WORK FLOW BALANCING IN PARALLEL MACHINES USING PARTICLE SWARMING ALGORITHM

A PROJECT REPORT P-2334

Submitted By

R. Dhanya - 71206409003

In partial fulfillment for the award of the degree of

MASTER OF ENGINEERING

in

INDUSTRIAL ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE – 641 006

ANNA UNIVERSITY :: CHENNAI 600 025

JUNE – 2008

ANNA UNIVERSITY :: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report entitled "**Work Flow Balancing in Parallel Machines Using Particle Swarming Algorithm**" is the bonafide work of

Ms.R.Dhanya - Register No. 71206409003

who carried out the project work under my supervision.

Signature of the HOD

Signature of the Supervisor

Internal Examiner

External Examiner

Department of Mechanical Engineering

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE – 641 006

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE , TAMILNADU

DEPARTMENT OF MECHANICAL ENGINEERING & KCT-TIFAC CORE

ADVANCES IN MECHANICAL SCIENCES

CERTIFICATE

s is to certify that Mr/Ms/Mrs R. DHANYA

articipated and presented a paper titled Work flow balancing in

rallel machines using particles swarm algorithm

in the 2nd National Conference on "ADVANCES

CHANICAL SCIENCES" during 27- 28, March 2008.

LANISAMY
DINATOR

Dr.C.SIVANANDAN
CONVENOR & DEAN

Dr.JOSEPH V.THANIKAL
PRINCIPAL

ANNARI AMMAN
STITUTE OF TECHNOLOGY
Stay Ahead

optest 2008

TÜV CERT
EN ISO 9001
Certificate : 04100021309
RWTÜV

Third National Conference on

Optimization Techniques in Engineering Sciences and Technologies

Certificate

This is to certify that Mr./Ms./Mrs. K. DHANYA

has participated / presented a paper entitled Work Flow Balancing in Parallel Machines using Particles Swarming Algorithm with Precedence Constraints

in the Third National Conference on "Optimization Techniques in Engineering Sciences and Technologies" (OPTEST -2008), during 27-28 March 2008, organized by the Department of Mechanical Engineering, Annari Amman Institute of Technology, Sathyamangalam.

G SASIKUMAR
Organizing Secretary

Dr A M K PODUVAL
Convener

Dr A SHANMUGAM
Chairman

ABSTRACT

Scheduling allocates workloads to specific work centers and determining the sequence in which the operation are to be performed. Parallel machine Scheduling involves Scheduling of independent jobs on parallel machines with the objective of minimizing the maximum flow time. Work flow is used to distribute the workloads among the machines to reduce the idle time. The machine with maximum workload is the bottleneck present in a system .The presence of bottleneck prevents the system from achieving high throughput. Thus the work flow balancing is used to remove the bottleneck available in the system. In this work an attempt is made to reduce the bottleneck using PSO. And is compared with longest processing time shortest processing time and random. The PSO Provides the better performance and the computer program has been coded for validation in standard manufacturing environment on an IBM/PC compatible system in the 'C' language. The performance of PSO and Relative percentage of Imbalance (RPI) is used as a parameter to analyze the performance of particles swarming algorithm. The particle swarm optimization algorithm was introduced to study social and cognitive behavior, but it has been largely applied as a problem-solving technique in engineering and computer science. There are two main types of information available to each individual of the population. The first is their own past experiences, and the second is the knowledge about how individuals around them have performed. The updating of velocity and adding with the current value give the minimum value of imbalance in work flow.

ஆய்வு சுருக்கம்

வேலையை பகிரிந்து கொடுத்தல் என்பது ஒரு உற்பத்தி நிறுவனத்தின் தடைகளை குறைக்க உதவுகிறது. துகள் பிழைத் தொகுதி உகப்பாக்கம் என்கிற முறை இணையான இயந்திரங்களில் உள்ள தடைகளை குறைக்க உதவுகிறது. இதன் முடிவுகள் மற்ற முறைகளின் முடிவுகளுடன் அதாவது ரேண்டம், சிறும செயலாக்கும் நேரம், மற்றும் நீண்ட செயலாக்கும் நேரம் முறைகளுடன் ஒப்பிடப்படுகிறது. இங்கு சார்பு நிறையின்மையின் சதவிகிதம் என்பது குறிப்பிட்ட குறைகளின் முடிவுகளை அலச உதவுகிறது. இதில் தகள் பிழைத் தொகுதி உகப்பாக்கம் என்ற முறை நல்ல முடிவுகளை தருகிறது.

மேலும் தகள் பிைத் தொகுதி உகப்பாக்கம் என்பது முதன்மை கட்டுத்திட்டம் உள்ள வேலையை பகிர்ந்து கொடுத்தல் பிரச்சினையும் தீர்க்க உதவுகிறது. இந்த முறையில் அனைத்து முக்கிய வேலைகளை முதலிலும் பிறகு மற்ற வேலைகளும் செய்யப்படுகின்றன. இங்கும் சார்பு நிறையின்மையின் சதவிகிதம் என்பது அனைத்து நெறிமுறைகளின் முடிவுகளை ஒப்பிடப் பயன்படுகிறது. இங்கும் தகள் பிழைத் தொகுதி உகப்பாக்கம் முறையே நல்ல முடிவுகளை தருகின்றன. இந்த துகள் பிழைத் தொகுதி உகப்பாக்கம் முறை மற்றும் அனைத்து முறைகளும், ‘c’ என்ற மொழியை பயன்படுத்தி ப்ரோகிராம் எழுதப்படுகிறது.

ACKNOWLEDGEMENT

All that have started well will end well and at this instant of time I would like to thank the people who were directly and indirectly instrumental in initiating me to do this work.

I register my hearty appreciation to **Mr. A. Rajesh**, my thesis advisor. I thank for his support, encouragement and ideas. I thank him for the countless hours he has spent with me, discussing everything from research to academic choices.

I take this opportunity to thank **Dr. C. Sivanandan**, Ph.D., Head of the Department, Mechanical Engineering, for the continuous motivation. I would like to thank **Dr. Joseph V. Thanikal**, Ph.D., Principal for providing the necessary facilities to complete my thesis.

Words are inadequate in offering my thanks to **Dr.R.Saravanan**, Head of the Department of Mechatronis Engineering, Kumaraguru College of Technology, Coimbatore for providing valuable guidance and innovative ideas in completing this project.

My sincere thanks to **Dr. T. Kannan**, Associate Professor and **other faculty members** of the Department for their encouragement and cooperation in carrying out my project work. My sincere thanks also to all friends for their help in all the matters during my project work.

Last but not least, I express my heartfelt thanks to the Almighty for the blessings. Without His permission and blessings it would not have been possible to complete this project work.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS/ ABBREVATIONS

Symbols / Abbreviations	Explanation	Page No.
P_{ij}	Processing Time	4
r_j	Ready Time	5
d_j	Due Date	5
w_j	Weight	5
C_{ij}	Completion Time	5
F_j	Flow time	5
L_j	Late time	5
T_j	Tardiness	5
F_{mean}	Mean Flow time	5
T_{mean}	Mean Tardiness	5
F_{max}	Maximum Flow Time	5
T_{max}	Maximum Tardiness	5
N_T	Number of Tardiness jobs	5
m	Number of Machines	16
n	Number of Jobs	16
W_k	Work Load of a Machines	16
W_{max}	Maximum Work Load	16
SPT	Shortest processing time	18
LPT	Longest processing time	18
GA	Genetic Algorithm	19
r	Random Number	34
S	List of all assembly operations	34
S_1	List of preceding operations	34
S_2	List of succeeding operations	34
X_1	Job position of pervious iteration	24
X_2	Job position of previous iteration	34
X_1^1	Job position of new iteration	34
RANDOM	Random	34
RPI	Relative percentage Of imbalance	35

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

In our day-to-day life time is an important constraint. Every human being in this world plans or schedules their activities according to the time given to them. Planning activities and working according to it leads to a bright future. Engineering is a wide area where planning activities solve many problems such as more time consumption, delivery of products within due date, high penalty costs, scale of production etc. Planning activities and scheduling are considered as the tracks of a train. Most industrial application deals with the scheduling problems like improper workflow balance, high makespan, increased flow time and lateness. Many heuristic or traditional techniques are available to solve the above said problems. The heuristic or traditional approaches consume more time when implemented for large number of unscheduled work or tasks. To rectify this problem there are many non-traditional techniques, which in combination with Information Technology produces optimum results. Moreover in heuristic procedures the objective function will be single i.e., makespan minimization or reduction in penalty cost or balancing workflow or reduction in idle time etc. particles Swarming is a technique, which gives best results for multilevel objective function.

1.2 SCHEDULING

Scheduling is a decision-making process that plays an important role in most manufacturing and service industries. It is used in procurement and production, in transportation and distribution, and in information processing and communication. The scheduling function in a company uses mathematical techniques or heuristic methods to allocate limited resources to the processing of tasks. A proper allocation of resources enables the company to optimize its objectives and achieve its goal. Resources may be machines in a workshop, runways at an airport, crews at a construction sites, or takeoffs and landings at an airport, stages in a construction

Each task may have a priority level, an earliest possible starting time to complete all tasks or minimizing the number of tasks completed after their committed due dates.

1.2.1 Example

A System Installation Project:

Consider the procurement, installation, and testing of a large computer system. The project involves a number of distinct tasks, including evaluation and selection of hardware, software development, recruitment and training of personnel, system testing, and system debugging. A precedence relationship structure exists among all these tasks: some can be done in parallel, whereas others cannot start until certain prior tasks are completed. The goal is to complete the entire project in minimum time.

Scheduling not only provides a coherent process to manage the project but also provides a good estimate for its completion time, reveals which tasks are critical and determines the actual duration of the entire project.

1 .3 SCHEDULING MODELS

The study of scheduling models contain the elements and relationships that frequently arise in scheduling problems, and they also suggest how feasible solutions are systematically be constructed. Format models are available to aid decision-making in a variety of scheduling problems. For example, one of the simplest and most widely used models is the Gantt chart, which is a graphical representation of scheduling relationships. In its basic form the Gantt chart is a graph of resource allocation over time. Generally, specific resources are shown in vertical axis, and a time scale is shown in horizontal axis, as in Figure 1.1.

In this concise format, analysis of the graphical relationships can yield inferences about the behavior of the given schedule, while manipulation of the graphical elements can yield comparative information about alternative scheduling decisions.

In this way, the Gantt chart serves as a focus for implementing the system approach in scheduling. The scheduling models are inherent in the systems approach and provide a direct decision-making method when the systems approach is utilized. When a

the scheduling function. It is the theory of scheduling that has developed this type of model, providing a useful framework for performing the scheduling function effectively.

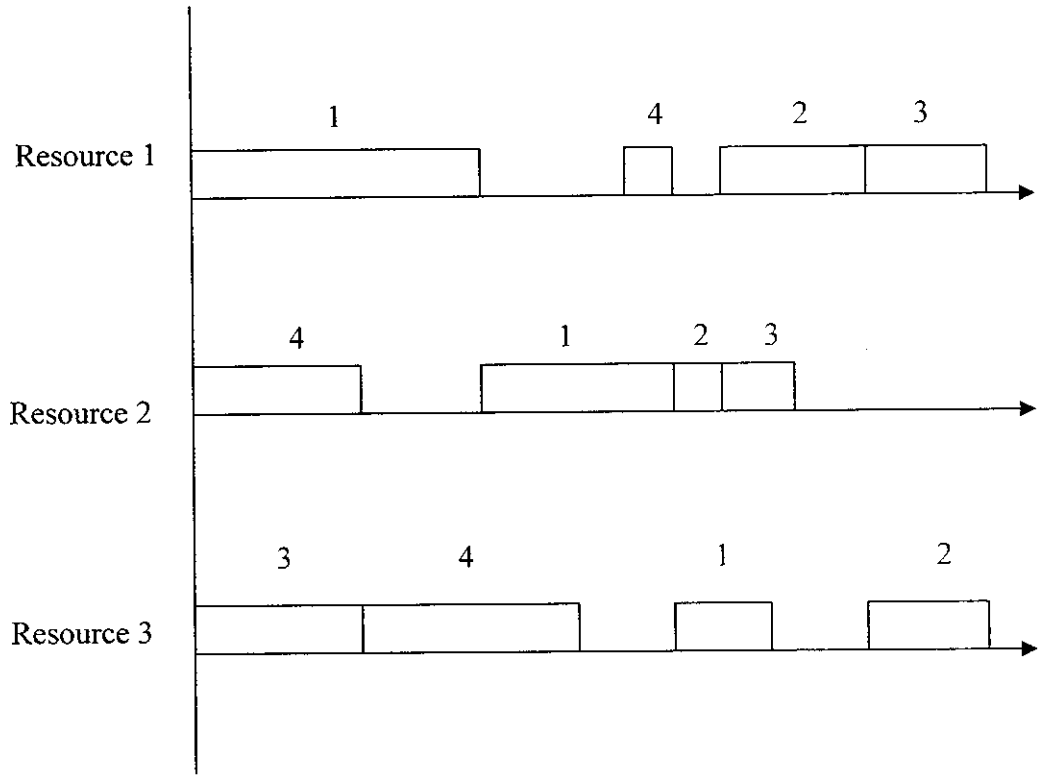


FIGURE 1.1 GANTT CHART FOR SCHEDULING 4 JOBS ON 3 RESOURCES.

In Figure 1.1, the resources used are called machines; the basic task schedules are called jobs. Sometimes, jobs may consist of some elementary tasks that are interrelated by precedence restrictions; such elementary tasks are referred to as operations.

The pure sequencing problem is a specialized scheduling problem in which an ordering of the jobs completely determines a schedule.

Moreover, the simplest pure sequencing problem is one in which there is a single resource, or machine. The basic single-machine problem is characterized by these conditions.

