CHAPTER 1.0

INTRODUCTION

1.0 INTRODUCTION

Today's dynamic production environment is characterized by a large volume of uncertainty such as rapid market changes, increased product variety, competitive prices and short product life cycles. Therefore, it is of prime importance to introduce flexible manufacturing systems (FMS) so that these uncertainties can be handled in an effective manner.

FMS is characterized as an integrated, computer controlled complex arrangement of automated material handling devices and computer numerically controlled (CNC) machine tools that can simultaneously process medium sized volumes of a variety of part types.

The aim of FMS is to achieve the efficiency of automated high volume mass production while retaining the flexibility of low volume of job shop production. In modern FMS, most of the real time activities such as actual machining operations, computer controls part movements and tool interchange. While an FMS possess the attractive combination of automation and flexibility; the production management problems are rather more as compared to mass production or batch production.

FMS operations can be broadly divided into pre-release and post-release decisions. Pre-release decisions include the FMS operational planning problem that deals with the pre-arrangement of jobs and tools before the processing begins whereas post-release decisions deal with the scheduling problems. Pre-release decisions viz., machine grouping, part type selection, production ratio determination, resource allocation and loading problems must be solved while setting up of a FMS. Amongst pre-release decisions, machine loading is considered as one of the most vital production planning problem because performance of FMS largely depends on it. Loading problem in particular deals with allocation of jobs to various machines under technological constraints with the objective of meeting certain performance measures.

The machine loading problem addressed in this is that although machine capacity might be sufficient, it may not be possible to process all job orders required in particular planning period due to limited number of tool slots and available machine time. Thus subset of job orders is to be processed. It is very difficult to evaluate all possible combinations of

operation-machine allocation in order to achieve minimum system unbalance and maximum throughput. This is because it takes a large search space as well as significant computational time.

As a new approach to enhance the solution quality for machine loading problem, this thesis proposes an iterative method using particle swarm optimization (PSO). The objective function is to minimize the system unbalance and maximize the throughput.

PSO is a population-based evolutionary computation technique based on the movement and intelligence of swarms. The technique is developed by Russel Eberhart (Electrical Engineer) and James Kennedy (Social Psychologist) in 1995 (both U. Indiana, Purdue), inspired by the social behavior of birds studied by Craig Reynolds (a Biologist) in late 80's and early 90's. He derived a formula for representation of the flocking behavior of birds. Eberhart and Kennedy recognized the suitability of this technique for optimization.

It uses a number of agents, i.e. particles, that constitute a swarm moving around in the search space looking for the best solution. Each particle is treated as a point in N-dimensional space which adjusts its "flying" according to its own flying experience as well as the flying experience of other particles.

In PSO, each member is called particle, and each particle moves around in the multidimensional search space with a velocity which is constantly updated by the particle's own experience and the experience of the particle's neighbors or the experience of the whole swarm. The members of the entire population are maintained throughout the search procedure so that information is socially shared among individuals to direct the search towards the best position in the search space. Two variants of the PSO algorithm have been developed, namely PSO with a local neighborhood, and PSO with a global neighborhood. According to the global neighborhood, each particle moves towards its best previous position and towards the best particle in the whole swarm, called the gbest model in the literature. On the other hand, based on the local variant so called the pbest model, each particle moves towards its best previous position and towards the best particle in its restricted neighborhood. Generally, PSO is characterized as a simple heuristic of well balanced mechanism with flexibility to enhance and adapt to both global and local

exploration abilities. Compared with GA, PSO has some attractive characteristics. It has memory that enables to retain knowledge of good solutions by all particles whereas previous knowledge of the problem is destroyed once the population changes in GAs. Due to the simple concept and easy implementation, PSO has gained much attention and been successfully applied to a wide range of applications such as system identification, neural network training, mass-spring system, task assignment, supplier selection and ordering problem, power and voltage control etc.

CHAPTER 2.0

LITERATURE REVIEW

The first mathematical formulation of the FMS loading problem was given by Stecke [9] Grouping and loading are formulated as non-linear 0-1 mixed integer programs. Allocate the operations and associated cutting tools of a selected set of part types among the machine groups subject to the technological and capacity constraints of the FMS and according to some loading objective. These problems are formulated in all detail as nonlinear mixed integer programs (the nonlinear terms are products of 0-1 integer variables).

Vidyarthi and Tiwari [3] proposed a fuzzy-based methodology to solve the machine-loading problem in an FMS. The job-ordering determination before loading is carried out by evaluating the membership contribution of each job to its characteristics, such as batch size, essential operation processing time, and optional operation processing time. The operation–machine allocation decisions are made based on the evaluation of the membership contribution of an operation– machine allocation vector.

Several heuristic solution based methods for the machine loading have been developed. M.K.Tiwari, B.Hazarika [4] developed a heuristic solution approach to the machine loading problem of an FMS, they used fixed pre-determined job ordering/job sequencing rule as input to their proposed heuristic. A Petri net model for the problem attempted by the proposed heuristic has been constructed to delineate its graphical representation and subsequent validation. A petrinet is a formal graph model for description and analysis of systems that exhibit both synchronous and concurrent properties and thus are well suited to model the dynamics of an FMS.

Nagarjuna N, Mahesh O [6] proposed a heuristic based on a multi-stage programming approach for minimizing the system unbalance while satisfying constraints such as the availability of tool slots and machining time. They developed a heuristic for job sequencing and operation allocation concurrently. Some studies [4],[5] generate job sequence based on either predetermined sequencing rules such as shortest processing time (SPT) rule or employing meta heuristics such as simulated annealing (SA), Genetic algorithm (GA) etc. and operation allocation problem is considered next. Based on multi-stage programming approach, the given problem is divided into number of stages. The maximum number of stages is equal to the number of jobs to be loaded. At each stage partial sequences are generated and evaluated. Only those partial sequences, which are feasible and

perform better in achieving the objective function, are retained and new partial sequences are generated from these and evaluated. This process is repeated iteratively until a best feasible job sequence with operation allocation on machines is obtained.

Swarnkar R, Tiwari MK [5] have addressed machine-loading problem having the bi-criterion objectives of minimizing system unbalance and maximizing the throughput using a hybrid algorithm based on tabu search (TS) and simulated annealing (SA). The main advantage of this approach is that a short-term memory provided by the tabu list (tabulist keeps track of previously visited solution, so that revisits are prevented, but it has a deterministic nature and cannot avoid cycling.) can be used to avoid revisiting the solution while preserving the stochastic nature (it can avoid cycling, but doesn't keep any track of previously visited solutions.) of the SA method.

Genetic algorithm (GA)-based approaches for loading problems is found to ensure an optimal solution and to be less computationally intensive. Tiwari and Vidyarthi [10], have addressed machine-loading problems having the bi-criterion objectives of minimizing system unbalance and maximizing the throughput.

S. G. Ponnambalam, and Low Seng Kiat [7], proposed a particle swarm optimization algorithm to solve machine loading problem in flexible manufacturing system (FMS), with bicriterion objectives of minimizing system unbalance and maximizing system throughput in the occurrence of technological constraints such as available machining time and tool slots. The objective function values obtained by the two algorithms are the same. The advantage of the proposed PSO is its shorter computational time.

