittest. In GA, the solutions are represented in the form of chromosomes [solutions]. Each of these chromosomes is then evaluated for fitness and ranked on the basis of the fitness value. The process of mimicking natural selection of living organisms is accomplished through the application of GA operators such as the selection, crossover and mutation. These three operators define the means of how GA moves towards a particular solution. In the first process of selection, better chromosomes are selected to represent parents, which will produce new offspring. In order to simulate the survival of the fittest, the chromosomes with better fitness have high probability of selection when compared the chromosomes with poor fitness. These selection probabilities are typically defined using the relative values of fitness values represented in the form of ranks.

Typically, there are two types of selection methods widely proposed in the literature. One is Roulette-wheel selection and the other one is Tournament selection. The next operator is crossover which combines the parent chromosomes to produce new offspring. The crossover probabilities define as to how the crossover operator functions and many cross over methods are reported in the literature. In order to maintain the diversity of solution and to avoid stagnation, the third operator mutation is used. In the mutation method, the probability is selected and defined in such a manner as to maintain the diversity of the population by injecting new elements into the chromosomes. Apart from these three factors, the population size, the maximum number of iterations, elite count, minimum fitness value and some of the other parameters that have to be defined for GA. The basic flow of a GA algorithm is summarized in Figure 4.3.
4.3.1.1 Steps involved in the GA optimization process

Step 1: Generate initial population and initialization of Genetic algorithm parameters.

Step 2: Evolution of fitness for the initial population by computing the value of objective function.

Step 3: Ranking the initial population based on fitness.

Step 4: Checking for stopping criteria, if the stopping criteria is satisfied to provide optimized schedule, otherwise to go to the next step.

Step 5: Select better individuals based on ranking and fitness.

Step 6: Employ crossover operator to generate new population from two different parent chromosomes.

Step 7: Employ mutation operator to generate new chromosome from a selected individual chromosome.

Step 8: Checking for stopping criteria, if yes, provide the optimized final schedule, otherwise go to step 2.

4.3.1.2 GA parameters used in optimization

Population Size: 100

Scaling Function: Rank

Selection Function: Uniform

Elite Count: 2

Cross over fraction: 0.8

Mutation fraction: 0.2

Mutation Function: Adaptive Feasible

Cross Over Function: Single Point
4.3.2 Differential Evolution (DE)

Differential Evolution was proposed by Storn and Price [74] for global optimization over continuous search space. It has a simple theoretical framework which is simple and requires a relatively few control variables but performs well in convergence. In DE, a solution is given by a D-dimensional vector. It starts with a random generation at initial population of size $N$ of D-dimensional vectors. In DE, the values in the D-dimensional space are commonly represented as real numbers. The concept of solution representation is similar as applied in GA.
The key difference of DE from GA is in the mechanism for generating new solutions. DE generates a new solution by combining several solutions with the candidate solution. The population of solutions in DE, evolves through repeated cycles of three main DE operators; mutation, crossover and selection. However, the operators are not at all exactly the same as those with the same names in GA.

The primary process in DE is the generation of trial vector. Consider a candidate or target vector in a population of size N of D-dimensional vectors. The generation of a trial vector is done by the mutation and crossover operations and can be defined as follows.

- Creating a mutant vector by mutation of three randomly selected vectors.
- Creating a trial vector by the crossover of mutant vector and target vector.

Initially, a mutant vector is generated by combining three randomly selected vectors from the population of which excludes the target vector. This process of combining three randomly selected vectors to form the mutant vector V is defined as \( V = X_1 + F(X_2 - X_3) \) where \( X_1, X_2 \) and \( X_3 \) are three randomly selected vectors from the population and \( F \) is a multiplier which is the main parameter of the DE algorithm [21]. The above process which is used to form the mutant vector is referred to as mutation. Its function is different from the context in which it is used in GA.