1. A set of 'n' independent, single-operation jobs is available for processing at time

2. Setup times for the jobs are independent of job sequence and can be included in processing times.
3. Job descriptors are known in advance.
4. One machine is continuously available and is never kept idle while work is waiting.
5. Once processing begins on a job, it is processed to complete without interruption.

Under these conditions, there is one-to-one correspondence between a sequence of the 'n' jobs, and a permutation of the job indices 1, 2, 3, n. The total number of distinct solutions to the basic single-machine problem is therefore n! which is the number of different permutations of 'n' elements.

1.4 TERMINOLOGIES USED

Three basic pieces of information that help to describe jobs in the deterministic single-machine case are:

Processing time (p_{ij}): The processing time 'p_{ij}' represents the time job 'j' has to spend on machine 'i'. The subscript 'i' is omitted if the processing time of job 'j' does not depend on the machine or if it needs processing on only one machine.

If a number of identical jobs need a processing time 'p_j' on one machine, then these jobs are referred as items of type 'j'. The machine's production rate of type 'j' items is then denoted by $q_j = 1/p_j$.

Ready time (r_j): The point in time at which job 'j' is available for processing. That is, it is the time the job arrives at the system, i.e., the earliest time at which job 'j' can start its processing. This is also called as release time.

Due date (d_j): The point in time at which the processing of job 'j' is due to be completed. That is, the due date 'd_j' of job 'j' represents the committed shipping or completion date (the date the job is promised to the customer). Completion of a job after its due date is allowed, but a penalty is then incurred. A *deadline* is a due date absolutely must be sent.

Weight (w_j): The weight 'w_j' of job 'j' is a priority factor, denoting the importance of job 'j' relative to other jobs in the system. It may represent the cost of keeping the job in the system. The weight can be a holding or inventory cost, or it can be the amount already added to the job.

they do not depend on the schedule.

but that do depend on the schedule are

e 'C_{ij}' is the time job 'j' is completed on

en 'C_j' refers to the time job 'j' leaves the

spends in the system.

which the completion time of job 'j' exceeds its

os 'j' if it fails to meet its due date, or zero

Schedules are ... by aggregate quantities that involve information about all jobs, resulting in one-dimensional *performance measures*. Measures of schedule performance are usually functions of the set of completion times in a schedule. For example, suppose that 'n' jobs are to be scheduled. Aggregate performance measures that might be defined include the following:

Mean flow time: $F_{mean} = 1/n \Sigma F_j$, $j = 1$ to n

Mean tardiness: $T_{mean} = 1/n \Sigma T_j$, $j = 1$ to n

Maximum flow time: $F_{max} = \max \{F_j\}$, $j = 1$ to n

Maximum tardiness: $T_{max} = \max \{T_j\}$, $j = 1$ to n

Number of tardy jobs: $N_T = \Sigma \delta (T_j)$, where $\delta (x) = 1$ if $x > 0$;
and $\delta (x) = 0$ otherwise.

1.5 MACHINE CONFIGURATIONS

Scheduling models are often characterized by the machine configuration, the processing restrictions and constraints, and the objectives. There are many important machine configurations. This section describes the most basic ones.

1.5.1 Single-Machine Models

Numerous production systems give rise to single-machine models. For instance, if a single bottleneck occurs in a multiple-machine environment, then the job sequence at the bottleneck typically determines the performance of the entire system. In this case all upstream and downstream operations are scheduled after the bottle-neck is scheduled. This approach implies that the original problem first has to be reduced to a single-machine scheduling problem. Single-machine models are also important in decomposition approaches, where scheduling problems in complicated environments are broken down into a number of smaller, single-machine scheduling problems.

1.5.2 Parallel-Machine Models

Jobs-to-machine assignment is based on several factors, such as precedence, availability, production rate, machine suitability (processing plan) and workload balancing with reference to the objective function.

Researchers have addressed parallel machine scheduling problems with many variant approaches. More objective functions are considered to obtain an efficient schedule for scheduling problems.

The number of machines is denoted by m and number of jobs by 'n'. Each job is indicated by its processing time p_j where j = 1, 2, 3… n.

1.5.3 Flow Shop Models

In many manufacturing or assembly environments, jobs have to undergo multiple operations on a number of different machines.

If the routes of all jobs are identical, that is, all jobs visit the same machines in the same order; the environment is referred to as a flow shop. The machines are set up in

since jobs may be resequenced between machines. However, if a material-handling system transports the jobs from one machine to the next, then the same job sequence is maintained throughout the system.

In some flow shops, if a job does not need processing at a particular machine, it may bypass that machine, and go ahead of the jobs being processed or waiting for processing there. Other flow shops, however, do not allow bypass.

A generalization of the flow shop is the flexible flow shop, which consists of a number of stages in series with a number of machines in parallel at each stage. Jobs are processed at each stage on any one of the parallel machines.

1.5.4 Job Shop Models

In multi operation shops jobs often have different routes. This environment is referred to as a job shop, which is a generalization of a flow shop. (A flow shop is a job shop in which each and every job has the same route).

The simplest job shop models assume that a job may be processed on a particular machine at most once on its route through the system. In others a job may visit a given machine several times on its route through the system.

These shops are said to be subject to recirculation, which significantly increases the complexity of the model.

A generalization of the job shop is the flexible job shop with work centers that have multiple machines in parallel. From a combinatorial point of view, the flexible job shop with recirculation is the most complex machine environment, and is very common in the semiconductor industry.

1.6 PARALLEL-MACHINE MODELS

A bank of machines in parallel is a generalization of the single-machine model. Many production environments consist of a number of stages or work centers, each with a number of machines in parallel.

The machines at a work center may be identical so that whenever a job arrives, it may be processed on any one of the available machines. Parallel-machine models are important for the same reason that single-machine models are important: if one particular work center is a bottleneck, then the schedule at that work center will determine the performance of the entire system. That bottleneck can be modeled as a bank of parallel machines and analyzed on its own.

At times the machines in parallel may not be identical. Some machines may be older than others and may operate at a lower speed, or one machine may be better maintained and able to do higher-quality work than another. In this case some jobs may be processed only on any of the 'm' machines in parallel, while others may be processed only on a specific subset of the 'm' machines. When the machines are people, then the processing time of an operation may depend on the job as well as on the operator. One operator may excel in one type of job, whereas another operator may specialize in another type of jobs.

1.7 BASIC DISPATCHING RULES

A dispatching rule is a rule that prioritizes all the jobs that are waiting for processing on a machine. The prioritization scheme may take into account the jobs' attributes and the machines' attributes, as well as the current time. Whenever a machine has been freed, a dispatching rule inspects the waiting jobs, and selects the job with the highest priority.

Dispatching rules can be classified in various ways. For example, a distinction can be made between static and dynamic rules. Static rules are not time-dependent. They are just a function of the job data, the machine data, or both. Dynamic rules, on the other hand, are time-dependent.

A second way of classifying dispatching rules is according to the information they are based on. A local rule uses only information pertaining to either the queue where the job is waiting or machine where the job is queued. A global rule may use information pertaining to other machines, such as the processing time of the job on the next machine on its route or the current queue length at that machine. Some of the rules are

SIRO (Service in random order) rule: According to this rule, whenever a machine is freed, the next job is selected at random from those waiting for processing.

ERD (Earliest release date first) rule: This rule is equivalent to the well-known first-come-first-served rule.

EDD (Earliest due date first) rule: Whenever a machine is freed, the job with the earliest due date is selected to be processed next.

MS (Minimum slack first) rule: This rule is a variation of the EDD rule. If a machine is freed at time't', the remaining slack of each job at that time, defined as $\max(d_j - p_j - t, 0)$, is computed. The job with the minimum slack is scheduled next.

WSPT (Weighted shortest processing time first) rule: Whenever a machine is freed, the job with the highest ratio of weight over processing time is scheduled next.

LPT (Longest processing time first) rule: This rule orders the jobs in decreasing order of their processing times. When there are machines in parallel, this rule tends to balance the workload over the machines.

SST (Shortest setup time first) rule: Whenever a machine is freed, this rule selects for processing the job with the shortest setup time.

LFJ (Least flexible job first) rule: Whenever a machine is freed, the job that can be processed on the smallest number of other machines is selected, that is, the job with the fewest processing alternatives.

CP (Critical path) rule: The CP is used when jobs are subject to precedence constraints. It selects as the next job the one at the head of the longest string of processing times in the precedence constraints graph.

LNS (Largest number of successors) rule: This rule may also be used when the jobs are subject to precedence constraints. It selects as the next job the one that has the largest number of jobs following it.

SQNO (Shortest queue at the next operation) rule: Whenever a machine is freed, the job with the shortest queue at the next machine on its route is selected for processing.

1.8 NON-TRADITIONAL TECHNIQUES

In the development of scheduling models more general than single-machine models, the parallel machine scheduling represents the most direct extension of multiple-resource situations. In general, solutions to parallel machine model problem appear to require either combinatorial techniques that take advantage of special problem structures or else clever heuristic methods. The effectiveness of any solution method can be evaluated only on relative terms, on the basis of the availability of computational resources and the acceptability of sub optimal results. From the above statements we can say or conclude that parallel machine model requires some computational techniques, which should be capable of giving optimal results.

Hence, we go for Particles Swarming Algorithm, it is one of the non-traditional techniques, which gives optimal results for scheduling problems, is discussed in the following sections.

CHAPTER 2

LITERATURE REVIEW

CHAPTER 2

LITERATURE REVIEW

LITERATURE REVIEW

Following are the overview of the relevant work done earlier related to the identified problem and the methodology to be adopted to solve the chosen problem for this work. This section describes the literature reviewed from various research papers published in journals, proceedings of various conferences and books.

Arunachalam.V.P, Rajakumar.S, and Selladurai.V. (2004), In this paper many manufacturing environments, multiple processing stations are used in parallel to obtain adequate capacity. In parallel machine scheduling there are 'm' machines to which 'n' jobs are to be assigned based on different strategies. The procedure is based on workload balancing among the machines. A lowest workload of a machine is selected for assignment of a new job from the list of unfinished jobs. Different priority strategies are followed for the selection of jobs. Three different strategies are considered, namely random (RANDOM), shortest processing time (SPT), and longest processing time (LPT) for the selection of jobs for workflow balancing. The relative percentage imbalance (RPI) is used to evaluate the performance of the three. The LPT provides the better results.

Arunachalam.V.P, Rajakumar.S and Selladurai.V. (2005). In this journal the assembly-planning of a textile machine in a shop floor which can help researchers and practitioners. The assembly planning of a textile machine involves the allocation of operations to cross-trained operators. Workflow is defined as the workloads assigned to the operators. Operators with smaller workloads are selected to be assigned new operations from the list of unscheduled operations. Three different scheduling strategies – random (R), shortest processing time (S) and longest processing time (L) – are adopted for the selection of operations to be assigned to operators. Different combinations of these strategies are considered

percentage of imbalance is adopted for evaluating the performance of these heuristics. The RL, SL, LL produced well balanced workload schedules with lesser RPI values for all operators other than heuristics.

Arunachalam.V.P, Rajakumar.S and Selladurai.V. (2006). Here the genetic algorithm (GA) is used to solve the parallel machine scheduling problem of the manufacturing system with the objective of workflow balancing.

The performance of GA is compared with three workflow strategies namely random (RANDOM), shortest processing time (SPT), and longest processing time (LPT).

The relative percentage of imbalance (RPI) is used to evaluate the performance of these heuristics. The GA provides the better performance for the combination of various job sizes and machines .

Arunachalam.V.P, Rajakumar.S and Selladurai.V (2006). In this paper the methodology is based on genetic algorithm (GA) to solve the parallel machine scheduling problems with precedence constraints. Workflow balancing helps to remove bottle-necks present in a shop floor yielding faster movements of components or jobs. Multiple machines are used in parallel for processing the jobs to meet the demand. In parallel machine scheduling with precedence constraints, there is 'm' machines to which 'n' jobs are assigned using suitable scheduling algorithms. Workflow of a machine is the sum of processing time of all jobs assigned. All the preceding jobs are allocated first to satisfy the constraints. GA is developed to solve parallel machine scheduling problems with precedence constraints based on the objective of workflow balancing. The RPI in workloads among the parallel machines is used to evaluate the performance of the GA developed. The proposed GA produces lesser RPI values against the RANDOM heuristics algorithm for a wider range of jobs and machines.

Asokan.P, Sachithanandam.M and Saravanan.R. (2001). This paper describes various optimization procedures for solving the CNC turning problem to find the optimum operating parameters such as cutting speed and feed rate. Total

as cutting force, power, tool-chip interface, temperature and surface roughness of the product. Conventional optimization techniques and non-conventional optimization techniques are employed in this work. Results are compared and their performances are analyzed .

Anand Kumar.P and Venketesan.R (2005).

This paper clearly explains the extrusion dies can be designed using non-traditional optimization techniques. Upper bond solution is used for formulating the mathematical equation for extrusion of a circular rod through a conical converging die. The design problem is formulated as a constrained optimization problem.

Simulated annealing algorithm is applied to minimize the extrusion force. Extrusion load and power also determined and result are shown graphically .

Emin Aydin.M and Terence C. Fogarty (2004). In this paper, the simulated annealing approach is applied to two-bicriteria scheduling problems on a single machine. The first problem is the strongly NP-hard problem of minimizing total flow time and maximum earliness. The second one is the NP-hard problem of minimizing total flow time and number of tardy jobs. Different neighborhood structures as well as other parameters of the simulated annealing approach are experimented to improve its performance. The computational experiments show that the developed approach yields solutions that are very close to lower bounds and hence very close to the optimal solutions of their corresponding problems for the minimization of total flow time and maximum earliness. For the minimization of total flow time and number tardy, the experiments show that the simulated annealing approach yields results that are superior to randomly generated schedules [8].