As a new approach to enhance the solution quality for machine loading problem, this thesis proposes an iterative method using particle swarm optimization (PSO). The objective function is to minimize the system unbalance and maximize the throughput. Using this meta-heuristic, based on position values, by applying SPV rule the job sequence is obtained and then allocation in machines is done. The main advantage of this algorithm is that with in few iterations (less than to 20) the required objective function can be obtained, whereas other meta-heuristics are considered they require more number of iterations (about 50). So,

by reducing number of iterations the computational time reduces and it requires less computational effort too.

The PSO algorithm is coded in C^{++} programming language to obtain the best job sequence for which system unbalance is minimum and throughput is maximum; thus overall objective function is maximum among given number of iterations. The results are compared with manual calculations as well as with the results of open literature.

CHAPTER 3.0

FLEXIBLE MANUFACTURING SYSTEMS

FLEXIBLE MANUFACTURING SYSTEMS

3.1 Definition

A flexible manufacturing system (FMS) is an integrated system of computer numerically controlled (CNC) machine tools, each having automatic tool interchange capability, and all are connected by an automated material handling system that together operates as an integrated system under the control of a central computer (s).

Flexible manufacturing system is realized to be an efficient alternative to conventional manufacturing that allows simultaneous machining of small to medium batches of a variety of part types. Parts can flow through the system in unit batch sizes.

3.2 Components of an FMS:

There are three basic components of flexible manufacturing systems.

1. Material handling and storage
2. Processing stations
3. Computer control system.

3.2.1 Material handling and storage

Various types of automated material handling equipments are used to transport the work parts and subassemblies between the processing stations, sometimes incorporating storage into the function.

The material handling and storage system in a FMS should perform the following function.

1. Random, independent movement of work parts between workstations:

This means that parts must be capable of moving from any one machine in the system to any other machine. This allows the system to achieve various processing sequences on the different machines in the cell and to make substitutions when certain machines are busy.

2. Handle a variety of Work part configurations:

For prismatic parts, this is usually accomplished by using pallet fixtures in the handling system. The fixture is located on the top face of the pallet and is designed to accommodate different part configurations by means of common components, quick-change features, and other devices that permit a rapid buildup of the fixture for a given part. The base of the pallet is designed for the material handling system. For rotational parts, industrial robots are often used to load and unload the turning type machine tools and to transfer parts between workstations.

3. Temporary Storage

The number of parts in the FMS typically exceeds the number of parts actually being processed. In this way, each machine can have a queue of parts waiting to be processed. Thus helps of increase machine utilization.

4. Convenient access for loading and unloading work parts:

The handling system must provide a means to load and unload parts from the FMS. This is often accomplished by having one or more load/unload stations in the system. Manual operators are used to build up the pallet fixtures, load the parts and unload the finished parts when processing has been completed.

5. Compatible with computer control:

The handling system must be capable of being controlled directly by the computer to direct it to its various workstations, load and unload stations and so on.

3.2.2 Processing Stations

The processing stations are typically computer numerical control (CNC) machine tools that perform machining operations on families of parts. The processing or assembly equipment used in a FMS depends on the type of work that is accomplished on the system. In a system designed for machining operations, the principal types of processing station are CNC machine tools.

Types of machines used in FMS work stations are:

1. Machining Centers
2. Head changers
3. Head Indexes
4. Milling Modules
5. Turning Modules
6. Assembly Work Stations
7. Inspection Stations
8. Sheet metal processing Machines
9. Forging Stations

3.2.3 Computer Control System

The operation of a FMS is computer controlled. Computer control is used to coordinate the activities of the processing stations and the material handling system in the FMS.

The function performed by the FMS computer control system are :

1. Control of each work station
2. Distribution of control instructions to work stations
3. Production control
4. Traffic control
5. Shuttle control
6. Work handling system monitoring
7. Tool control
8. System performance monitoring and reporting.

3.3 Types of Flexibilities

There are four types of flexibilities, namely:

i. Machine flexibility
ii. Product flexibility
iii. Routing flexibility
iv. Operation flexibility

3.3.1 Machine Flexibility

This measures the case with which a machine can change over from one-part types to another. The change over time generally involves setup, tool changing, and part-program transfer and transportation times. The changeover time could be used as a single characteristic to measure machine flexibility. The universe of part types that a machine can produce is another measure of machine flexibility.

3.3.2 Product Flexibility

This is a measure of the ability to change over to anew product mix economically and quickly. The changeover time in this case includes the design, process planning, part program preparation; tool development and fixture would certainly heighten the company's potential responsiveness to a competitor or to market changes.

3.3.3 Routing Flexibility

The flexibility measures the ability to manage internal changes such as breakdown and failures. One can define the routing flexibility as the ability to produce the given set of part types in required volumes in the face of failures and breakdowns. The failures could be tool break, controller or machine breakdowns.

3.3.4 Operation Flexibility

This is the ability to interchange the ordering of several operations for each part type. There is usually some required partial predetermined structure for a particular part type. The operation flexibility is part dependant and it increases the number of alternate routes that a part can flow through on the factory floor.

3.4 Decision Problems in FMS Design and Operation

Stecke [9] observes that the design and use of flexible manufacturing systems involve some intricate operations research problems, such as :

1. Design problems
2. Planning problems
3. Scheduling problems
4. Control problems

3.4.1 FMS Design Problems

In developing an FMS design, there is a partial ordering of some of the decisions that have to be made. Some decisions must precede others in time. These can be partitioned into initial specification decisions and subsequent implementation decisions. These decisions are described as follows:

1. Initial Specification Decisions

- At first, the manufacturing requirements need to be specified. Determine the range of families or components or part types to be produced.
- Determine how these parts shall be manufactured. The consideration eventually specifies the numbers and types of machine tools and robots that are required.
- Specify what type of different flexibilities is required or desired and the amounts of each.
- Determine the type of FMS that shall be developed.
- Specify the type, then capacity of the material handling system.
- The type, and then the size, of the buffers has to be specified.
- The hierarchy, among the computers controlling the different aspects of production has to be specified.
- The vendors have to be chosen.

2. Subsequent Implementation Decisions

- The layout of the FMS has to be determined.
- The number and design of pallets has to be determined.
- The number and design of the feature of each fixture type has to be determined.
- The strategies for running the FMS have to be specified.

The software development tasks need to be specified and implemented perhaps within a project management setting.

3.4.2 FMS Planning Problems

- From the list of part types for which production orders of various size are specified, choose a subset of part types for immediate and simultaneous manufacture.
- Partition the machines of each type into machine groups.
- Determine the production ratios at which the selected set of part types should be produced in the system.
- Allocate the limited number of pallets and the fixtures of each fixture type among the selected part types.
- Allocate the operations and the association cutting tools of the selected part types among the machines.

