A Study on Flexible Flow Shop and Job Shop Scheduling using Meta-heuristic Approaches

A THESIS SUBMITTED IN FULFILLMENT OF

THE REQUIREMENT FOR THE AWARD OF THE DEGREE

OF

Doctor of Philosophy

IN

MECHANICAL ENGINEERING

BY

Manas Ranjan Singh

(ROLL NO. 510ME805)

NATIONAL INSTITUTE OF TECHNOLOGY

ROURKELA - 769008, INDIA

August – 2014

CERTIFICATE

This to certify that the thesis entitled "**A Study on Flexible Flow Shop and Job Shop Scheduling using Meta-heuristic Approaches**" being submitted by **Manas Ranjan Singh** for the award of the degree of Doctor of Philosophy (Mechanical Engineering) of NIT Rourkela, is a record of bonafide research work carried out by him under our supervision and guidance. Mr. Manas Ranjan Singh has worked for more than three years on the above problem at the Department of Mechanical Engineering, National Institute of Technology, Rourkela and this has reached the standard fulfilling the requirements and the regulation relating to the degree. The contents of this thesis, in full or part, have not been submitted to any other university or institution for the award of any degree or diploma.

Dr. Siba Shankar Mahapatra
Professor
Department of Mechanical Engineering
National Institute of Technology,
Rourkela

Dr. Ratikanta Mishra
Professor
Department of Mechanical Engineering
Silicon Institute of Technology,
Bhubaneswar

Place: Rourkela

Date:

ACKNOWLEDGEMENT

This thesis is a result of research that has been carried out at **National Institute of Technology, Rourkela**. During this period, I came across with a great number of people whose contributions in various ways helped my field of research and they deserve special thanks. It is a pleasure to convey my gratitude to all of them.

In the first place, I would like to express my deep sense of gratitude and indebtedness to my supervisors **Prof. S.S. Mahapatra** and **Prof. R. K. Mishra** for their advice and guidance from early stage of this research and providing me extraordinary experiences throughout the work. Above all, they provided me unflinching encouragement and support in various ways which exceptionally inspire and enrich my growth as a student, a researcher and a scientist.

I specially acknowledge **Prof. S.S. Mahapatra** for his advice, supervision and crucial contribution as and when required during this research. His involvement with originality has triggered and nourished my intellectual maturity that will help me for a long time to come. I am proud to record that I had opportunity to work with an exceptionally experienced scientist like him.

I am grateful to **Prof. S.K. Sarangi**, Director, **Prof. S.S. Mahapatra**, Head of Mechanical Engineering Department, and **Prof. K.P. Maity**, former Head of Mechanical Engineering Department, National Institute of Technology, Rourkela, for their kind support and concern regarding my academic requirements.

I express my thankfulness to the faculty and staff members of the Mechanical Engineering Department for their continuous encouragement and suggestions. Among them, **Sri P. K. Pal** deserves special thanks for his kind cooperation in non-academic matters during the research work.

I am indebted to **Dr. Saurav Dutta, Dr. Gouri Shankar Beriha, Mr. Chinmaya Prasad Mohanty, Mr. Swayam Bikash Mishra** and **Mr. Suman Chatterjee** for their support and co-operation which is difficult to express in words. The time spent with them will remain in my memory for years to come.

Thanks are also due to my colleagues at **Silicon Institute of Technology, Bhubaneswar** for their whole hearted support and cooperation during the course of this work.

My parents deserve special mention for their inseparable support and prayers. They are the persons who show me the joy of intellectual pursuit ever since I was a child. I thank them for sincerely bringing up me with care and love.

The completion of this work came at the expense of my long days of absence from home. Words fail me to express my appreciation to my wife **Nibedita** and my little angel **Anushka** for their understanding, patience and active cooperation throughout the course of my doctoral dissertation. I thank them for being supportive and caring.

Last, but not the least, I thank the one above all of us, the omnipresent God, for giving me the strength during the course of this research work.