The next step is to create the trial vector by performing crossover between the mutant vector and the target vector. There are typically two crossover methods in DE; binomial crossover and exponential crossover. Here, the crossover probability must be specified. A small crossover probability leads to a trial vector that is more similar to the target vector while the opposite favors the mutant vector. After the trial vector is formed for a given target vector, selection is done to keep only one of the two vectors. The vector with better fitness value is retained. In other words, the target vector will survive if the trial vector has poorer fitness. Otherwise, the trial vector replaces the target vector immediately and becomes eligible for selection in the formation of the next mutant vector [21]. The basic flow of a DE algorithm is summarized in figure 4.4.

In the figure 4.4, the first process is the generation of a population of new solutions called vectors. Each vector in the population is evaluated for fitness value.
Each vector takes turns as a candidate or target vector and for each target vector, a trial vector is formed. The selection process is to choose between the target vector and the trial vector. Since a new solution would be selected only if it has better fitness, the average fitness of the population would be equal or better from iteration to iteration. Any improvement in the solution is immediately available to be randomly selected to form a mutant vector for the next target vector. This is different from GA where an improvement would take effect only after all the solutions have completed the iteration.

### 4.3.2.1 Steps involved in the DE optimization process

Step 1: Generation of initial population and parameters for DE.

Step 2: Evaluating a fitness function for random initial population.

Step 3: Generating a target vector

Step 4: Generating a trail vector and evaluating the fitness of the trail vector.

Step 5: Selecting the better vector which has the least fitness function value between the trail vector and target vector.

Step 6: Updating the global best vector.

Step 7: Checking for stopping criteria, if yes, provide the optimized final schedule, otherwise go to step 3.

### 4.3.2.2 The parameter settings for DE are as follows:

Population Size: 100;

Maximum Iterations: 1000

Mutation Factor: 0.5

Crossover Rate: 0.9
In contrast with GA, where parent solutions are selected based on fitness, every solution in DE takes turns to be a target vector (one of the parents) and thus all vectors play a role as one of the parents with certainty. The second parent is the mutant vector which is formed from at least three different vectors. In other words, the trial vector is formed from at least four different vectors and would replace the target vector only if this new vector is better than the target vector; otherwise, it would be abandoned. This replacement takes place immediately without waiting for the whole population to complete the iteration. This improved vector would then immediately be available for random selection of vectors to form the next mutant vector.
4.3.3 Bacterial Foraging Optimization Algorithm (BFOA)

In 2002, Passino[13] was inspired by the social foraging behavior of Escherichia coli and proposed the Bacteria Foraging Optimization Algorithm (BFOA). Since its inception, BFOA has drawn the attention of researchers in different fields of knowledge, in terms of its biological motivation, elegant structure and its problem solving abilities. The advantages of BFOA include some inherent capabilities such as parallel distributed processing, insensitivity to the initial value and global optimization. Four stages are involved in foraging process. These stages are chemotaxis, swarming, reproduction, and elimination and dispersal. BFOA obtains optimum value through the chemotaxis of bacteria and realize quorum through assemble function between bacteria. The rule of evolution is satisfied by choosing the fittest through reproduction operation and the elimination-dispersal mechanism is used to avoid premature convergence [60].

The motion patterns that the bacteria will generate in the presence of chemical attractants and repellents are referred as chemotaxis [13]. This process is achieved by two different moving ways; run or tumble. A Bacterium switches between these two modes of operation in its entire lifetime. The bacterium sometimes tumbles after a tumble or tumbles after a run. This alternation between the two modes will move the bacterium and this enables it to search for nutrients. An interesting group behavior that has been observed for several motile species of bacteria including E.coli and S. typhimurium is a group of E. coli cells is placed in the center of a semisolid agar with single nutrient chemo-effectors. They move out from the center in a traveling ring of cells by moving up the nutrient gradient created by consumption of the nutrient by the group. In order to achieve this, they have cell-to-cell signaling via an attractant and a repellent [65].