Esin Onbasoglu and Linet Ozdamar (2001). In this paper Global optimization involves the difficult task of the identification of global extremities of mathematical function. Such problems are often encountered in practice in various fields, such as molecular biology, physics, and industrial chemistry. In this work

we develop five different simulated annealing algorithms and compare them in various solution approaches in global optimization.

Erin Ayden's and Terence C. Fogarty (2004). In this paper, a parallel implementation of the modular simulated annealing algorithm for classical job-shop scheduling is presented. The implementation is for a multi agent system running on the distributed resource machine, which is a novel, scalable, distributed virtual machine based on Java technology. The problems tackled are well known, difficult benchmarks, widely used to measure the efficiency of metaheuristics with respect to both the quality of the solutions and the central processing unit time. The empirical results obtained show that the method proposed is successful in comparison with a sequential version of modular simulated annealing algorithm .

Mohanasundaram.K.M , Suresh.R.K (2004). Here the real world scheduling problems are multi objective in nature. The objective of such problems is to find a set of Pareto solutions. In JSSP literature or the best solutions to the benchmark problems are available for the makespan performance measure and for the other performance measure. The major contribution of this paper is to present computational results of flow time performance for the benchmark JSSPs .

Michael w.trosset (2001). In this paper the Simulated annealing was marketed as a global optimization methodology that mimics the physical annealing process by which molten substances cool to crystalline lattices of minimal energy .

Shiu-hong choi and james siu-hung lee (2004). In this paper the job sequencing is an important stage in any hierarchical production control model, especially when a real time dispatching rule is not employed. The problems become complicated, when constraints such as different parts requiring different operation processes at different machines. and with different production priorities. This paper first describes a mathematical programming model developed for small flexible manufacturing systems.,(FMS) make span minimization sequencing

each part type require different number of operation processes at different machines.

Nick vagenas and Tony nuziale (2001). In this paper the test mining equipment reliability assessment models based on genetic algorithms. Genetic algorithms are powerful and broadly. applicable stochastic search technique based on the principles of natural selection, heredity and genetics. The reason for selecting genetic algorithm is the fact that the reliability of mining equipment changes overtime due to its dependence upon several covariates. These factors combine to create a complex impact on an equipment reliability function

CHAPTER 3

PROBLEM

CHAPTER 3

PROBLEM DEFINITION

3.1 PROBLEM DEFINITION

The parallel machine-scheduling problem deals with assigning of 'n' jobs on 'm' machines. Each job has only one operation. The jobs may be assigned on any one of the machines based on an objective function. Each job is indicated with the processing time 'p_j' where j = 1, 2, 3 ... n. a machine can process only one job at once. The objective of the problem is to minimize the maximum workload on parallel machines.

Minimize

$$W_{max} \qquad \text{... (3.1)}$$

Subject to

$$\sum_{j=1}^{n} X_{jk} = 1 \qquad k = 1, 2, 3 \ldots m \qquad \text{... (3.2)}$$

$$W_k \leq W_{max} \quad k = 1, 2, 3 \ldots m \qquad \text{... (3.3)}$$

$$X_{jk} \in \{0, 1\} \; k = 1, 2, 3 \ldots m \qquad \text{... (3.4)}$$

where

m	Number of machines
n	Number of jobs
W_k	Workload of a machine
W_{max}	Maximum workload
X_{jk}	1 if job 'j' is assigned to machine 'k' and 0 otherwise.

The equation (3.1) is used to minimize the maximum workload on parallel machines. Workload of a machine is the sum of processing time of jobs assigned to it. This objective is used to obtain the optimum utilization of machines. This objective also ensures the earliest completion of all jobs.

The constraint equation (3.2) indicates that each job is scheduled only once on a machine for processing. The constraint equation (3.3) defines the maximum workload. The constraint equation (3.4) indicates the decision variable 'X' is binary in overall domain.

3.2 OBJECTIVE

To reduce the imbalance of jobs among the machines using Simulated Annealing Algorithm. Workflow balancing among parallel machine is essential to maximize the optimum uses of the resources. It helps to remove the bottleneck present in a manufacturing system to improve the throughput. Workflow refers to the workloads of the machines. Workload of a machine is the sum of processing time of all jobs assigned to it. Idle time is not considered for calculating the workflow. A machine with the lowest workload is selected for assigning a new job from lists of jobs. Many heuristics are available to prepare the lists of jobs.

A Particles Swarming algorithm (PSA) is used to prepare the list of jobs to be assigned. The performance of PSO is compared with three workflow balancing strategies are random (RANDOM), Shortest Processing Time (SPT), and Longest Processing Time (LPT). The jobs are assigned to parallel machines to minimize the imbalance in workloads. The minimum imbalance in workloads ensures balanced workflow among the machines. The Relative Percentage of Imbalance (RPI) is adopted among the parallel machines for evaluating the performance of the heuristics to a standard manufacturing environment. A computer program will be coded on an IBM/PC compatible system in the 'C' language for experimentation to a standard manufacturing system environment in operation.

CHAPTER 4

METHODOLOGY

CHAPTER 4

METHODOLOGY

4.1 STEPS TO BE FOLLOWED

The parallel machine scheduling problem consists of a set of 'm' machines M_i, i=1, 2, 3...m and a set of 'n' jobs J_i, j=1, 2, 3...n. Job J_i requires processing time P_j and it can be processed on any one of the machines. A machine can process only one job at a time. The machine M_i with less cumulative workload will be selected for assigning a job from the list. The workload W_i of i^{th} machine (M_i) is calculated by summing up of processing time of all jobs assigned to it. The list of jobs is prepared using RANDOM, SPT, LPT, and PSO. The workload allocation algorithm is described below:

Step 1: Find out the cumulative workload W_i of each machine, i.e., sum of all processing time of jobs assigned to i^{th} machine.

Step 2: Select a machine M_i with smaller cumulative workload W_i among 'm' machines available.

Step 3: Assign job J_i from the list of jobs prepared using RANDOM, SPT, LPT, and PSO to the machine M_i.

Step 4: Repeat Steps 1 to 3 until all the jobs from the list are assigned.

4.2 PARTICLES SWARMING ALGORITHM

The particles swarm optimization algorithm was introduced to study social and cognitive behavior, but it has been largely applied as a problem-solving technique in engineering and computer science. There are two main version of the PSO algorithm: binary and a real valued version. With the exception of the representation, the two version of the algorithm are very much the same, thus only the real valued version will be discussed here.

The particles swarm approach assumes a population of individuals represented as binary strings or real valued vectors-particles, which suffer an iterative procedure of adapt ion to their environment, It also assumes that these individuals are social, which implies that they are capable of interacting with other individuals with a given neighborhood.

There are two main types of information available to each individual of the population. The first is their own past experience PSO algorithm is not only a tool for optimization, but also a tool for representing sociocognition of human and artificial agents, based on principles of social psychology. Some scientists suggest that knowledge is optimized by social interaction and thinking is not only private but also interpersonal. PSO as an optimization tool provides a population-based search procedure in which individuals called particles change their position (state) with time.

In a PSO system, particles fly around in a multidimensional search space. During flight, each particle adjusts its position according to its own experience, and according to the experience of a neighboring particle, making use of the best position encountered by itself and its neighbor.

Thus, as in modern GA and mimetic algorithms, a PSO system combines local search methods with global search methods, attempting to balance exploration and exploitation

When dealing with complex optimizations problems, evolutionary computation techniques have proven very helpful. Amongst optimization algorithms driven by evolutionary computation techniques, particle swarm algorithms have proven to be a very good alternative to genetic algorithms because of their faster convergence.

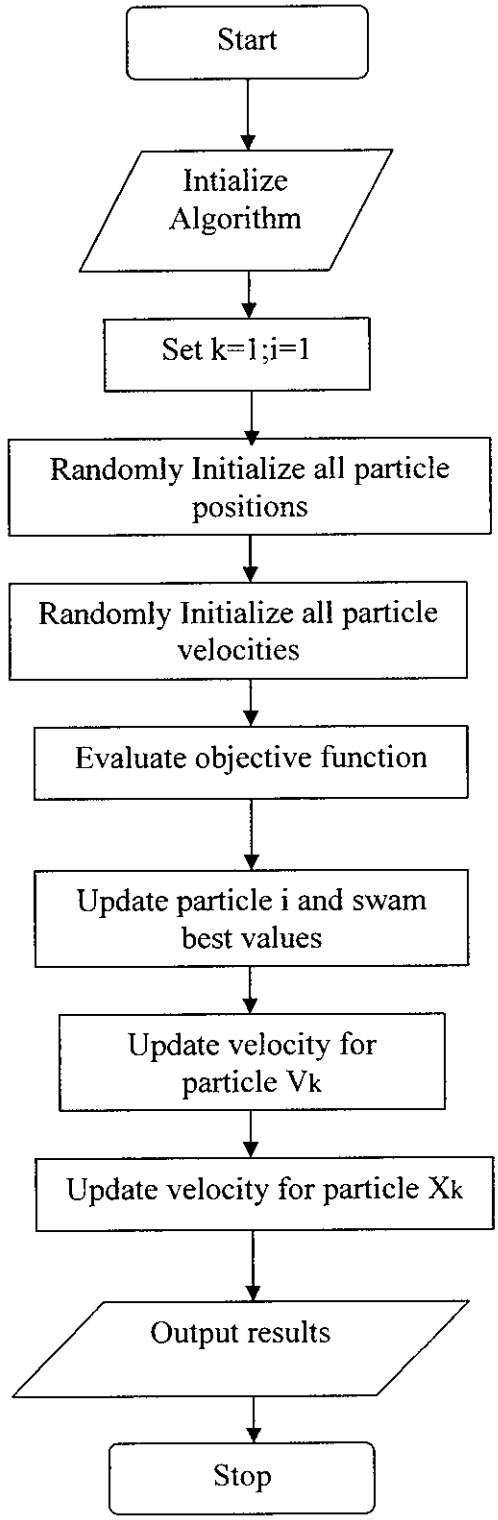


FIGURE 4.1 STRUCTURE OF PARTICLES SWARMING ALGORITHM

4.3 PROGRAM OF PARTICLE SWARMING

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
#define N_M  3
#define schd_cnt 4
#define N_J   9
void xsort(void);
//global variables
int
wrk[schd_cnt][100][N_J],i,j,k,rand1,rand2,wrk_cnt=0,s_or[3],max,p_ch1[schd_cn
t][100],p_ch2[schd_cnt][100];
int pai1=2,pai2=2,vi,g_bst_run[2],l_bst_run[2],run_cnt;
double avg_rpi[schd_cnt][100],l_bst,g_bst,v;
double
vel1[schd_cnt]={1.5,0.41},vel2[schd_cnt]={0.9,0.74},vel3[schd_cnt]={0.8,0.73},
vel4[schd_cnt]={0.55,0.71};
int wrk_wt[N_J]={92,411,158,359,550,9,716,724,745,};
int p_time[N_J]={6,4,7,2,3,8,1,9,3};
int m_proc_time[N_M];
void main()
{
int iii,jjj;
clrscr();
//first schedule
wrk[0][0][0]=2;
wrk[0][0][1]=1;
wrk[0][0][2]=8;
wrk[0][0][3]=5;
wrk[0][0][4]=7;
wrk[0][0][5]=0;
wrk[0][0][6]=4;
wrk[0][0][7]=6;
wrk[0][0][8]=3;
//second schedule
wrk[1][0][0]=8;
wrk[1][0][1]=0;
wrk[1][0][2]=2;
wrk[1][0][3]=3;
wrk[1][0][4]=7;
wrk[1][0][5]=4;
wrk[1][0][6]=5;
wrk[1][0][7]=6;
wrk[1][0][8]=1;
//Third scedule
wrk[2][0][0]=5;
```

```
wrk[2][0][3]=7;
wrk[2][0][4]=0;
wrk[2][0][5]=1;
wrk[2][0][6]=6;
wrk[2][0][7]=4;
wrk[2][0][8]=3;
//Fourth schedule
wrk[3][0][0]=8;
wrk[3][0][1]=2;
wrk[3][0][2]=0;
wrk[3][0][3]=3;
wrk[3][0][4]=4;
wrk[3][0][5]=1;
wrk[3][0][6]=7;
wrk[3][0][7]=5;
wrk[3][0][8]=6;
wrk_cnt=0;
for(iii=0;iii<schd_cnt;iii++)
p_ch1[iii][wrk_cnt]=p_ch2[iii][wrk_cnt]=0;
for(jjj=0;jjj<schd_cnt;jjj++)
{
for(iii=0;iii<N_J;iii++)
printf("%d  ",wrk[jjj][wrk_cnt][iii]);
printf("\n");
}
calc_rpi(wrk_cnt);
l_bst=g_bst=1;
for(iii=0;iii<schd_cnt;iii++)
{
if(avg_rpi[iii][wrk_cnt] < g_bst)
{
g_bst=avg_rpi[iii][wrk_cnt];
g_bst_run[0]=iii;
g_bst_run[1]=wrk_cnt;
}
}
for(iii=0;iii<schd_cnt;iii++)
{
if(avg_rpi[iii][wrk_cnt] > g_bst && avg_rpi[iii][wrk_cnt] < l_bst)
{
l_bst=avg_rpi[iii][wrk_cnt];
l_bst_run[0]=iii;
l_bst_run[1]=wrk_cnt;
}}
printf("\n%lf %lf",g_bst,l_bst);
printf("\ng_bst_pos is %d %d",g_bst_run[0],g_bst_run[1]);
printf("\nl_bst_pos is %d %d",l_bst_run[0],l_bst_run[1]);
getch();
```