3.4.3 FMS Scheduling Problems

- Determine the optimal sequence at which the parts of the selected part types are to be input into the system.
- Develop appropriate scheduling methods and algorithms.
- If there are several parts waiting to be processed by the same machine tool, determine the priorities among these parts.

3.4.4 FMS Control Problems

- Determine a policy to handle machine tool and other breakdown. (The policy should be determined during the design phase. It is implemented during the control of an FMS).
- Determine the schedule, periodic, preventive maintenance policies.
- Determine in-process and/or finished goods inspection policies.

Procedures for tool life and process monitoring and data collection, as well as for updating the estimates of tool life, have to be specified.

3.5 Machine Loading Problem

The machine loading problem in an flexible manufacturing system (FMS) is specified as to assign the machine, operations of the selected jobs, and the tools necessary to perform these operations by satisfying the technological constraints (available machine time and tool slot constraint) in order to ensure the minimum system unbalance and maximum throughput, when the system is in operation.

3.5.1 Problem Description

The FMS under consideration consists of a number of multifunctional CNC machines, tools with the potential to execute several operations, automated material handling devices and other amenities, where several types of jobs arrive with varied processing requirements.

Jobs are to be sequenced and processed over a given planning horizon. The loading problem addressed in this is that, although machine capacity might be sufficient, it may not be possible to process all the job orders required in particular planning period due to limited number of tool slots and available machining time. Thus subsets of job orders are to be processed.

Jobs are available in batches and each job has one or more operations. Each operation can be performed by one or more machines. The processing time and tool slots

required for each operation of the job and its batch size are known before hand. Essential and optional types of operations are allied with each job. Essential operations of a job mean that this operation can be performed only on a particular machine using a certain number of tool slots. Whereas, optional operations imply that they can be carried out on a number of machines with same or varying processing time and tool slots. In this problem, the flexibility lies in the selection of a machine for processing the optional operation of the job. The operation machine allocation combinations are to be evaluated using two common yardsticks, system unbalance and throughput. System unbalance can be defined as the sum of underutilized and over utilized time on all the machines available in the system. Minimization of the system unbalance is same as maximization of machine utilization. Whereas throughput refers to the unit of part types produced.

The complexity associated with machine loading problem given in Table 1 can be exemplified as follows:

There exist 8 jobs, which can be sequenced in 8! ways, all together 2592 combinations of operations. Machine allocations are possible for one of all the job sequences. Here for 8! Job sequences the total number of possible allocations turns out to be 2592*8! = 104509440.Some of these allocations are not possible because they are not able to satisfy system constraints. It is fairly difficult to evaluate optimal solutions of operation machine allocations on machines in the presence of several alternatives for the given problem. Therefore to arrive at an optimal / near optimal solution, Particle Swarm Optimization algorithm has been used to solve the complexities of the machine-loading problem.

3.5.2 Formulation of Objective Function and Constraints

The above-described problem is formulated as the bi-criterion objective problem where the two objectives are combined.

The first objective is to minimize the system unbalance:

Minimize $\sum_{m=1}^{M}\left(UT_m - OT_m\right)$

This is equivalent to maximize the system utilization:

Maximize $F_1 = \dfrac{M * H - \sum_{m=1}^{M}\left(UT_m - OT_m\right)}{M * H}$ ----------(1)

The second objective is maximizing throughput or equivalently maximizing the system efficiency.

Maximize $F_2 = \dfrac{\sum_{j=1}^{J} B_j * X_j}{\sum_{j=1}^{J} B_j}$ ----------(2)

Thus the overall objective function is

Maximise F = $\dfrac{M * H - \sum_{m=1}^{M}\left(UT_m - OT_m\right)}{M * H} + \dfrac{\sum_{j=1}^{J} \boldsymbol{B}_j * \boldsymbol{X}_j}{\sum_{j=1}^{J} \boldsymbol{B}_j}$

Thus F = F1 + F2 ----------(3)

Constraints

1. Tool Slots:

The constraint guarantees that the number of tool slots needed for the operation of the jobs to be performed on a machine must always be less than or equal to the total slots available in that machine. This constraint can be expressed as:

$$\sum_{K=1}^{P_i} T_{jm}\ x_{ijk} \le T_{jm}^{a} \qquad \text{i = 1, 2,…N} \qquad (1)$$

2. Available time on machine:

The available time on each machine should be greater than or equal to the time required by the next job to be assigned to this machine.

$$\sum_{K=1}^{P_i} t_{jm}\ x_{ijk} \le t_{jm}^{a} \qquad \text{i= 1,2,…,N} \qquad (2)$$

3. System Unbalance:

System unbalance equals to the sum of the idle time remaining on machines after allocation of all feasible jobs. The value of system unbalance must either be zero (100% utilization of system) or a positive value.

$$\sum_{m=1}^{M} \left(UT_m - OT_m\right) \ge 0 \qquad (3)$$

4. Non Splitting of Job:

This constraint implies that once a job is considered for processing, all the operations are to be completed before undertaking a new job.

5. The number of tool slots and remaining time on any machine after any assignment of job should always be positive or zero.

6. Unique Job Routing:

Despite the flexibility existing in the selection of a machine for optimal operations, once a machine is selected, the operation has to be completed on the same machine.

7. Integrity of decision variables:

The decision variables possessing the value of 0 and 1 integers are as follows:

$$x_i = \begin{cases} 1 & \text{if job } i \text{ is selected} \\ 0 & \text{otherwise,} \quad i=1,2,\ldots,N \end{cases} \tag{4}$$

$$x_{ij} = \begin{cases} 1 & \text{if job } i \text{ is assigned to machine } j \\ 0 & \text{otherwise,} \quad i=1,2,\ldots,N \quad j=1,2,\ldots,M \end{cases} \tag{5}$$

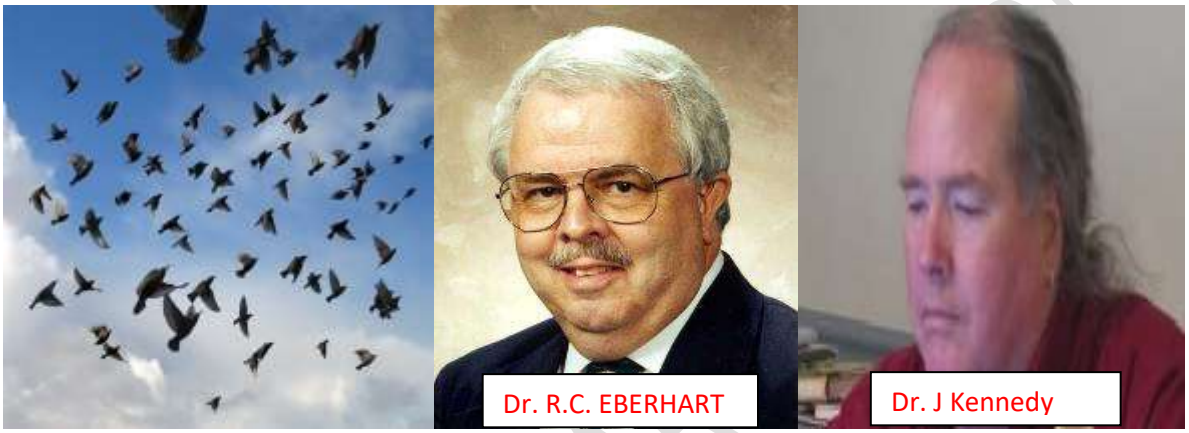
$$x_{ijk} = \begin{cases} 1 & \text{if operation } k \text{ of job } i \text{ is assigned to machine } j \\ 0 & \text{otherwise,} \quad i=1,2,\ldots,N, \quad j=1,2,\ldots,M, \quad k=1,2,\ldots,P \end{cases} \tag{6}$$

CHAPTER 4.0

PARTICLE SWARM OPTIMIZATION

4.1 Introduction

Particle swarm optimization (PSO) is a population based evolutionary computation technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling.