In the case of bacteria a reproduction step takes place after the completion of all chemotactic steps. The process of elimination and dispersal also aid in the process of chemotaxis. From the point of evolution, elimination and dispersion guarantees diversity of the population. From the stand point of optimization, it helps to have the ability to achieve global optimization. The bacteria are usually eliminated with a probability value. The basic flow of a BFO algorithm is summarized in Figure 4.5.
4.3.3.1 Steps involved in the BFOA optimization process

Step 1: Initialization of BFOA parameters.

Step 2: Evaluation of fitness by computing value of the objective function.

Step 3: Initiating chemotaxis tumble / run.

Step 4: Checking for the end of chemotaxic, if true, going to step 5 otherwise going to step 2.

Step 5: Starting the reproduction.

Step 6: Checking for the end of reproduction as per initialization, if yes, going to step 7 or else going to step 2.

Step 7: Initiating elimination and dispersion.

Step 8: If end of elimination and dispersion, going to next step or else going to step 2.

Step 9: Providing the optimal schedule.

4.3.3.2 Parameters of BFOA used in this research work

The number of bacteria: 20

Number of chemotactic steps: 10

Limit length of a swim: 4

The number of reproduction steps: 4

The number of elimination-dispersal events: 2

The number of bacteria reproductions: 2

The probability that each bacterium will be eliminated / dispersed: 0.2
Fig. 4.5: Flowchart of the Bacterial Foraging Optimization Algorithm
4.4 Promodel Simulation software

Promodel, the windows based simulation and animation tool, is a powerful application for modeling, simulating and analyzing manufacturing systems of different types and sizes. It is particularly suitable for supply chain systems and other typical applications which include assembly lines, Job Shops, Transfer lines, JIT, KANBAN systems, Flexible Manufacturing systems etc. It provides the scope for using several built-in distribution functions, elements of production systems, which helps to focus and analyze resource utilization, productivity, production capacity and inventory levels [26].

Promodel helps the user to try different operating strategies and designs which can maximize the results. It is a typical discrete event simulation tool, which is highly suitable for modeling discrete part manufacturing systems. The tool is designed in such a fashion that it models event to occur at predefined points of time. The tool has the provision for controllable time resolution. Promodel processes the parts or entities using some processing logic on machines or work stations, which can be viewed as an arrangement of processing locations. These systems can also include parts, supporting resources that are used in processing and movement of the parts. The model development is completely graphical and object oriented [27].

4.4.1 Modeling elements

The modeling elements form necessary building blocks that represent the physical and the logical components of the system being modeled. These physical elements are either referred to graphically or by name. Figure 4.6 shows various elements of Promodel and figure 4.7 shows the FMS model building setup considered in the case problem 2.

4.4.1.1. Locations

Routing locations are predefined fixed points in the system. Different parts or entities are routed to these fixed points for storage for processing or making decisions about further routing. These locations can be either multi unit locations, comprising a group of similar machines performing operations parallel to, or a single unit location having a single machine. These locations can have capacity which is usually greater
than one. They can also have periodic down time, represented as a function of clock time. These locations can also have usage time, usage frequency, machine setup time etc., Input and output rules are assigned to these routing locations.

These rules are particularly useful in the multi capacity locations where the input rules define what entity has to be processed next, whereas the output rules are used for defining the ranks of the entities like FIFO, LIFO etc. A queue is used to mimic the behavior of the waiting line, which usually includes the movement of the entities through these lines. Two different types of conveyors like accumulating and non-accumulating having a particular speed and load spacing are typically used. These conveyors can be configured to provide conveyor networks.

![Fig. 4.6: Physical elements of Promodel]
4.4.1.2 Entities or Parts:

These are the items processed in the system, which include raw materials, assemblies, loads, finished products etc. These entities can be defined in a very flexible manner. They can be assigned attributes, which can be used in making decisions or gathering specialized statistics. The appearance and the graphic of an entity can be altered as a result of an operation to exhibit physical changes during animation.