```
//printf("%d",wrk_cnt);
p_ch1[0][wrk_cnt]=0;
p_ch2[0][wrk_cnt]=4;
p_ch1[1][wrk_cnt]=4;
p_ch2[1][wrk_cnt]=1;
p_ch1[2][wrk_cnt]=7;
p_ch2[2][wrk_cnt]=4;
p_ch1[3][wrk_cnt]=7;
p_ch2[3][wrk_cnt]=2;
for(jjj=0;jjj<schd_cnt;jjj++)
for(iii=0;iii<N_J;iii++)
{
//printf("%d  %d",p_ch1[wrk_cnt],p_ch2[wrk_cnt]);
if(iii == p_ch1[jjj][wrk_cnt])
{
//   printf("p_ch1 matched");
wrk[jjj][wrk_cnt][iii]=wrk[jjj][wrk_cnt-1][p_ch2[jjj][wrk_cnt]];
//   printf("%d ",wrk[wrk_cnt][iii]);
continue;
}
if(iii == p_ch2[jjj][wrk_cnt])
{
//   printf("p_ch2 matched");
wrk[jjj][wrk_cnt][iii]=wrk[jjj][wrk_cnt-1][p_ch1[jjj][wrk_cnt]];
//   printf("%d ",wrk[wrk_cnt][iii]);
continue;
}
wrk[jjj][wrk_cnt][iii]=wrk[jjj][wrk_cnt-1][iii];
}
printf("\n");
for(jjj=0;jjj<schd_cnt;jjj++)
{
for(iii=0;iii<N_J;iii++)
printf("%d  ", wrk[jjj][wrk_cnt][iii]);
printf("\n");
}
getch();

calc_rpi(wrk_cnt);

for(iii=0;iii<schd_cnt;iii++)
{
if(avg_rpi[iii][wrk_cnt] < g_bst)
{
g_bst=avg_rpi[iii][wrk_cnt];
g_bst_run[0]=iii;
g_bst_run[1]=wrk_cnt;
}
```

```
l_bst=avg_rpi[iii][wrk_cnt];
l_bst_run[0]=iii;
l_bst_run[1]=wrk_cnt;
}
}
printf("\n%lf %lf",g_bst,l_bst);
printf("\ng_bst_pos is %d %d",g_bst_run[0],g_bst_run[1]);
printf("\nl_bst_pos is %d %d",l_bst_run[0],l_bst_run[1]);
getch();
/*
if(avg_rpi[wrk_cnt] < g_bst || avg_rpi[wrk_cnt] == g_bst)
{
g_bst=avg_rpi[wrk_cnt];
g_bst_run=wrk_cnt;
}
if(avg_rpi[wrk_cnt] > g_bst && (avg_rpi[wrk_cnt] < l_bst || avg_rpi[wrk_cnt] ==
l_bst))
{
l_bst = avg_rpi[wrk_cnt];
l_bst_run=wrk_cnt;
}
*/
//printf("\n%d %d",p_ch1,p_ch2);
//printf("\n%lf %lf",vel1,vel2);
//p_ch1=(int)ceil((p_ch1+vel1));
//p_ch2=(int)ceil((p_ch2+vel2));
wrk_cnt++;
//printf("%d",wrk_cnt);
//printf("%d %d", p_ch1[wrk_cnt-1],p_ch2[wrk_cnt-1]);
for(jjj=0;jjj<schd_cnt;jjj++)
{
p_ch1[jjj][wrk_cnt]=p_ch1[jjj][wrk_cnt-1]+ceil(vel1[jjj]);
p_ch2[jjj][wrk_cnt]=p_ch2[jjj][wrk_cnt-1]+ceil(vel2[jjj]);
}
for(jjj=0;jjj<schd_cnt;jjj++)
printf("\n%d %d", p_ch1[jjj][wrk_cnt],p_ch2[jjj][wrk_cnt]);
getch();


for(jjj=0;jjj<schd_cnt;jjj++)
for(iii=0;iii<N_J;iii++)
{
if(iii == p_ch1[jjj][wrk_cnt])
{
wrk[jjj][wrk_cnt][iii]=wrk[jjj][wrk_cnt-1][p_ch2[jjj][wrk_cnt]];
//continue;
}
else if(iii == p_ch2[jjj][wrk_cnt])
```

```
//continue;
}
else
wrk[jjj][wrk_cnt][iii]=wrk[jjj][wrk_cnt-1][iii];
}
printf("\n");
for(jjj=0;jjj<schd_cnt;jjj++)
{
for(iii=0;iii<N_J;iii++)
printf("%d  ", wrk[jjj][wrk_cnt][iii]);
printf("\n");
}
getch();

calc_rpi(wrk_cnt);
for(iii=0;iii<schd_cnt;iii++)
{
if(avg_rpi[iii][wrk_cnt] < g_bst)
{
g_bst=avg_rpi[iii][wrk_cnt];
g_bst_run[0]=iii;
g_bst_run[1]=wrk_cnt;
}
if(avg_rpi[iii][wrk_cnt] > g_bst && avg_rpi[iii][wrk_cnt] < l_bst)
{
l_bst=avg_rpi[iii][wrk_cnt];
l_bst_run[0]=iii;
l_bst_run[1]=wrk_cnt;
}
}
printf("\n%lf %lf",g_bst,l_bst);
printf("\ng_bst_pos is %d %d",g_bst_run[0],g_bst_run[1]);
printf("\nl_bst_pos is %d %d",l_bst_run[0],l_bst_run[1]);
getch();
//printf("%lf",avg_rpi[wrk_cnt]);
/*
if(avg_rpi[wrk_cnt] < g_bst || avg_rpi[wrk_cnt] == g_bst)
{
g_bst=avg_rpi[wrk_cnt];
g_bst_run=wrk_cnt;
}
if(avg_rpi[wrk_cnt] > g_bst && (avg_rpi[wrk_cnt] < l_bst || avg_rpi[wrk_cnt] ==
l_bst))
{
l_bst = avg_rpi[wrk_cnt];
l_bst_run=wrk_cnt;
}
*/
```

```
//applying formulae - particle swarming
for(run_cnt=0;run_cnt<30;run_cnt++)
{
wrk_cnt++;
for(jjj=0;jjj<schd_cnt;jjj++)
{
v+=vel1[jjj];
v+=(float)(pai1*(p_ch1[l_bst_run[0]][l_bst_run[1]]-p_ch1[jjj][wrk_cnt-1]));
v+=(float)(pai2*(p_ch1[g_bst_run[0]][g_bst_run[1]]-p_ch1[jjj][wrk_cnt-1]));
//printf("\nv=%lf",v);
if(v<0)v=-2.25;
if(v>0)v=2.25;
vi=ceil(v+p_ch1[jjj][wrk_cnt-1]);
if(vi > 8) vi=8;
if(vi <0) vi=0;
//printf("\n\nvi value is %d",vi);
p_ch1[jjj][wrk_cnt]=vi;
v=vel2[jjj]+(float)(pai1*(p_ch2[l_bst_run[0]][l_bst_run[1]]-p_ch2[jjj][wrk_cnt-
1])+pai2*(p_ch2[g_bst_run[0]][g_bst_run[1]]-p_ch2[jjj][wrk_cnt-1]));
//printf("\nv=%lf",v);
if(v<0)v=-2.25;
if(v>0)v=2.25;
//printf("\np_ch2=%d",p_ch2[wrk_cnt-1]);
vi=ceil(v+p_ch2[jjj][wrk_cnt-1]);
if(vi > 8) vi=8;
if(vi <0) vi=0;
//printf("\n\nvi value is %d",vi);
p_ch2[jjj][wrk_cnt]=vi;
//printf("\np_ch1=%d p_ch2=%d",p_ch1[jjj][wrk_cnt],p_ch2[jjj][wrk_cnt]);
for(iii=0;iii<N_J;iii++)
{
if(iii == p_ch1[jjj][wrk_cnt])
{
wrk[jjj][wrk_cnt][iii]=wrk[jjj][wrk_cnt-1][p_ch2[jjj][wrk_cnt]];
//continue;
}
else if(iii == p_ch2[jjj][wrk_cnt])
{
wrk[jjj][wrk_cnt][iii]=wrk[jjj][wrk_cnt-1][p_ch1[jjj][wrk_cnt]];
//continue;
}
else
wrk[jjj][wrk_cnt][iii]=wrk[jjj][wrk_cnt-1][iii];
}
}
printf("\n");
for(jjj=0;jjj<schd_cnt;jjj++)
{
```

```
printf("\n");
}
getch();
calc_rpi(wrk_cnt);
for(iii=0;iii<schd_cnt;iii++)
{
if(avg_rpi[iii][wrk_cnt] < g_bst)
{
g_bst=avg_rpi[iii][wrk_cnt];
g_bst_run[0]=iii;
g_bst_run[1]=wrk_cnt;
}
if(avg_rpi[iii][wrk_cnt] > g_bst && avg_rpi[iii][wrk_cnt] < l_bst)
{
l_bst=avg_rpi[iii][wrk_cnt];
l_bst_run[0]=iii;
l_bst_run[1]=wrk_cnt;
}
}


//printf("%lf",avg_rpi[wrk_cnt]);
/*
if(avg_rpi[wrk_cnt] < g_bst || avg_rpi[wrk_cnt] == g_bst)
{
g_bst=avg_rpi[wrk_cnt];
g_bst_run=wrk_cnt;
}
if(avg_rpi[wrk_cnt] > g_bst && (avg_rpi[wrk_cnt] < l_bst || avg_rpi[wrk_cnt] ==
l_bst))
{
l_bst = avg_rpi[wrk_cnt];
l_bst_run=wrk_cnt;
}
*/
printf("\n\n%lf %lf", g_bst,l_bst);
printf("\n\ng_bst_schd=%d g_bst_run=%d",g_bst_run[0],g_bst_run[1]);
printf("\nl_bst_schd=%d l_bst_run=%d",l_bst_run[0],l_bst_run[1]);
printf("\n\n");
getch();
}
}
calc_rpi(int cnt)
{
int i,temp,j,kk;
for(j=0;j<schd_cnt;j++)
{
for(kk=0;kk<N_M;kk++)
```

```
}
//        m_proc_time[1]=p_time[wrk[j][cnt][1]]*wrk_wt[wrk[j][cnt][1]];
//        m_proc_time[2]=p_time[wrk[j][cnt][2]]*wrk_wt[wrk[j][cnt][2]];
//        m_proc_time[3]=p_time[wrk[j][cnt][3]]*wrk_wt[wrk[j][cnt][3]];
//        printf("\n%d\t %d\t
%d\n",m_proc_time[0],m_proc_time[1],m_proc_time[2]);
for(i=N_M;i<N_J;i++)
{
xsort();
k=s_or[0];
//        printf("%d\t",k);
temp=p_time[wrk[j][cnt][i]];
m_proc_time[k]+=temp*wrk_wt[wrk[j][cnt][i]];
//        printf("%d\t",m_proc_time[k]);
//        getch();
}
//        printf("%d %d",k,m_proc_time[k]);
//        getch();

if(k == 0)
{
if(m_proc_time[1] > m_proc_time[2])
{
max=m_proc_time[1];
avg_rpi[j][wrk_cnt] =(float)(max-m_proc_time[0])/max;
avg_rpi[j][wrk_cnt]+=(float)(max-m_proc_time[2])/max;
}
else
{
max = m_proc_time[2];
avg_rpi[j][wrk_cnt]=(float)(max-m_proc_time[1])/max;
avg_rpi[j][wrk_cnt]+=(float)(max-m_proc_time[0])/max;
}
}
else if(k == 1)
{
if(m_proc_time[0] > m_proc_time[2])
{
max=m_proc_time[0];
avg_rpi[j][wrk_cnt] =(float)(max-m_proc_time[1])/max;
avg_rpi[j][wrk_cnt]+=(float)(max-m_proc_time[2])/max;
}
else
{
max = m_proc_time[2];
avg_rpi[j][wrk_cnt]=(float)(max-m_proc_time[0])/max;
avg_rpi[j][wrk_cnt]+=(float)(max-m_proc_time[1])/max;
}
```

```
{
if(m_proc_time[1] > m_proc_time[0])
{
max=m_proc_time[1];
avg_rpi[j][wrk_cnt]= (float)(max-m_proc_time[0])/max;
avg_rpi[j][wrk_cnt]+=(float)(max-m_proc_time[2])/max;
}
else
{
max = m_proc_time[0];
avg_rpi[j][wrk_cnt] =(float)(max-m_proc_time[1])/max;
avg_rpi[j][wrk_cnt]+=(float)(max-m_proc_time[2])/max;
}
}
//for(i=0;i<3;i++)
//printf("%d\t",s_or[i]);
avg_rpi[j][wrk_cnt]/=2;
printf("\navg_rpi is %lf",avg_rpi[j][wrk_cnt]);
}
}
void xsort(void)
{
int ii,jj,temp[N_M],ttemp;
for(ii=0;ii<N_M;ii++)
temp[ii]=m_proc_time[ii];

//        for(ii=0;ii<3;ii++)
//        printf("%d\t" ,temp[ii]);

for(ii=0;ii<N_M;ii++)
{
for(jj=ii+1;jj<N_M;jj++)
{
if(temp[ii] < temp[jj])
s_or[ii]=ii;
else
{
ttemp=temp[ii];
temp[ii]=temp[jj];
temp[jj]=ttemp;
s_or[ii]=jj;
}
}
}
for(ii=0;ii<N_M;ii++)
if(temp[0] == m_proc_time[ii])
s_or[0]=ii;
/*
```