Particle swarm optimization has roots in two main component methodologies. Perhaps more obvious are its ties to artificial life (A-life) in general, and to bird flocking, fish schooling, and swarming theory in particular. It is also related, however, to evolutionary computation, and has ties to both genetic algorithms and evolutionary programming.

PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles.

Compared to GA, the advantages of PSO are that PSO is easy to implement and there are few parameters to adjust. PSO has been successfully applied in many areas: function optimization, artificial neural network training, fuzzy system control, and other areas where GA can be applied.

There are two popular swarm inspired methods in computational intelligence areas: Ant colony optimization (ACO) and particle swarm optimization (PSO). ACO was inspired by the behaviors of ants and has many successful applications in discrete optimization problems.

The particle swarm concept originated as a simulation of simplified social system. The original intent was to graphically simulate the choreography of bird of a bird block or fish school. However, it was found that particle swarm model can be used as an optimizer.

PSO learned from the scenario and used it to solve the optimization problems. In PSO, each single solution is a "bird" in the search space. We call it "particle". All of particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In every iteration each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called pbest. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called gbest.

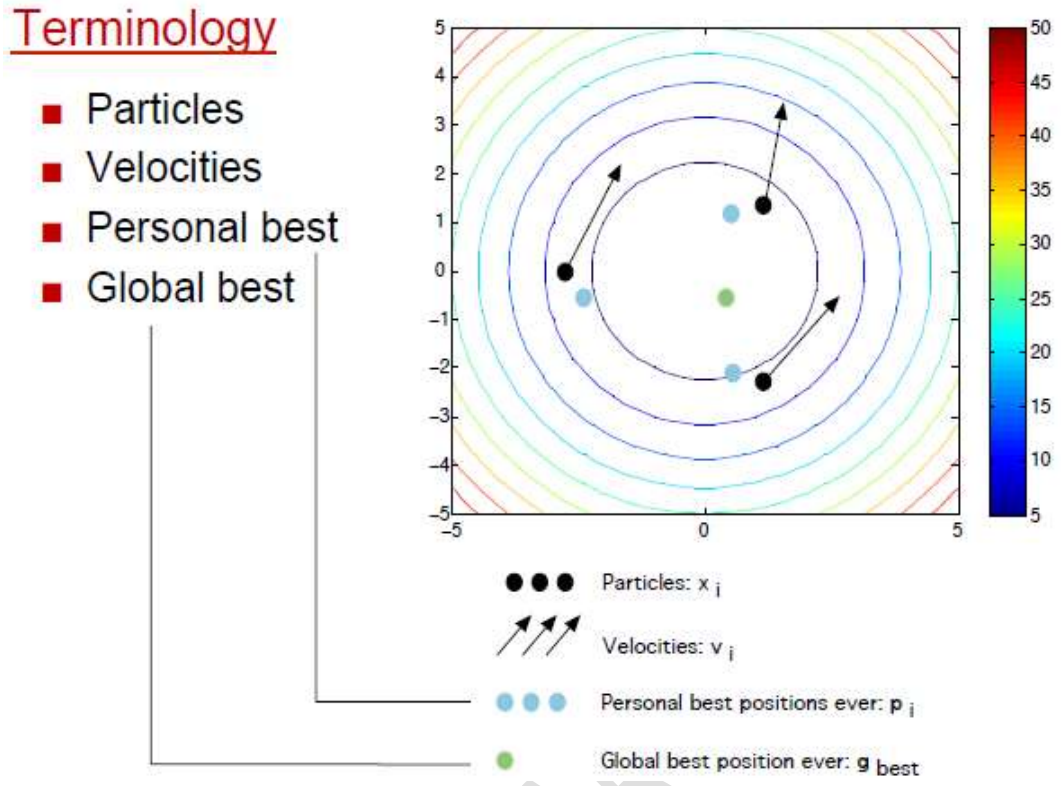


Fig. 4.1

After finding the two best values, the particle updates its velocity and positions with following equation (1) and (2).

$$V_{ij}^{t} = w^{t} * V_{ij}^{t-1} + C_1 r_1 \left(P_{ij}^{t-1} - X_{ij}^{t-1}\right) + C_2 r_2 \left(g_j^{t-1} - X_{ij}^{t-1}\right) \text{----------(4)}$$

$$X_{ij}^{t} = V_{ij}^{t} + X_{ij}^{t-1} \text{----------(5)}$$

$$W^{t} = w^{t-1} * \alpha \text{----------(6)}$$

Where

V_{ij}^{t} represents velocity of particle i at iteration t with respect to jth dimension (j =1,2,……n).

P_{ij}^{t-1} represents the position value of the ith personal best with respect to the jth dimension.

X_{ij}^{t} is the position value of the ith particle with respect to jth dimension.

X_{ij}^{t-1} is previous position value

g_j^{t-1} previous global best value of the particle

C1 and C2 are positive acceleration parameters, called cognitive and social parameter, respectively and r1 and r2 are uniform random numbers between (0,1).
W is known as inertia weight, α is a decrement factor.
The parameter 'w' controls the impact of the previous velocities on the current velocity.

4.2 PSO Algorithm:

As stated before, PSO simulates the behaviors of bird flocking. Suppose the following scenario: a group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. All the birds do not know where the food is. But they know how far the food is in each iteration. So what's the best strategy to find the food? The effective one is to follow the bird which is nearest to the food.

t = 0;
for (j=1,2..J); //swarm size
Generate V_{ij}^t, X_{ij}^t; //X in between 0 & 4, V in between -4 & 4
Search on all particle positions;
Obtain job sequence using SPV rule;
Evaluate F; //objective function value
$P_{ij}^t \longrightarrow X_{ij}^{t-1}$; //personal best position is previous iterations best position of that particle
$G^t \longrightarrow X_{ij}^{t-1}$; //Global best position is previous iterations best position among all particle positions
While ($t<t_{max}$)
{
t = t+1;
for (j=1,2..J);

update velocity V_{ij}^{t};

update position X_{ij}^{t};

search on all particle positions;