4.4.1.3 Path Networks:

These define the probable paths, the entities and resources may travel when moving through the system. These paths are defined graphically and consist of nodes interconnected by path segments. These paths can be shared by one or more resources and entities and the movement along the path is defined in terms of distance, speed or time.

4.4.1.4 Resources

A resource can be a person or a tool or a vehicle that can be either static or assigned to a path network for dynamic. These resources may be used to

- transport material between routing locations
- perform an operation on material at a location
- perform maintenance on location or other resource that is down

Some of the characteristics like speed, acceleration, deceleration, pickup and delivery time can also be specified.

4.4.1.5 Processing or Routing

This element defines the processing sequence and flow logic of entities between different routing locations. Some of the features can be described using the processing element or operation, service time at locations, process logic, input or output relationships, routing conditions and move times or requirements.
4.4.1.6 Arrivals or Production Schedules

These elements can be used to model, deterministic, conditional, or stochastic arrivals. Built in or user defined functions can be used to define inter arrivals times and quantities.

Fig.4.7 : Screenshot of FMS model building setup.

4.5 MATLAB R2012a

Matlab is a high level language that has an interactive environment for numerical computations, programming and visualization [25]. It can be used for analyzing data, developing algorithms and creating models and applications. Matlab in fact stands for MATrix LABoratory. The software is built up around vectors and matrices. It is a great tool for solving algebraic equations, differential equations and for numerical integration. With the help of its tools and inbuilt math functions, it is capable of providing multiple approaches to reach a solution in a short time compared with other traditional programming languages.

The data structures of the Matlab are highly sophisticated. Further, Matlab also has built-in, editing and debugging tools and it supports object oriented programming.
All these factors combine to make Matlab as an excellent research tool. With the presence of powerful built in routines aid, Matlab performs a wide variety of computations. The visualization of results is immediate and is facilitated by easy graphic commands as shown in figure 4.8.

![Automated Analysis Tool for Optimisation of FMS Scheduling](image)

Fig.4.8: Matlab based GUI tool for optimization of FMS scheduling

Various special applications are collected and presented as packages, referred to as tool box covering different disciplines of applied science and engineering. It has many advantages like c, c++, java etc., one of them is, its interactive system that has basic data element as an array that does not require dimensioning.

### 4.5.1 Design of an Automated Tool

The primary objective of this tool is to automate and facilitate scheduling using the best possible approach for a particular job scenario involving multiple machines and multiple jobs. The tool is coded and designed using Matlab software. In this research, data coded in Excel sheet is used to provide the necessary inputs for the tool. The data which include the machine number, job number, machining sequence, batch size, penalty value, reward value and due date are provided in the form of an excel sheet in a predefined format as shown in figures 4.9, 4.10 and 4.11.
Fig. 4.9: Input data include the machine number, job number, machining sequence, batch size, penalty value, reward value and due date.

Fig. 4.10: Input data include the machine number, job number, and machining sequence.
Fig. 4.11: Input data include, batch size, penalty value, reward value, due date

This is a onetime operation and based on this, any number of optimizations can be done using conventional or evolutionary techniques. Once the data is loaded, the number of jobs and the number of machines involved in the data are displayed in the GUI. The tool box has the following sections for easy and simple use of interface for the user.

1. Loading the sequence and setup details as shown in figure 4.12

2. Visualizing the timing details as shown in figure 4.13

3. Interface to run scheduling Priority rules as shown in figure 4.14

4. Interface to run Metaheuristic approaches as shown in figure 4.15

5. Display of the optimized schedule, penalty, idleness and COF values are as shown in figure 4.16.

6. Display of the graphical results is as shown in figure 4.17.
Fig. 4.12: Input data include, batch size, penalty value, reward value, due date for case problem 1

Fig. 4.13: Visualizing the timing details
Fig. 4.14: Interface to execute priority rules

Fig. 4.15: Interface to execute Metaheuristics
Fig. 4.16: Optimized schedule, penalty value, idleness and COF values

Fig. 4.17: Display of the graphical results