```
if(m_proc_time[0] < m_proc_time[2])
{
s_or[0]=0;
if(m_proc_time[1] < m_proc_time[2])
{
s_or[1]=1;
s_or[2]=2;
}
else
{
s_or[1]=2;
s_or[2]=1;
}
}
else
{
s_or[0]=2;
if(m_proc_time[0] < m_proc_time[1])
{
s_or[1]=0;
s_or[2]=2;
}
else
{
s_or[1]=2;
s_or[2]=0;
}
}
}
if(m_proc_time[0] < m_proc_time[2])
{
s_or[1]=0;
s_or[2]=2;
}
else
{s_or[1]=2;
s_or[2]=0;
}
*/
}
```

4.4 ALGORITHM OF PARTICLES SWARMING

Initialisation :Randomly initialize a population of a particles.

Population Loop : for each particle do

Goodness Evaluation and update :evaluate the goodness of the particle.If its goodness is greater than its best goodness So far,then this particle becomes best particle so far.

Neighborhood Evaluation : if the goodness of this particle is best along all its neighbors, then this particle becomes the best particle of the Neighborhood.

Determine V_i :apply equation

Particle Update : apply the updating rule

Cycle : repeat step 2 until a given convergence criterion is met

$X_i(t) = f(X_i(t-1), V_i(t-1), P_i, P_g)$

$V_i(t) = V_i(t-1) + \text{constant}(P_i - X_i(t-1)) + \text{constant}(P_g - X_i(t-1))$

4.5 GENERATION OF RANDOM NUMBERS

A computer simulation model was written in the 'C' language on an IBM/PC compatible system to represent the real situation. The number of operations and machines are the input to the program. The operation times are generated randomly using the built-in random generator of the computer. The number of preceding operations for each succeeding operation and their job identities are also generated randomly.

4.6 IMPORTANT PARAMETERS

The 20 Schedules are taken ,Assign 2 random values for each Schedules ,and find the RPI ,then Exchange the position for each Schedule and find the RPI ,and find the local value and global value, Global value is the most minimum value update the velocity, Apply the formula ,and find the position, Iteration is repeated and most minimum value is found.

4.7 APPLICATION OF PSO IN WORKFLOW BALANCING

Workflow balancing is used to distribute the workloads among the machines to reduce the idle time. The machine with the maximum workload is the bottle-neck

achieving high throughput. It also affects the system's utilizations. Hence, the workflow balancing is used to remove the bottle-neck available in the system. The manufacturing system attains the optimal overall performance due to the maximum output at this most constrained resource. This results in reduced work-in-process and finished goods inventories and minimized operating expenses. Shop floor scheduling prepares a more detailed production plan by determining the timings, sequences and workload assignments. The workflow balancing is used to find the optimum allocation of 'n' jobs on 'm' machines as the part of a manufacturing plan.

Step 1: The jobs are scheduled as they arrived to the machines in the first iteration. The jobs represented as the processing times of the respective jobs. For example if the job orders may as follows.

Job order: 0.87 0.34 0.55 0.79 0.10 0.56 0.51 0.19 0.55 0.23

The jobs are allocated to machines in such way that the machine with the smallest workload of the machine is selected for allocating the jobs. For example if the machines may 4 then the allocation may follows.

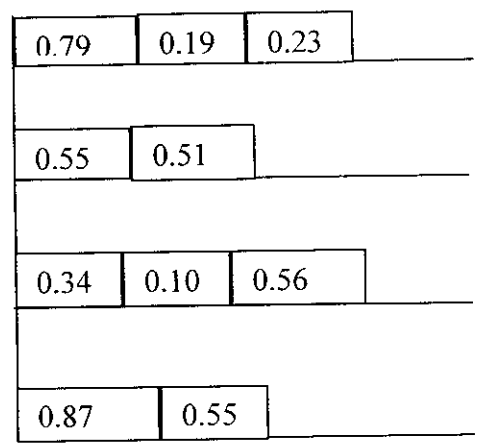


FIGURE 4.2 ALLOCATION OF FIRST SET OF JOBS

The work load of each machine is calculated by summing all the processing time of jobs assigned to the particular machine. Then relative percentage of imbalance

percentage of deviation of workloads on machines from the upper bound of maximum workload. Finally, the average RPI value is calculated and the value is 17.43.

Step2: Any two job's positions are changed randomly. Here 8^{th} and 6^{th} job's positions are changed. Then the Job order and allocation of the jobs to the machines may as follows and allocation of the jobs.

Job order: 0.87 0.34 0.55 0.79 0.10 0.19 0.51 0.56 0.55 0.23

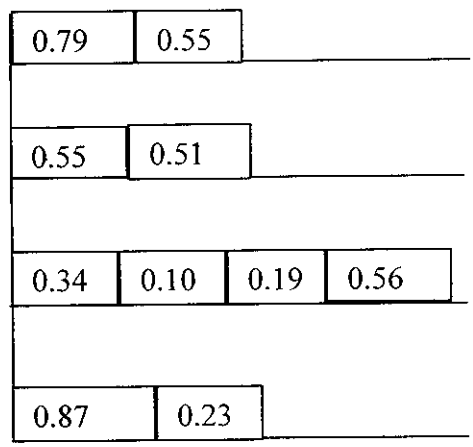


FIGURE 4.3 ALLOCATION OF SECOND SET OF JOBS

Here also the same procedure is adopted as explained earlier and here the average RPI is 12.5.Further Operation is done by using Two Formulas.

$X_i(t) = f(X_i(t-1), V_i(t-1), P_i, P_g)$

$V_i(t) = V_i(t-1) + \text{constant}(P_i - X_i(t-1)) + \text{constant}(P_g - X_i(t-1))$

4.8 ITERATIONS

In parallel machine scheduling all the machines are identical and a job can be processed on any one of the machines. The PSO is used to compute the relative percentage of imbalance in workloads among the machines. The initial population or seed is generated randomly. The machine with the less cumulative workload is selected to assign the jobs from the unscheduled list. The maximum workload among all the machines is identified, and the RPI values of each machine

positions of the jobs to evaluate the next iteration. This is calculated by using the following formulae,

$$X_i(t) = f(X_i(t-1), V_i(t-1), P_i, P_g)$$

$$V_i(t) = V_i(t-1) + \text{constant}(P_i - X_i(t-1)) + \text{constant}(P_g - X_i(t-1))$$

where

X_1	Job position of previous iteration
X_2	Job position of previous iteration
X_1^1	Job position of new iteration
X_2^1	Job position of new iteration
P_g	Global Best
P_i	Local Best
V_i	Velocity
r_i	Random numbers
n	Number of jobs

The next step is calculating the average RPI values using the new position values.

Initialisation: Randomly initialize a population of a particles.

Population Loop : for each particle do

Goodness Evaluation and update: evaluate the goodness of the particle.If its goodness is greater than its best goodness So far, then this particle becomes best particle so far.

Neighborhood Evaluation : if the goodness of this particle is best along all its neighbors, then this particle becomes the best particle of the Neighborhood.

Determine V_i :apply equation

Particle Update : apply the updating rule

Cycle : repeat step 2 until a given convergence criterion is met

4.9 EVALUATION PARAMETER

Many performance measures are available to evaluate the performance of various scheduling heuristics. The best allocation of jobs amongst the machines can be obtained when their workloads are balanced. The performance measure of relative percentage of imbalance (RPI) in workload is adapted to evaluating the performance of the scheduling strategies: RANDOM, SPT, LPT, and PSO in parallel machine scheduling.

Scheduling strategies are used for the selection of jobs for allocation on machines in parallel machine scheduling. These scheduling strategies mostly affect shop floor performance. In order to achieve higher productivity on the shop floor, an appropriate strategy needs to be followed based on all the machines are calculated and compared with the maximum workload of a particular machine. The machine with the maximum workload is the bottleneck, which affects the shop floor performance.

The maximum load 'W_{max}' on a machine is taken as the index for comparing the workload on other machines for evaluation purposes. The difference between the maximum workload and the available workload on a machine is called an imbalance in workload. Imbalances in machine workloads are expressed as,

$$I_i = W_{max} - W_i \quad i = 1, 2, 3 \ldots m$$

The various workload balancing strategies are adopted to minimize these imbalances on the machines. The best schedule is the uniform distribution of workloads on all the machines. A careful study of the allocation of work in the bottle-neck machine contributes towards the improvement of shop floor performance. The allocation of jobs may be altered on the machines to minimize the maximum workload on the bottleneck machine. Some of the allocated jobs from the bottle-neck machine can be moved to other imbalanced machines to minimize the workload on it. The RPI in workloads of all the machines is adopted to compare the performance of the scheduling strategies. This indicates the percentage of deviation of workloads on machines from the upper bound of

$$RPI = [(W_{max} - W_i) / W_{max}] \times 100$$

The average RPI values are calculated using 'EXCEL' using the output of the simulation model. The output of the simulation model shows the work allotments to the machines in terms of job processing times. The four scheduling strategies produce four different work allotments.

The maximum workload on the 'W_{max}' machine is identified and the percentage of deviation from this maximum workload for all the four rules. The workload is balanced perfectly if the percentage of imbalance among the machines is zero.

CHAPTER 5

COMPUTATIONAL

CHAPTER 5

COMPUTATIONAL EXPERIMENTS

5.1 COMPUTATIONAL EXPERIMENTS