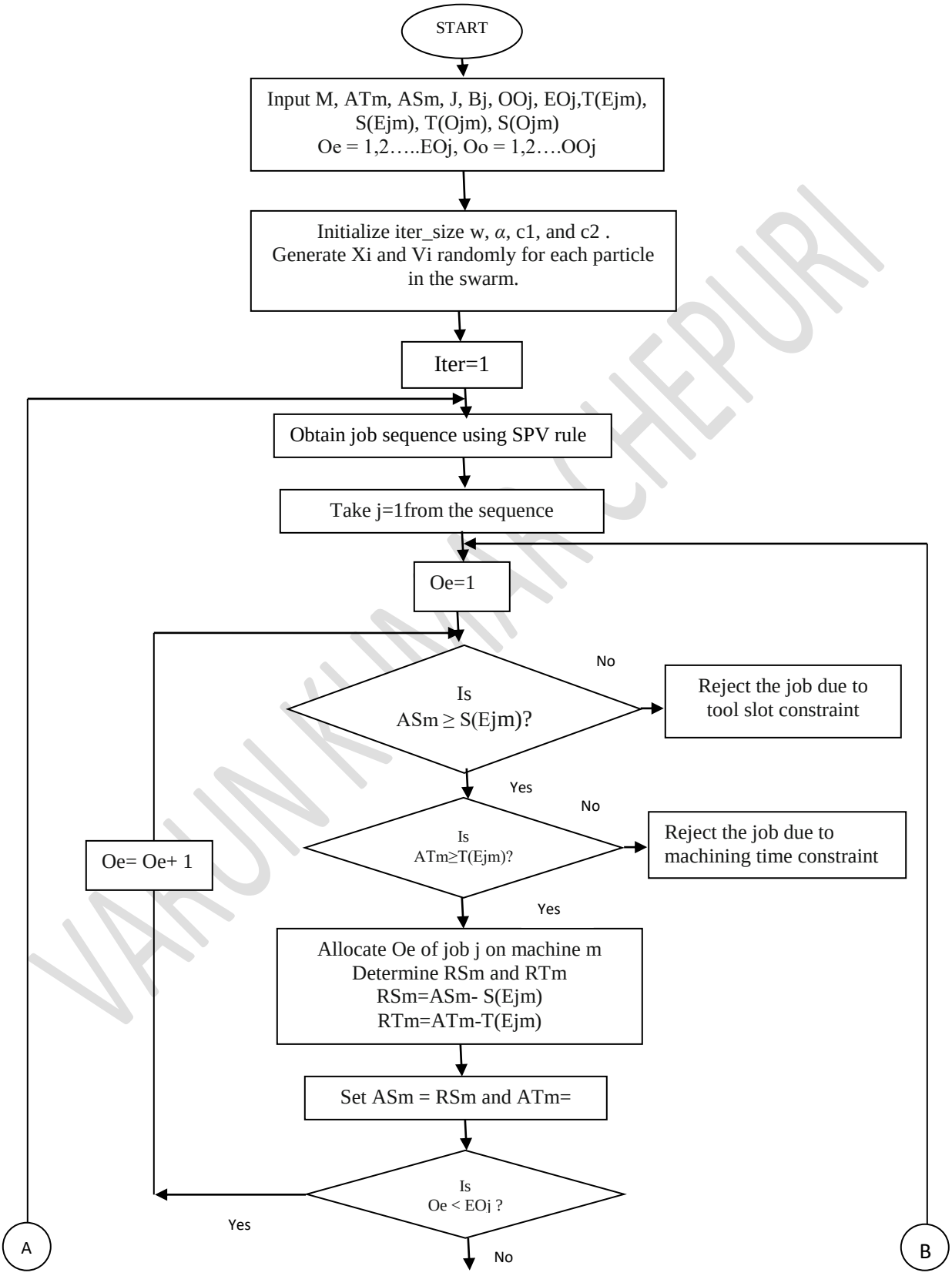
obtain job sequence using SPV rule;

Evaluate F;