The simulation model has been coded in the 'C' language on the IBM/PC compatible system. The number of jobs and machines are the input data given to the program. The processing time of jobs is generated using the built-in random number generator of the compiler. The experiments have been conducted on an IBM/PC compatible system. The number of jobs in each set has been varied and the total number of sets considered for the study is 10. The output of all the instances is directed to the 'EXCEL' worksheet.

The maximum workloads on machines for each heuristic are found. The maximum workload is compared with the workloads of each machine. The difference between the maximum and available workloads on each machine is calculated. This measure indicates the imbalance of workloads on machines. Considering the maximum workload as the index, the RPI for each machine is calculated for all the heuristics. This shows the deviation of workloads on machines from the upper bound of maximum workload. The average of the RPI on machines for all heuristics is calculated for each set of 'n' jobs. The charts for all the sets of 'm' machines are also shown. The behavior of heuristics is shown in plotted lines by using number of jobs and average RPI on 'X' and 'Y' axes respectively. The results are shown in below.

5.2 'n' JOBS Vs TWO MACHINES

The parallel machine-scheduling problem is considered for assigning jobs of various sizes on two machines. The average RPI values of workloads on the machines are shown in Table 5.2 and Fig. 5.2. The RANDOM strategy produces moderate RPI values. The SPT produces large RPI values. The LPT produces smaller RPI than the RANDOM, and SPT, but it produces large RPI than PSO.

smaller RPI values indicate the better distribution of workloads between two machines.

TABLE 5.2 RPI Vs TWO MACHINES

No. Of jobs	RANDOM	SPT	LPT	PSO
10	6.94	8.78	1.09	0.11
20	3.44	4.61	0.35	0.02
30	2.18	3.21	0.14	0.11
40	1.56	2.3	0.08	0.01
50	1.53	1.85	0.04	0.05
60	1.05	1.61	0.05	0.02
70	0.95	1.38	0.03	0.03
80	0.83	1.23	0.02	0.01
90	0.72	1.08	0.02	0.02
100	0.47	0.96	0.02	0.09

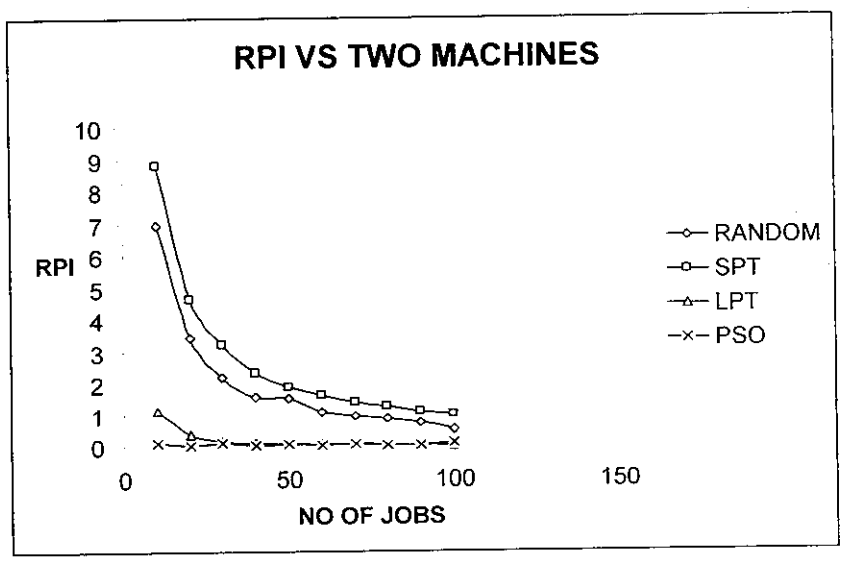


FIGURE 5.2 RPI Vs TWO MACHINES

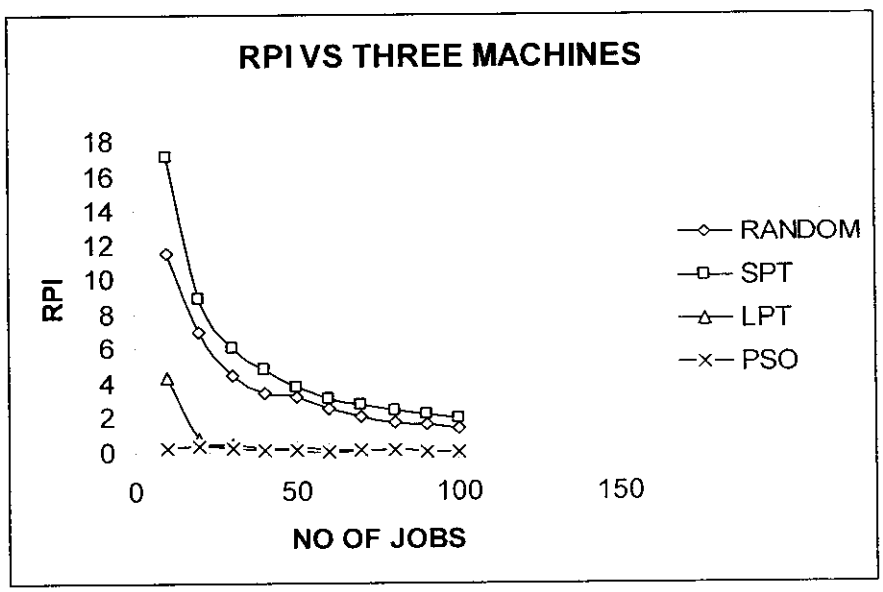
5.3 'n' JOBS Vs THREE MACHINES

The performances of four heuristics on three machines are shown in Table 5.3and

smaller RPI values. The RANDOM produces moderate values and LPT produces smaller values than RANDOM, and SPT, but greater than PSO as same as two machines Vs RPI. Thus, PSO produces balanced schedules for three machines problems.

TABLE 5.3 RPI Vs THREE MACHINES

No. Of jobs	RANDOM	SPT	LPT	PSO
10	11.46	17.12	4.36	0.24
20	6.99	8.89	0.8	0.35
30	4.44	6.05	0.45	0.21
40	3.42	4.83	0.24	0.08
50	3.15	3.77	0.18	0.16
60	2.49	3.07	0.1	0.04
70	2.03	2.77	0.08	0.07
80	1.75	2.42	0.07	0.08
90	1.59	2.14	0.05	0.03
100	1.35	1.94	0.03	0.04



5.4 'n' JOBS Vs FOUR MACHINES

The behavior of scheduling of jobs on four parallel machines is shown in Table 5.4 and Fig. 5.4. Here also the PSO produces the balanced workflow among all the machines and the RANDOM produces the moderate RPI values, the SPT produces the large RPI values and the LPT produces the large RPI than PSO and less value than SPT, and Random.

TABLE 5.4 RPI Vs FOUR MACHINES

No. of jobs	RANDOM	SPT	LPT	PSO
10	18.19	23.96	5.19	0.19
20	10.15	12.5	1.78	0.31
30	7.08	9.12	0.82	0.27
40	5.65	6.76	0.44	0.15
50	4.5	5.64	0.28	0.25
60	4.1	4.42	0.22	0.23
70	3.3	4.14	0.19	0.2
80	3.03	3.53	0.13	0.11
90	2.75	3.29	0.12	0.12
100	2.2	2.87	0.06	0.06

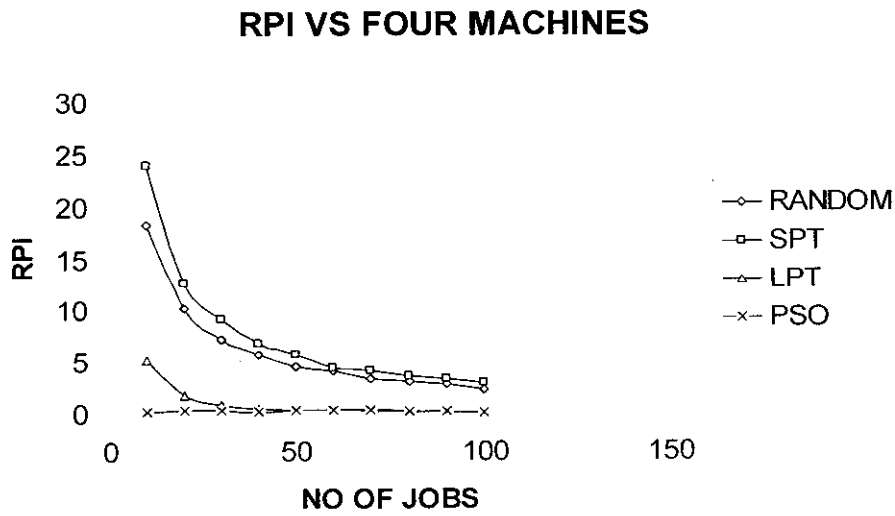


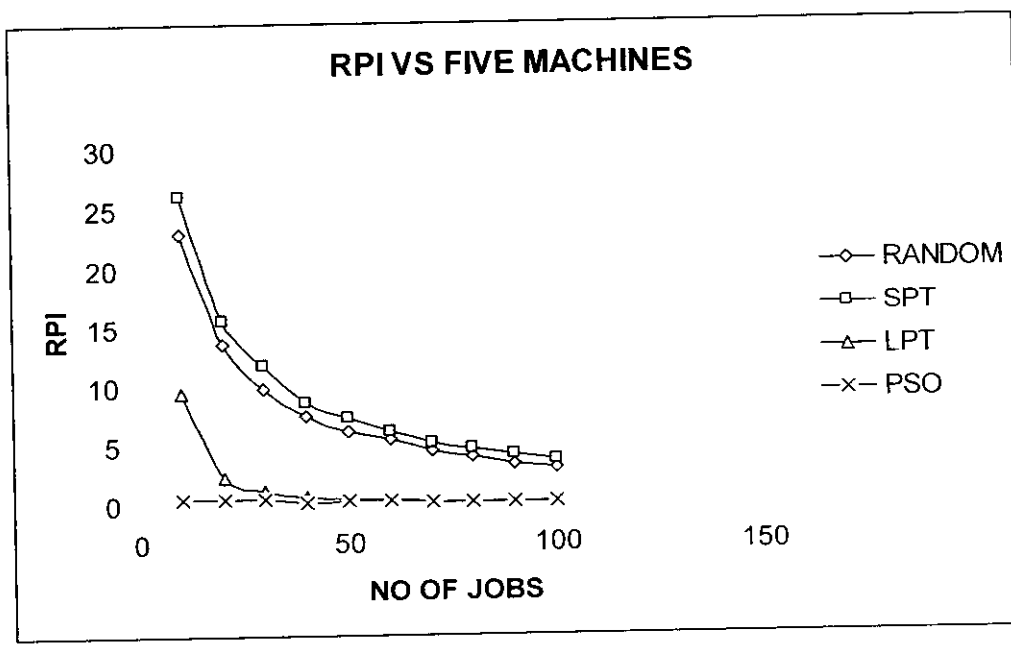
FIGURE 5.4 RPI Vs FOUR MACHINES

5.5 'n' JOBS Vs FIVE MACHINES

This shows the behavior of RPI and five machines in Table 5.5 and Fig 5.5. This also shows that PSO produces the better results.

TABLE 5.5 RPI Vs FIVE MACHINES

No. of jobs	RANDOM	SPT	LPT	PSO
10	23.04	26.18	9.49	0.48
20	13.61	15.6	2.36	0.43
30	9.88	11.85	1.23	0.45
40	7.58	8.7	0.67	0.18
50	6.1	7.26	0.45	0.36
60	5.54	6.16	0.38	0.3
70	4.51	5.18	0.25	0.15
80	3.97	4.63	0.17	0.14
90	3.38	4.16	0.13	0.17
100	3.02	3.68	0.14	0.16



5.6 'n' JOBS Vs SIX MACHINES

Table 5.6 and Fig 5.6shows the results of performances of various heuristics for six machines. However, here also PSO produces improved results over all the other heuristics.

TABLE 5.6 RPI Vs SIX MACHINES

No. of jobs	RANDOM	SPT	LPT	PSO
10	29.22	35.44	16.49	0.35
20	16.38	20.53	4.09	0.17
30	12.47	14.28	1.71	0.68
40	9.08	11.23	0.96	0.29
50	7.09	9.15	0.74	0.39
60	6.11	7.51	0.45	0.36
70	6.24	6.84	0.4	0.44
80	5.13	5.8	0.26	0.29
90	4.55	5.02	0.2	0.19
100	3.87	4.66	0.23	0.15

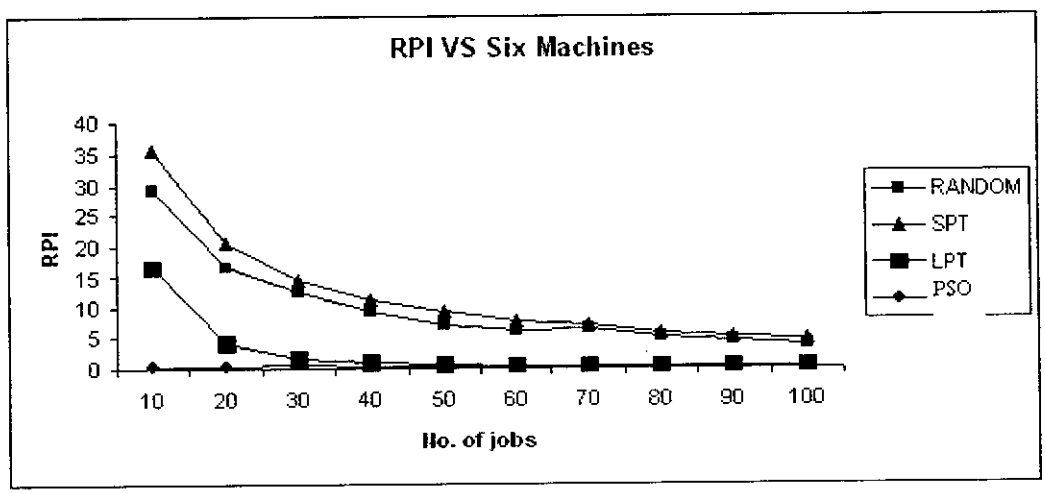


FIGURE 5.6 RPI Vs SIX MACHINES

CHAPTER 6

PRECEDENCE

CHAPTER 6

PRECEDENCE CONSTRAINTS

6.1 INTRODUCTION

Machine scheduling involves the allocation of jobs to machine based on some objective function. Different algorithms have been developed by researchers to estimate the performance of scheduling. The assembly planning of the textile machine considered in this project is a repetitive type manufacturing system, which involves allocation of assembly operations to be executed in parallel by operators. The operators are cross-trained to perform all kinds of operations required for a complete assembly. The product of the assembly has the same geometry and assembly operational sequences. There are 'n' operations to be executed by 'm' operations in parallel to complete the assembly. There are some precedence operations to be allocated first among the all operations. This is modeled as a classical parallel machine scheduling problem with precedence constraints.

In parallel machine scheduling with precedence constraints, there are 'n' jobs to be scheduled on 'm' machines satisfying the given precedence constraints. An operation cannot be started until all its precedence operations are completed. In this project, the workflow balancing is used to allocate the operations to the operators in the assembly section. Workflow balancing among parallel machines is essential to maximize the optimum uses of the resources. It helps to remove the bottle-necks present in a manufacturing system to improve the throughput. Workflow refers to the workloads of machines. Workload of a machine is the sum of processing time of all jobs assigned to it. Idle time is not considered for calculating the workflow.

A machine with the lowest workload is selected for assigning a new job from the lists of jobs. The assembly operations are scheduled to the operators in such a way that their workloads need to be balanced. The selection of operations for the

assignment may be based on many different scheduling strategies. Particles Swarming Algorithm (PSA) is used to prepare the lists of jobs to be assigned.

The performance of PSO is compared with three workflow balancing strategies, namely random (RANDOM), shortest processing time (SPT), and longest processing time (LPT). All the preceding operations are scheduled first and succeeding operations are scheduled subsequently. Both the preceding and succeeding operations are selected individually according to the above four assembly scheduling strategies. The jobs are assigned to parallel machines to minimize the imbalance in workloads. The minimum imbalance in workloads ensures balanced workflow among the machines.

The relative percentage of imbalance (RPI) is adopted among the parallel machines for evaluating the performance of the heuristics to a standard manufacturing environment. A computer simulation model has been developed to replicate the actual situation and its output obtained for analysis. The PSO shows better performance for the combination of various operations and operators. A computer program has been coded on an IBM/PC compatible system in the 'C' language for experimentation to a standard manufacturing system environment in operation and charts are drawn using 'EXCEL' for analysis.

6.2 APPLICATION OF PRECEDENCE CONSTRAINTS IN ASSEMBLY SECTION

The textile machinery-manufacturing unit considered here produces machines, namely carding, draw, and speed frames. These machines are used to produce yarn from raw cotton. Assembly planning of a speed frame is considered in this project. The length of speed frame is 25 meters. The required components of the complete assembly are brought to the assembly shop floor since it has a fixed position layout.

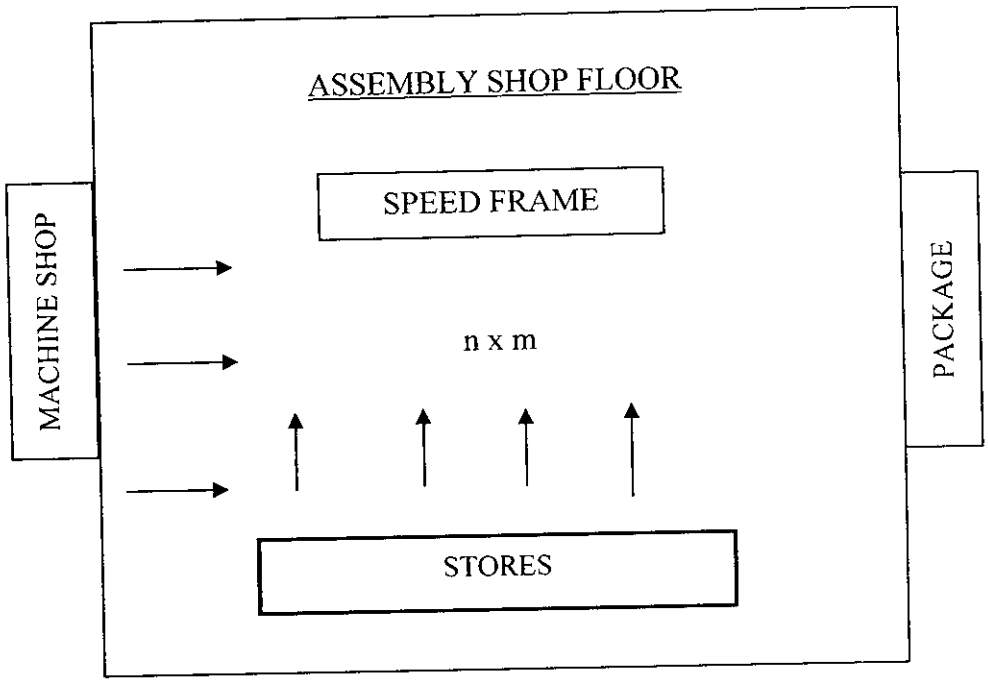


FIGURE 6.1 LAYOUT OF THE ASSEMBLY FLOOR

→ Flow of material

m Total number of operators

n Total number of assembly operations