Update P_{ij}^{t} & G^{t}

}

4.3 PSO FLOWCHART



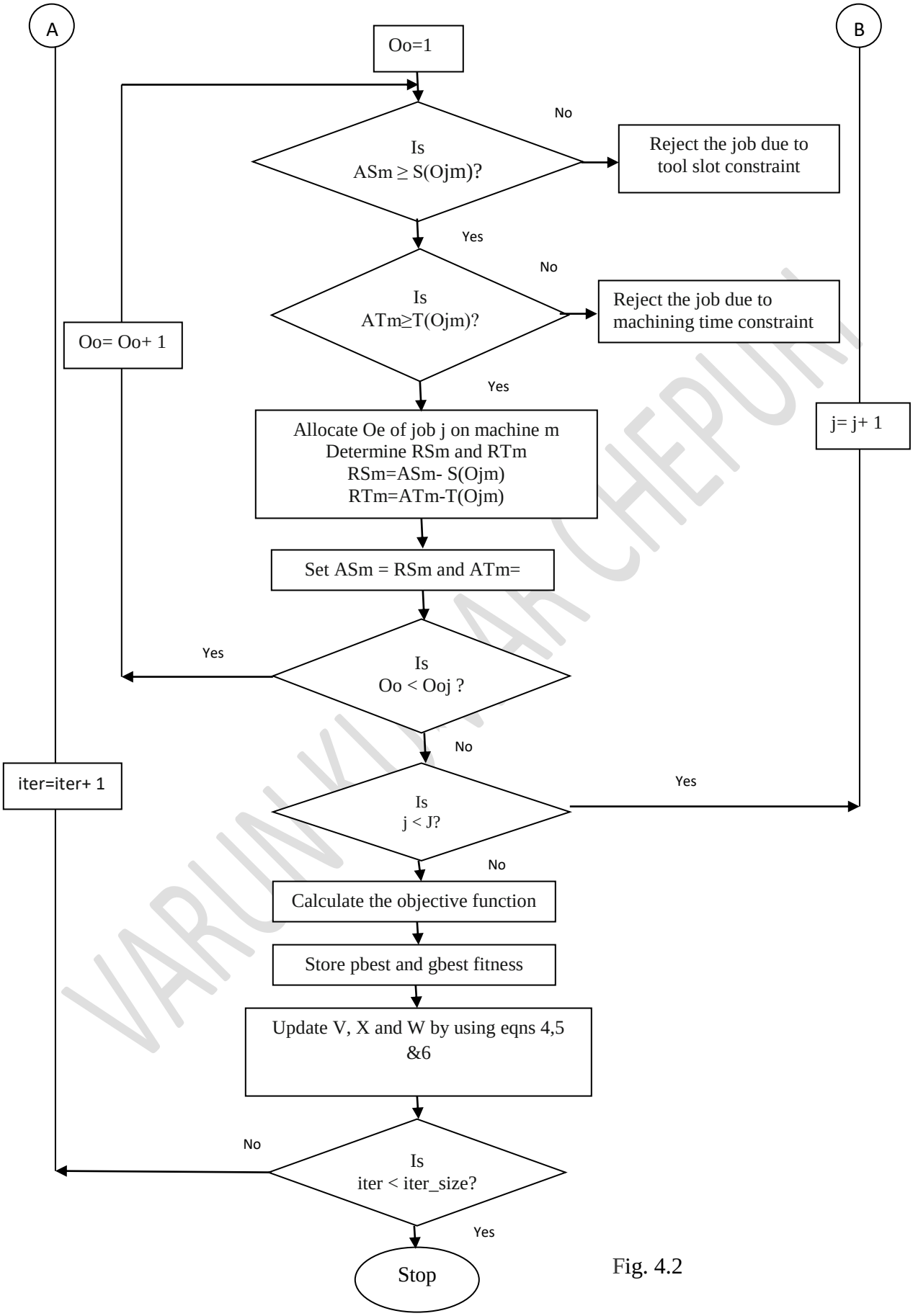


Fig. 4.2

4.4 Some applications of PSO

Since its initial development PSO has had an exponential increase in applications (either in the algorithm's original form or in an improved or differently parameterized fashion). A first look into the applications of this algorithm was presented by one of its creators Eberhart and Shi in 2001focusing on application areas such as: Artificial Neural Networks training (for Parkinson Diagnostic), Control Strategy determination (for electricity management) and Ingredient Mix Optimization (for micro organisms strains growth).

Later in 2007 a survey by Riccardo Poli reported the exponential growth in applications and identified around 700 documented works on PSO. He also categorized the areas of applications in 26 categories being those most researched (with around or more than 6% of the works reviewed by poli)

Antennas - especially in its optimal control and array design. Aside from there, there are many others like failure correction and miniaturization. Control especially in PI (Proportional Integral) and PID (Proportional Integral Derivative) controllers. These systems control a process based on its current output and its desired value (the difference between this two is its current error, P), past errors (I) and a prediction of future errors(D); Distribution Networks - especially in design/restructuring and load dispatching in electricity networks. Electronics and Electromagnetic - here the applications are very disperse, but some of the main applications are: on-chip inductors, fuel cells, generic design and optimization in electromagnetics semi-conductors optimization Image and Video - this area is the one with most documented works in a wide range of applications, some examples are: face detection and recognition, image segmentation, image retrieval, image fusion, microwave imaging, contrast enhancement, body posture tracking. Power Systems and Plants - especially focused are power control and optimization.

However, other specific applications are: load forecasting, photovoltaic systems control and power loss minimization. Scheduling - especially focused are flow shop scheduling, task scheduling in distributed computer systems, job-shop scheduling and holonic manufacturing systems. But other scheduling problems are addressed such as assembly, production, train and project.

The other categories identified are: Biomedical, Communication Networks, Clustering and Classification, Combinatorial Optimization, Design, Engines and Motors, Entertainment, Faults, Financial, Fuzzy and Neuro-fuzzy, Graphics and Visualization, Metallurgy, Modeling, Neural Networks, Prediction and Forecasting, Robotics, Security and Military, Sensor Networks, Signal Processing.

CHAPTER 5.0

PARTICLE SWARM OPTIMIZATION FOR MACHINE LOADING PROBLEM

This chapter proposes particle swarm optimization algorithm for machine loading problem in FMS. In this problem; the number of jobs are considered as number of particles and the particles initial positions and velocities are adopted from Sandhyarani Biswas [8]. This algorithm uses smallest position value (SPV) rule for sequencing the jobs.

5.1 Procedure to Enumerate System Unbalance and Throughput

Step 1:

Define initial conditions

X_{ij}^{t}, V_{ij}^{t}, C1=C2=2, α=0.9, w=0.85, r1=0.2, r2=0.3

(positions, velocities and constants are adopted from Sandyarani Biswas [8])

Step 2:

Adopt smallest position value (SPV) rule for job sequencing

Step 3:

- Take 1st job from the sequence, calculate the time required for essential operation(s) and load the job 'j' on machine 'm'.
- Now calculate the time required for optional operation(s) of the same job and load on the required machine(s).

Step 4:

Calculate the System Unbalance (Sum of remaining time on all the machines).

$$SU = \sum_{m=1}^{M} RTm$$

Where $RT_m = AT_m - T(E_{jm}) - T(O_{jm})$

Step 5:

If the 'SU' is positive allocate the job 'j' on machine 'm' virtually, else reject the job and go to next step.

Step 6:

- Check, whether the available tools slots on machine 'm' is greater than or equal to the tool slots required by the job 'j'.
- If yes assign the job 'j' on machine 'm', else reject the job.

Step 7:

If the job is assigned, allocate it to set 'AS', else allocate to set 'UA'

Step 8:

After assigning the job, store the remaining time and tool slots on each machine for next job.

Step 9:

Take next job in sequence and repeat steps 3-8 for all operations of job 'j' and continue till last job.

Step 10:

Final job allocation set is to be chosen based on minimum positive system Unbalance or zero.

Step 11:

Determine throughput, which is the sum of batch size of jobs allocated to set AS.

$\sum_{j=1}^{J} B_j * X_j$ Where, B_j is batch size, X_j is 1if job j is selected otherwise 0.

5.2 Methodology

Step 1:

Table 1, initialize positions and velocities for all the particles randomly (adopted from Sandhyarani Biswas [8]).

Job,j	1	2	3	4	5	6	7	8
X_{ij}^t	1.67	2.82	1.23	0.11	3.47	0.57	0.98	3.72
V_{ij}^t	2.98	-0.87	1.51	-3.54	0.45	2.32	-1.50	3.04

The initial feasible solution for the problem is generated using smallest position value (SPV) job sequencing rule .The system unbalance and throughput are calculated for the initial solution. Calculate the objective function value also.

1.1 The initial positions can be calculated from the following equation

$X_{ij}^{t} = X_{min} + (X_{max} - X_{min})*U(0, 1)$

Where $X_{min} = 0$, $X_{max} = 4$ (search space 0 to 4)

U (0, 1) is uniform random number between 0 & 1

$X_{ij}^{t} = 0 + (4 - 0) * 0.0275 = 0.11$

$X_{ij}^{t} = 0 + (4 - 0) * 0.1425 = 0.57$

$X_{ij}^{t} = 0 + (4 - 0) * 0.245 = 0.98$

$X_{ij}^{t} = 0 + (4 - 0) * 0.3075 = 1.23$

$X_{ij}^{t} = 0 + (4 - 0) * 0.4175 = 1.67$

$X_{ij}^{t} = 0 + (4 - 0) * 0.705 = 2.82$

$X_{ij}^{t} = 0 + (4 - 0) * 0.8675 = 3.47$

$X_{ij}^{t} = 0 + (4 - 0) * 0.93 = 0.11$

(The above position values are randomly arranged for 8 particles)

1.2 The initial velocities can be calculated from the following equation

$V_{ij}^{t} = V_{min} + (V_{max} - V_{min})*U(0, 1)$

Where $V_{min} = -4$, $V_{max} = 4$

U(0,1) is uniform random number between 0 & 1

$V_{ij}^{t} = -4 + (4 – (-4)) * 0.0575 = -3.54$

$V_{ij}^{t} = -4 + (4 – (-4)) * 0.3125 = -1.5$

$V_{ij}^{t} = -4 + (4 – (-4)) * 0.39125 = -0.87$

$V_{ij}^{t} = -4 + (4 – (-4)) * 0.55625 = 0.45$

$V_{ij}^{t} = -4 + (4 – (-4)) * 0.68875 = 1.51$

$V_{ij}^{t} = -4 + (4 – (-4)) * 0.79 = 2.32$

$V_{ij}^{t} = -4 + (4 – (-4)) * 0.8725 = 2.98$

$V_{ij}^{t} = -4 + (4 – (-4)) * 0.88 = 3.04$

(The above velocity values are randomly arranged for 8 particles)

Step 2:

After assigning positions and velocities for 8 jobs, based on smallest position value (SPV) rule, obtain the job sequence i.e. 4, 6, 7, 3, 1, 2, 5, 8

Step 3:

Calculate the system unbalance (F1) and throughput (F2) then objective function value (F)

Step 4:

Find out the personal (pbest) best of each particle and the global best (gbest).