The mainframe of the speed frame is erected at one place on the shop floor, and other components and sub assemblies are fixed on it along its entire length at various locations. Some components are fixed first because of precedence relationship to accommodate other fitting components. The fitting of sub assemblies and other components can be executed simultaneously at different places along the entire length of the mainframe. The aim of this project is to determine suitable scheduling strategies in allocating assembly operations to the operators so that their workloads are well balanced. Balanced workloads reduce idle time and ensure the earliest completion of assembly. This problem of workflow balancing for the assembly of a speed frame is modeled as a parallel machine scheduling problem using the analogy of operations for jobs and operators for machines with the precedence constraints.

The assembly scheduling problem consists of 'n' operations J_i, i = 1, 2, 3... n and 'm' operators M_i, i = 1, 2, 3 ... m. the operation 'J_i' requires a processing time

There are some precedence operations which must be completed before starting new operations. The operations are allocated such that the workloads of the operators are evenly distributed. An operation may be allocated only after ensuring that all precedence operations have been allocated. Let 'S' represent the list of assembly operations to be assigned to the operators. This list is constructed with operations numbers and their processing times:

$S = \{p_i\}, i = 1, 2, 3 \ldots n,$

where 'p_i' indicates the processing time of the i^{th} operation. Let 'S_1' be the list of preceding operations that must be assigned before succeeding operations, and let 'S_2' are extracted from the main list 'S':

$S_1 = \{p_i\}, i = 1, 2, 3 \ldots p,$ if i precedes j,

where 'p' is the number of preceding operations, and;

$S_2 = \{p_j\}, j = 1, 2, 3 \ldots q,$ if j succeeds i,

where 'q' is the number of succeeding operations.

The assembly operations must be assigned such that workloads of operators are well balanced to achieve maximum efficiency on the shop floor. Scheduling strategies play a vital role in producing good schedules with balanced workloads. Workload balancing is useful in achieving maximum system utilization. Workload balancing can be achieved by adopting different strategies to the list of operations.

6.3 METHODOLOGY

The assembly operations are partitioned into two sub-lists, 'S_1' and 'S_2', of the mail list 'S'. 'S' is the list containing all the assembly operations. Sub-list 'S_1' contains all the preceding operations and 'S_2' contains all the succeeding operations. All preceding operations are allocated before succeeding operations. It forms a list of jobs and which satisfy the precedence constraints. The objective of allocation is to ensure that the workloads of all operators are well balanced.

A performance measure based on workload is introduced to evaluate the different scheduling strategies to be used for assigning operations to operators. The four scheduling strategies RANDOM, SPT, LPT, and PSO are used for the selection of an operation from the list of unscheduled operations. The operator with the smallest workload is selected for assigning a new operation.

The scheduling strategies are applied to the list of operations. The performances of strategies are compared and the results are analyzed.

6.3.1 Workload Allocation

Step 1: Select an operation from the list of precedence operations 'S_1'. If all the precedence operations are allocated, then select an operation from the list of succeeding operations 'S_2'. The new list is sorted according to the strategies RANDOM, SPT, LPT, and PSO.

Step 2: Find out the cumulative workload 'W_i' of each operator. The workload 'W_k' of an operator is calculated by adding all the processing time of operations assigned to her. The workload of any operator is:

$$W_i = \sum_{j=1}^{t} p_j, \quad \text{if } j \in R_i \ (i = 1, 2, 3 \ldots m),$$

where 'R_i' is the list of allocated jobs,'t' is the total number of operations assigned, and 'm' is the number of operators.

Step 3: Select the operator 'M_k' with the smallest workload 'W_k' among 'm' operators for assigning the operations from the list of unscheduled operations. The lists of unscheduled operations are prepared according to the heuristics.

Step 4: Assign the operation 'J_i' from the lists of operations 'S_1' or 'S_2' to the operator 'M_k'.

Step 5: Repeat steps 2-4 until all the operations are assigned.

6.3.1.1 Assumptions

1. No pre-emption is allowed (i.e. once started, an operation must be completed before starting another operation).
2. Operators are cross-trained to handle all operations.
3. Operations are executed in parallel.
4. No operator may process more than one operation at a time.

6.4 APPLICATION OF PRECEDENCE CONSTRAINTS USING PSO

In parallel machine scheduling all the machines are identical and a job can be processed on any one of the machines. The PSO is used to compute the relative percentage of imbalance in workloads among the machines. The initial population or seed is generated randomly. The machine with the less cumulative workload is selected to assign the jobs from the unscheduled list.

The unscheduled list 'S' is divided into two list 'S1' and 'S2'. 'S1' contains all the preceding jobs and 'S2' contains all the succeeding jobs. The preceding jobs are first allocated and then the succeeding jobs are allocated to the available machine. The maximum workload among all the machines is identified, and the RPI values of each machine calculated. Then the average RPI is calculated. The next step is changing the positions of the jobs to evaluate the next iteration. This is calculated by using the following formulae,

$$X_i(t) = f(X_i(t-1), V_i(t-1), P_i, P_g)$$

$$V_i(t) = V_i(t-1) + \text{constant}(P_i - X_i(t-1)) + \text{constant}(P_g - X_i(t-1))$$

CHAPTER 7

COMPUTATIONAL

CHAPTER 7

COMPUTATIONAL EXPERIMENTS

7.1 COMPUTATIONAL EXPERIMENTS

The simulation model has been coded in the 'C' language on the IBM/PC compatible system. The number of operations and operators are the input data given to the program. The processing time of operations is generated using the built-in random number generator of the compiler

The experiments have been conducted on an IBM/PC compatible system. The number of operations in each set has been varied and the total number of sets considered for the study is 10. The output of all the instances is directed to the 'EXCEL' worksheet.

Maximum workloads on operators for each heuristic are found. The maximum workload is compared with the workloads of each machine. The difference between the maximum and available workloads on each machine is calculated. This measure indicates the imbalance of workloads on operators. Considering the maximum workload as the index, the RPI for each machine is calculated for all the heuristics.

This shows the deviation of workloads on operators from the upper bound of maximum workload. The average of the RPI on operators for all heuristics is calculated for each set of 'n' operations. The charts for all the sets of 'm' operators are also shown. The behavior of heuristics is shown in plotted lines by using number of operations and average RPI on 'X' and 'Y' axes respectively. The results are shown in below.

7.2 'n' OPERATIONS Vs TWO OPERATORS

The Fig. 7.2 and the Table 7.2 shows the performances of the RANDOM, SPT, LPT, and PSO strategies for two operators. This shows that the PSO provides the better

TABLE 7.2 RPI Vs TWO OPERATORS

Number of Operations	RANDOM	SPT	LPT	PSO
10	5.91	6.69	6.97	0.23
20	3.52	3.26	0.40	0.30
30	2.10	1.81	0.17	0.14
40	1.62	2.72	0.44	0.12
50	1.30	2.10	0.08	0.08
60	1.10	1.69	0.11	0.18
70	0.90	1.28	0.09	0.12
80	0.75	1.12	0.07	0.15
90	0.86	0.96	0.04	0.05
100	0.61	0.92	0.02	0.36

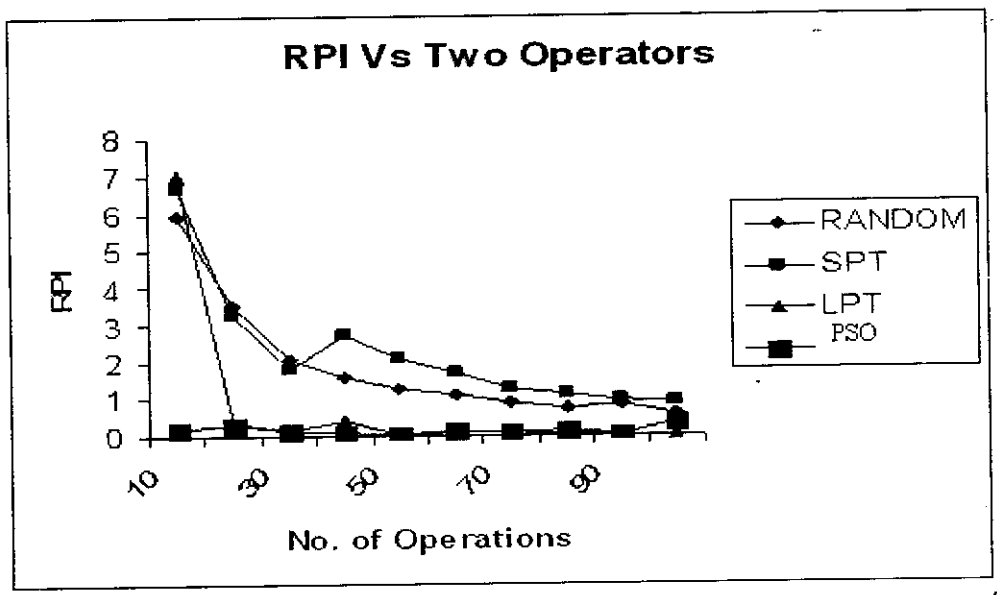


FIGURE 7.2 RPI Vs TWO OPERATORS

7.3 'n' OPERATIONS Vs THREE OPERATORS

The Table 7.3 and Fig. 7.3 provide the performances of RANDOM, SPT LPT, and PSO for three operators. This also shows that the PSO provides the optimal

TABLE 7.3 RPI Vs THREE OPERATORS

Number of Operations	RANDOM	SPT	LPT	PSO
10	12.29	14.29	9.09	0.16
20	6.76	5.20	0.36	0.17
30	4.58	5.20	0.97	0.08
40	4.07	4.71	0.55	0.42
50	3.07	6.02	0.63	0.49
60	2.34	3.63	0.31	0.38
70	2.25	2.65	0.16	0.46
80	1.80	2.21	0.20	0.36
90	1.54	2.04	0.13	0.17
100	1.49	1.87	0.10	0.23

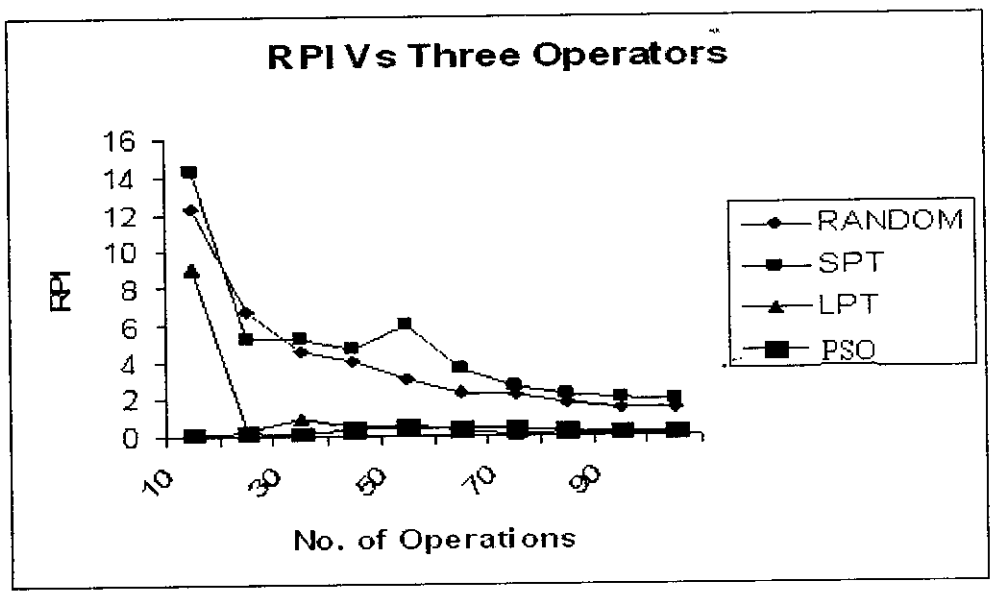


FIGURE 7.3 RPI Vs THREE OPERATORS

7.4 'n' OPERATIONS Vs FOUR OPERATORS

The Fig 7.4 and the Table 7.4 give the result for four operators Vs RPI values of RANDOM, SPT, LPT, and PSO strategies. From this also we have to know that the PSO provides the good result.

TABLE 7.4 RPI Vs FOUR OPERATORS