Step 5:

If no progress in the pbest value is observed for an elapsed period, carry out the mutation of a particle using the Gaussian mutation strategy [11].

Mutate (Xid) = Xid (1 + gaussian(σ))

Where gaussian σ returns a random number drawn form a gaussian distribution with a standard deviation of σ. The value of σ chosen by the authors is 0.1 times the length of search space

Step 6:

Update the position, velocity, and inertia weight by using eqs. 4, 5 and 6 respectively.

Step 7:

Now find the new job sequence by using SPV rule and calculate the objective function.

Step 8:

Find the new personal best (pbest) and the global best (gbest).

Step 9:

Terminate if the maximum number of iterations is reached and store the gbest and the maximum objective function value; otherwise, go to Step 1.

CHAPTER 6.0

RESULTS AND DISCUSSIONS

RESULTS AND DISCUSSIONS

6.1 Illustrative example

For the present work the problem from M.K.Tiwari, B.Hazarika [4] has been considered for solving machine loading problem of FMS to lend support to the present methodology.

Table 2: Description of Problem

Job Number	Operation Number	Batch Size	Unit Processing time(min)	Machine number	Tool Slot needed
1	1	8	18	3	1
2	1	9	25	1,4	1
	2		24	4	1
	3		22	2	1
3	1	13	26	4,1	2
	2		11	3	3
4	1	6	14	3	1
	2		19	4	1
5	1	9	22	2,3	2
	2		25	2	1
6	1	10	16	4	1
	2		7	4,2,3	1
	3		21	2,1	1
7	1	12	19	3,2,4	1
	2		13	2,3,1	1
	3		23	4	3
8	1	13	25	1,2,3	1
	2		7	2,1	1
	3		24	1	3

The problem consists of 4 machines, the maximum available time on each machine is 480min. for each machine the maximum number of tool slots are 5. Therefore the maximum total available time on the machines is 4 * 480 = 1920min.

The initial positions and velocities [8],

Job,j	1	2	3	4	5	6	7	8
X_{ij}^t	1.67	2.82	1.23	0.11	3.47	0.57	0.98	3.72
V_{ij}^t	2.98	-0.87	1.51	-3.54	0.45	2.32	-1.50	3.04

To obtain the initial solution, SPV job sequencing rule is used.

The initial solution is = {4, 6, 7, 3, 1, 2, 5, 8}

The Table 3 gives the procedure to calculate the system unbalance and throughput for the initial solution.

The system unbalance for the assigned job set SU=141

The set of assigned jobs AS = {4, 6, 7, 3}

The set of unassigned jobs UA= {1, 2, 5, 8}

$$\text{Throughput} = \sum_{j=1}^{J} \text{Batch size of jobs assigned.}$$

Throughput, TH = 41.

Objective function F(S) = 1.439

6.2. To find the job sequences by using PSO metaheuristic:

Step 1: Consider initial positions and velocities randomly

The initial positions and velocities of particles are (adopted from Sandhyarani Biswas [8])

Job,j	1	2	3	4	5	6	7	8
X_{ij}^t	1.67	2.82	1.23	0.11	3.47	0.57	0.98	3.72
V_{ij}^t	2.98	-0.87	1.51	-3.54	0.45	2.32	-1.50	3.04

Step 2: Based on SPV rule the job sequence for above positions is: 4, 6, 7, 3, 1, 2, 5, 8. For the current job sequence the SU = 141, TH = 41, F = 1.439

Step 3: For iteration 2 the new job sequence is required, in order to obtain the new sequence the positions and velocities are need to be updated as follows:

i. Velocity update:

$$V_{ij}^t = w^t * V_{ij}^{t-1} + C_1 r_1 \left(P_{ij}^{t-1} - X_{ij}^{t-1}\right) + C_2 r_2 \left(g_j^{t-1} - X_{ij}^{t-1}\right)$$

$$V_{21}^t = w^t * V_{21}^{t-1} + C_1 r_1 \left(P_{21}^{t-1} - X_{21}^{t-1}\right) + C_2 r_2 \left(g_j^{t-1} - X_{21}^{t-1}\right)$$

ii. Inertia weight update:

$$W^t = w^{t-1} * \alpha$$

Where, w^{t-1} = 0.85, α = 0.9

W^t = 0.765

V_{21}^{t-1} = 2.98, P_{21}^{t-1} = 1.67, X_{ij}^{t-1} =1.67, g_j^{t-1} = 0.11, C1 = C2 = 2, r1 = 0.2, r2 = 0.3

V_{21}^t = 0.765 * 2.98 + (2*0.2)(1.67-1.67)+(2*0.3)(0.11-1.67)

V_{21}^t = 1.3457

iii. Position update:

$$X_{ij}^t = V_{ij}^t + X_{ij}^{t-1}$$

$$X_{21}^t = V_{21}^t + X_{21}^{t-1}$$

X_{21}^t = 1.3457 + 1.67 = 3.0157

Similarly the new positions and velocities are to be calculated for the remaining particles

Table 4. The new positions and velocities at iteration 2:

Job,j	1	2	3	4	5	6	7	8
X_{ij}^t	3.0157	0.5284	1.71	-2.598	1.8	2.068	-0.6876	3.87
V_{ij}^t	1.3457	-2.2915	0.483	-2.7	-1.67	1.4988	-1.66	0.1596

Step 4: Obtain the new job sequence based on SPV rule, and then the sequence is: 4, 7, 2, 3, 5, 6, 1, 8.

Step 5: Allocate the jobs in machines according to the procedure illustrated in table 2. As per the newly obtained job sequence

Step 6: Calculate system unbalance and throughput

Step 7: List out assigned jobs and unassigned jobs.

Step 8: Terminate when the maximum number of iterations is reached and find the best sequence for which least system unbalance and maximum throughput is found.