Number of Operations	RANDOM	SPT	LPT	PSO
10	18.06	13.27	6.93	0.22
20	10.50	12.01	0.70	0.53
30	6.82	11.76	0.58	0.26
40	5.68	7.40	0.51	0.47
50	4.07	7.95	0.57	0.42
60	3.67	4.57	0.53	0.40
70	3.06	4.89	0.15	0.03
80	3.00	3.71	0.27	0.14
90	2.65	3.38	0.19	0.27
100	2.28	2.91	0.38	0.58

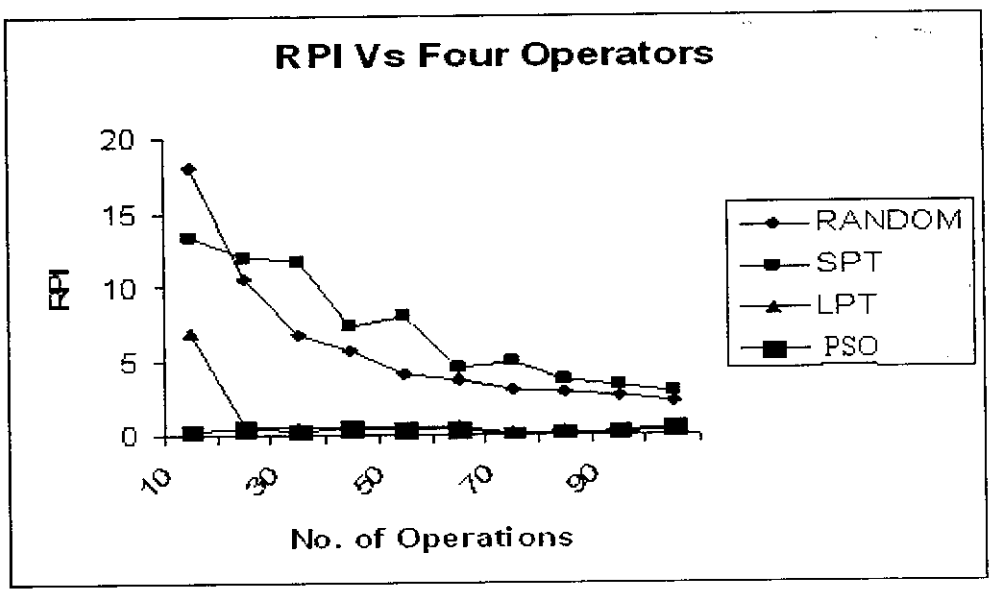


FIGURE 7.4 RPI Vs FOUR OPERATORS

7.5 'n' OPERATIONS Vs FIVE OPERATORS

From the Fig. 7.5 and the Table 7.5 we have to conclude that the PSO provides the better result for five operators' problem rather than the RANDOM, SPT, and LPT.

TABLE 7.5 RPI Vs FIVE OPERATORS

Number of Operations	RANDOM	SPT	LPT	PSO
10	22.89	16.77	7.86	0.31
20	14.05	15.61	1.96	0.22
30	9.10	10.85	1.61	0.26
40	7.41	8.92	0.31	0.26
50	5.39	7.39	0.66	0.31
60	4.97	6.76	0.64	0.36
70	4.06	5.53	0.21	0.20
80	4.25	4.14	0.31	0.26
90	3.43	3.58	0.29	0.18
100	2.79	3.59	0.22	0.26

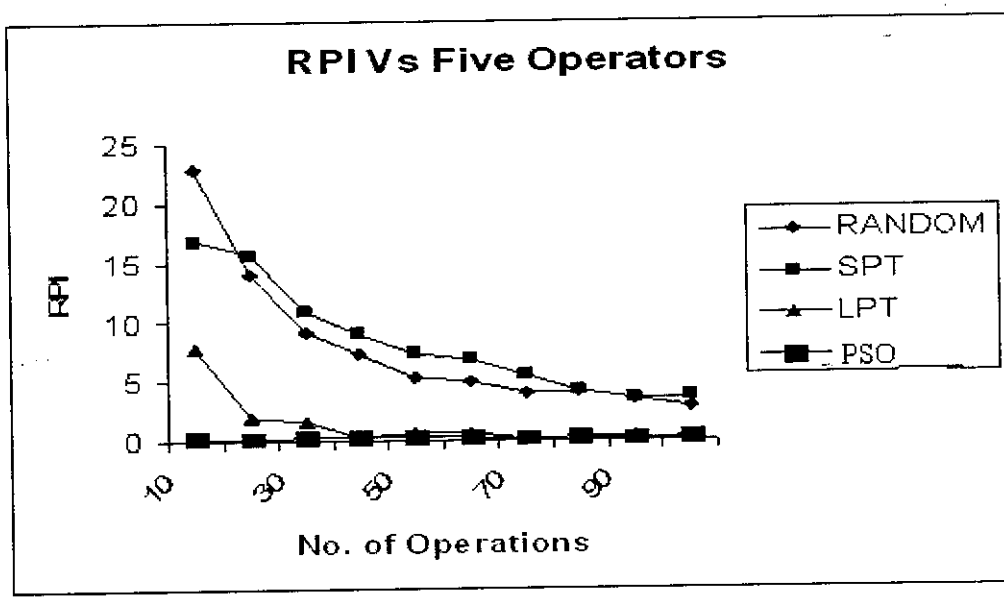


FIGURE 7.5 RPI Vs FIVE OPERATORS

7.6 'n' OPERATIONS Vs SIX OPERATORS

The Fig. 7.6 and Table 7.6 show the performances of RANDOM and PSO for six operators. The proposed PSO produces balanced workloads among the operators for all job ranges. The RPI values produced by the RANDOM show the existence of

TABLE 7.6 RPI Vs SIX OPERATORS

Number of Operations	RANDOM	SPT	LPT	PSO
10	29.20	35.69	9.22	0.46
20	17.60	18.52	1.88	0.25
30	13.11	15.51	0.64	0.26
40	9.33	11.27	063	0.16
50	8.02	7.83	0.67	0.33
60	6.89	8.16	0.34	0.28
70	5.74	6.19	0.84	0.39
80	4.75	6.35	0.43	0.21
90	4.07	5.81	0.51	0.36
100	4.09	4.21	0.34	0.30

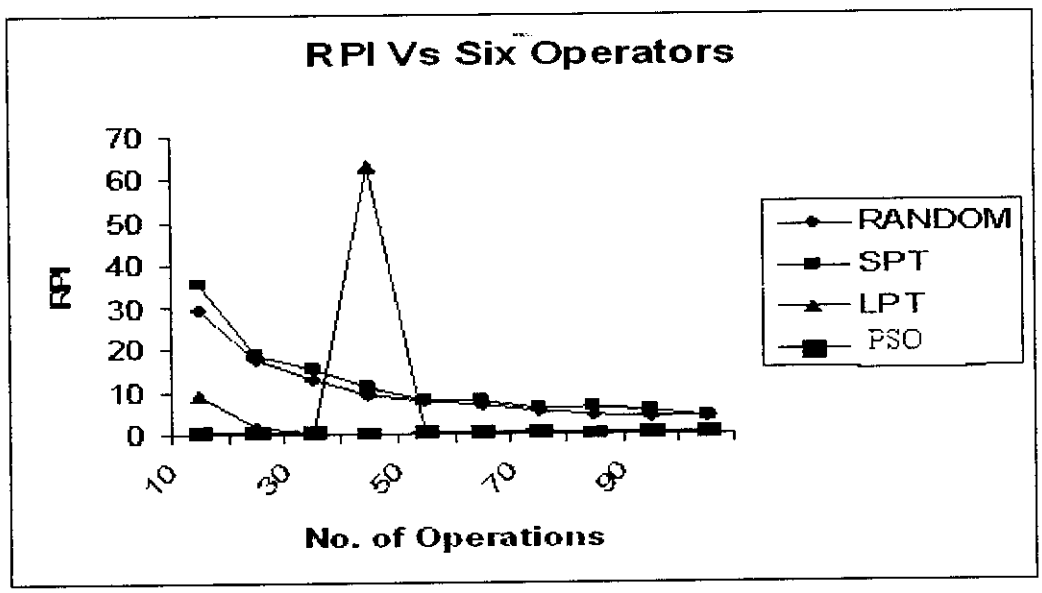


FIGURE 7.6 RPI Vs SIX OPERATORS

CHAPTER 8

RESULTS AND

CHAPTER 8

RESULTS AND DISCUSSIONS

The number of machines may be 'm', and the number of jobs may be 'n' are considered for allocating the 'n' jobs to the 'm' machines in the parallel machine model. The procedure is based on the idea of workload balancing and on balancing the workload among machines. A machine with the lowest workflow is selected for assignment of a new job from the list of unfinished jobs. The relative percentage of imbalance of each machine is found out. Then the average RPI value is calculated. Changing the positions of any two jobs using the formula performs the next iteration. Then finding the Global value and Local value, Global value is the most minimum value and repeating the iteration and then updating the velocity and changing the position. The process is continued until the optimum solution is obtained.

In workflow balancing of parallel machine scheduling with precedence constraints problem the proposed Particles Swarming Algorithm was used to prepare the lists of preceding and succeeding operations separately. The workflow algorithm was used to assign operations to the operators. The average RPI value for PSO is lesser than the current methods.

The proposed PSO produces balances workflow schedule. The smaller RPI values indicate the balanced workflow among the operators. Thus the proposed Particle Swarming performs better than the other heuristics of allocation of assembly operations to the operators in the problem of workflow balancing with precedence constraints.

CHAPTER 9

CHAPTER 9

CONCLUSION

Workflow in parallel machine scheduling using Particles Swarming Algorithm is done. The RPI of various machines is evaluated. Workload of a machine is the sum of processing time of jobs assigned to it. The machine with the maximum workload is the bottle-neck present in the shop floor. This workload needs to be distributed evenly to other machines using heuristics. The balanced workflow increases the system utilizations as well as reduces the completion time of the jobs. The least loaded machine has been selected for assigning a new job from the list of jobs. Then the average RPI value is calculated. The smaller RPI values indicate the better distribution of workflow among machines.

And the workflow balancing in parallel machine scheduling with precedence constraints using PSA was also done. The textile machinery-manufacturing unit considered here produces machines namely carding draw and speed frames. These machines are used to produce yarn from raw cotton. Assembly planning of a speed frame is considered in this project.

The mainframe of the speed frame is erected at one place on the shop floor and other components and sum assemblies are fixed on it along its entire length at various locations. Some components are fixed first because of precedence relationship to accommodate other fitting components.

So there are some precedence operations which must be completed before starting new operations. The operations are allocated such that the workloads of the operators are evenly distributed. An operation may be allocated only after ensuring that all precedence operations have been allocated.

The least loaded operator has been selected for assigning a new operation from the list of unscheduled operations. Then the average RPI value is calculated. The smaller

The heuristics have been coded with the 'C' language and charts are drawn using 'EXCEL' for analysis. The results obtained out of this project shows that Particles swarming algorithm performs better than other heuristics by producing balanced workflow in parallel machine scheduling with precedence constraints and without precedence constraints.

CHAPTER 10

CHAPTER 10

REFERENCES

REFERENCES

1. Anand Kumar.P, and Venketesan.R, (2005), "*Optimization of extrusion process parameter using simulated annealing algorithm*", Industrial Engineering Journal, vol .11, pp. 15 -17.

2. Arunachalam.V.P, Rajakumar.S, and Selladurai.V, (2004), "*Work flow balancing strategies in parallel machines scheduling*", International Journal for Advanced Manufacturing Technology vol. 23, pp 366-374

3. Arunachalam.V.P, Rajakumar.S, and Selladurai.V, (2005), "*Simulation of work flow balancing in assembly shop floor operations*", Journal for Advanced Manufacturing Technology, 16, pp. 265 – 281.

4. Arunachalam.V.P, Rajakumar.S, and Selladurai.V, (2006), "*Work flow balancing in parallel machines through genetic algorithm*", International Journal for Advanced Manufacturing Technology.

5. Arunachalam.V.P, Rajakumar.S, and Selladurai.V, (2006), "*Work flow balancing in parallel machine scheduling with precedence constraints using genetic algorithm*", International Journal for Advanced Manufacturing Technology, vol .17, pp. 239 – 254.

6. Asokan.P, Sachithanandam.M, and Saravanan.R, (2001), "*Comparative analysis of conventional and non-conventional optimization techniques for CNC turning process*", International Journal for Advanced Manufacturing Technology, 17, pp. 471 – 476.

7. Balagurusamy.E, Programming in ANSI C, Second edition.

8. Emin Aydin.M, and Terence C. Fogarty, (2004), "*A simulated annealing algorithm for multi-agent systems: a job-shop scheduling application*", Journal of Intelligent Manufacturing, vol .15, pp. 805 – 814.

9. Erin Aydin.M, and Terence C. Fogarty, "*Modular simulated annealing algorithm for job shop scheduling running on distributed resource machine*" (DRM), White Paper.

10. Esin Onbasoglu and Linet Ozdamar (2001), "*Parallel simulated annealing algorithms in global optimization*", Journal of global optimization,vol-19,page27-50.

11. Esin Onbasoglu and Linet Ozdamar (2001),"*Parallel Simulated Annealing Algorithms in Global Optimization*", Journal of Global Optimization, vol .19, pp.27-50.

12. Margarita Rayes-Sierra and Carlos A.Coello Coello (2006),"*Multi Objective Particle Swarm Optimizers A Survey Of the state- of- the Art*", International Journal of computation intelligence research,vol .2,page 287-308.

13. Michael W. Trosset, (2001), "*What is simulated annealing*"? Optimization and engineering, Vol. 2, pp. 201 – 213.

14. Mohanasundaram.K.M., Suresh.R.K, (2004), "*Simulated annealing algorithm guided by local search for job shop scheduling with minimizing mean flow time objective*, Industrial Engineering Journal, 35, pp. 33 – 38.

15. Nick Vagenas and Tony Nuziale (2001), "*Genetic Algorithm for Reliability Assessment of Mining Equipment*", Journal of quality in maintenance engineering,vol .7,No .4 pp .302-311.

16. Orosz J.E. and S.H.Jacobson (2002), "*Analysis of Static Simulated Annealing Algorithm*", Journal Of Optimization Theory and Application, vol .115, No.1, pp.165-182.

17. Rahimi A. R. -Vahed.S.M.Mirghorbani (2007),"*A Multi –Objective Particle Swarm for a Flow Shop Scheduling Problem*", Journal Comb Optim, vol.13, pp.79-102.

18. Shiu-Hong Choi and James Siu-Lung Lee, (2004),"A sequencing algorithm for makes span minimization in FMS", Journal of manufacturing technology Vol .15, No.3, pp.291-297.