Table 5. Generation of new sequences based on particle positions for 20 iterations

Iteration	New Sequence	System Unbalance	Throughput	Objective function	Unassigned jobs
1	4,6,7,3,1,2,5,8	141	41	1.439	1,2,5,8
2	**4,7,2,3,5,6,1,8**	**14**	**48**	**1.5927**	**2,6,8**
3	4,7,2,5,3,6,1,8	14	48	1.5927	2,6,8
4	2,5,4,7,8,1,3,6	0	36	1.45	8,3,6
5	8,5,1,2,6,3,7,4	185	40	1.403	2,3,7,4
6	8,1,6,3,5,2,7,4	127	44	1.483	5,2,7,4
7	1,6,8,3,5,7,2,4	127	44	1.483	5,7,2,4
8	6,3,4,1,7,8,2,5	18	46	1.565	7,8,5
9	4,7,2,3,6,5,1,8	14	48	1.592	2,6,8
10	4,7,2,5,3,6,1,8	14	48	1.592	2,6,8
11	4,2,5,7,8,3,1,6	0	36	1.45	8,3,1,6
12	8,5,2,1,3,6,7,4	144	43	1.4625	2,6,7,4
13	8,1,6,5,3,2,7,4	185	40	1.403	3,2,7,4
14	8,1,6,3,5,2,7,4	127	44	1.483	5,2,7,4
15	6,1,3,8,7,5,4,2	127	44	1.483	7,5,4,2
16	4,7,6,3,1,2,5,8	14	48	1.5927	6,2,8
17	4,7,2,3,5,6,1,8	14	48	1.5927	2,6,8
18	4,7,2,5,3,6,1,8	14	48	1.5927	2,6,8
19	4,2,5,7,8,3,1,6	0	36	1.45	8,3,1,6
20	5,2,8,1,7,3,4,6	130	31	1.319	1,7,3,4,6

6.3 Results

System Unbalance: **14**

Throughput: **48**

Assigned jobs are...

4 7 3 5 1

Unassigned jobs are...

2 6 8

Best Sequence...

4 7 2 3 5 6 1 8

Table 6: Comparison of Results

Total Number of jobs	M.K.Tiwari et al.[4]		N.K.Vidyarthi et al.[2]		M. K. Tiwari & N.K. Vidyarthi,et al.[10]		Akhilesh et al.[12]		Proposed PSO heuristic	
8	SU	TH	SU	TH	SU	TH	SU	TH	SU	TH
	76	42	122	42	14	48	0	36	**14**	**48**
Objective function	1.4854		1.4614		1.5927		1.4500		**1.5927**	

6.4 Discussions

The proposed PSO algorithm for the FMS loading problem is coded in C++ programming language. Initially, for eight jobs (particles) the position (search space 0 to 4) and velocity (-4 to 4) values are randomly arranged (adopted from Sandhyarani Biswas [8]). Based on the position values, applying SPV rule the job sequence is obtained. For the next iteration the values for position and velocity will be updated by using equations 4, 5 &6 and new job sequence will be obtained. At every iteration, the system unbalance, throughput and objective function value is generated and compared with manual calculations.

The algorithm is shown as a flow chart in Fig. 2. Since different methods have been used by researchers for the calculation of system unbalance, the machine-loading problem of an FMS has been solved by considering two cases to examine the performance of the PSO algorithm. Computational results are summarized in Tables 6, using standard PSO.

It is evident from Table 6 that the PSO algorithm is capable of either outperforming existing methods or at least producing an equal solution, as indicated in Table 6. For example, the PSO algorithm yields a system unbalance of 14 and a throughput of 48, whereas Vidyarthi [2] arrived at a system unbalance of 122 and a throughput of 42 for the same problem.

From the literature [13], it was observed that, for the test problem selected, around 50 iterations were required for obtaining a global/near-optimal solution, but the PSO algorithm could produce best solutions within 20 iterations and, hence, it requires less computational effort.

CONCLUSIONS AND FUTURE SCOPE

The main contribution of this work is to develop an efficient and reliable evolutionary based approach to solve the flexible manufacturing system (FMS) loading problem. The key advantage of PSO is its computational efficiency and a smaller number of parameters are required to be adjusted in order to obtain the optimum result compared to related techniques. A comparative study has been carried out for the same problem with similar objective functions and constraints, and the computational experience manifests that the proposed meta-heuristic approach based on PSO outperforms or at least producing an equal solution of the existing methodologies as far as solution quality is concerned with reasonable computational efforts. The objective of this study is not only to minimize system unbalance, but also simultaneously increase the throughput by using a meta-heuristic based on PSO.

From the comparative study (table 6), it has been observed that the proposed algorithm offers better results.

In future, the study can be extended to solve the loading problem using the same meta-heuristic approach in a multiple-objective framework, i.e., combined objective of the minimization of the system unbalance and the maximization of throughput.

By considering more realistic variables and constraints, such as the availability of pallets, jigs, fixtures, automatic guided vehicles (AGVs), etc., in addition to tool slots and machining time may be included to solve loading problems.

REFERENCES

1. Kennedy J, Eberhart RC (1995) “Particle swarm optimization”. In: Proceedings of the 1995 IEEE International Conference on Neural Networks (ICNN’95), Perth, Australia, November/December 1995, vol 4, pp 1942–1948

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

3. K.Vidyarthi and M.K.Tiwari. “Machine loading problem of FMS: a fuzzy-based heuristic approach”, International Journal of Production Research, 2001, 39(5), 953-979.

4. M.K.Tiwari, B.Hazarika “Heuristic solution approach to the machine loading problem of an FMS and its Petrinet model”, International Journal of Production Research,1997, 35(8), 2269-2284.

5. Swarnkar R, Tiwari MK (2004) Modeling machine loading problem of FMSs and its solution methodology using a hybrid tabu search and simulated annealing-based heuristic approach. Robot Comput-Int Manuf 20(3):199–209

6. Nagarjuna N, Mahesh O, Rajagopal K (2006) A heuristic based on multi-stage programming approach for machine-loading problem in a flexible manufacturing system. Robot Comput-Int Manuf 22:342–352

7. S. G. Ponnambalam, and Low Seng Kiat (2008) “Solving Machine Loading Problem in Flexible Manufacturing Systems Using Particle Swarm Optimization” World Academy of Science, Engineering and Technology 39

8. Sandhyarani Biswas, S S Mahapatra, “Machine loading in Flexible manufacturing System: A swarm optimization approach” Eighth Int. Conference on Opers. & Quant. Management, 2007, NITR, 621-628

9. Kathryn E. Stecke, (1986), “A hierarchical approach to solving machine grouping and loading problems of FMS’S”, European journal of operational research, 24, 369-378

10. M. K. Tiwari & N.K. Vidyarthi (2000): Solving machine loading problems in a flexible manufacturing system using a genetic algorithm based heuristic approach, International Journal of Production Research, 38:14, 3357-3384

11. Paul S. Andrews, An Investigation into Mutation Operators for Particle Swarm Optimization

12. Akhilesh kumar, prakash, MK Tiwari(2006), "Solving machine loading problem of a FMS with constraint based GA" , European Journal of Operational Research 175 (2006) 1043–1069

13. Sandhyarani Biswas, S S Mahapatra, "Modified particle swarm optimization for solving machine-loading problems in flexible manufacturing systems". Int J Adv Manuf Technol (2008) 39:931–942