Manas Ranjan Singh

ABSTRACT

Scheduling aims at allocation of resources to perform a group of tasks over a period of time in such a manner that some performance goals such as flow time, tardiness, lateness, and makespan can be minimized. Today, manufacturers face the challenges in terms of shorter product life cycles, customized products and changing demand pattern of customers. Due to intense competition in the market place, effective scheduling has now become an important issue for the growth and survival of manufacturing firms. To sustain in the current competitive environment, it is essential for the manufacturing firms to improve the schedule based on simultaneous optimization of performance measures such as makespan, flow time and tardiness. Since all the scheduling criteria are important from business operation point of view, it is vital to optimize all the objectives simultaneously instead of a single objective. It is also essentially important for the manufacturing firms to improve the performance of production scheduling systems that can address internal uncertainties such as machine breakdown, tool failure and change in processing times. The schedules must meet the deadline committed to customers because failure to do so may result in a significant loss of goodwill. Often, it is necessary to reschedule an existing plan due to uncertainty event like machine breakdowns. The problem of finding robust schedules (schedule performance does not deteriorate in disruption situation) or flexible schedules (schedules expected to perform well after some degree of modification when uncertain condition is encountered) is of utmost importance for real world applications as they operate in dynamic environments.

According to the shop environments, the shop scheduling problems can be classified as flow shop, flexible flow shop, job shop, and flexible job shop scheduling. Shop scheduling problems are combinatorial optimization class of problems which means searching for an optimal solution in a finite set of potential solutions. Exact or complete algorithms guarantee to find an optimal solution for every finite size instance of a combinatorial optimization problem in bounded time. The typical combinatorial problems like the shop scheduling problem are usually NP-hard i.e. hardly any algorithm exist can solve the problem in polynomial time. Therefore, exact algorithm needs unexpected computation time leading to impractical computational burden for large scale application. Despite the relative success of exact algorithms and heuristic methods, they are still incapable of solving medium and large instances and too complex for real world problems. Therefore, non-exact but efficient heuristics must be explored to find the solution in a reasonable period of time. Efficient meta-heuristics procedures like tabu

search (TS), ant colony optimization (ACO), artificial immune system (AIS), simulated annealing (SA), particle swarm optimization (PSO) and genetic algorithm (GA) have been proposed to find an approximate solution close to the optimum with considerably less computational time in various engineering applications. Extensive literature review suggests that the flexible flow-shop scheduling problems (FFSP) and flexible job-shop scheduling problems (FJSP) are least explored. As FFSP and FJSP are more complex problems than the FSP and JSP problems, it encourages the researchers to apply the meta-heuristic techniques which will provide high quality solutions in a reasonable computational time.

PSO is an effective algorithm which gives quality solutions in a reasonable computational time and requires less number parameters to be tuned in comparison to other evolutionary approaches. However, PSO has an inherent drawback of getting trapped at local optimum due to large reduction in velocity values as iteration proceeds and poses difficulty in reaching at best solution. This drawback can be effectively addressed using quantum-behaved particle swarm optimization (QPSO) due to its advanced global search ability. Mutation, a commonly used operator in genetic algorithm, can be introduced in QPSO so that premature convergence can be avoided. Logistic mapping can be used to generate chaotic numbers instead of random numbers to improve the solution diversity.

In this dissertation work, a novel particle swarm optimization (PSO) and quantum particle swarm (QPSO) optimization algorithm have been proposed for solving the single objective as well as multi-objective scheduling for flexible flow shop and job shop scheduling problems. Methodology for obtaining robust schedule is proposed to deal with uncertain situation in flexible flow shop and job shop scheduling. It is demonstrated that solution quality improves when PSO and QPSO algorithms are embedded with chaotic numbers and mutation.

Keywords: Flexible flow shop; Flexible job shop; PSO; QPSO; Multi-objective optimization; MOPSO; Makespan; Flow time; Tardiness, Chaotic Number; Mutation; Maximum deviation theory

CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSSARY OF TERMS

%PD	Percentage deviation
2SGA	Two-stage genetic algorithm
ACO	Ant colony optimization
AIS	Artificial immune system
ANOVA	Analysis of variance
APD	Average percentage deviation
B&B	Branch and bound
BD	Break down
BD1	Low disruption level
BD2	High disruption level
CCQGA	Competitive co-evolutionary quantum genetic algorithm
CDDS	Climbing depth-bound discrepancy search
CDS	Climbing discrepancy search
CMOEA	Multi-objective evolutionary algorithm
CPM	Critical path method
DDS	Depth-bounded discrepancy search
DM	Diversification matrix
DPSO	Discrete particle swarm optimization
EDD	Earliest due date
EGA	Efficient genetic algorithm
EH	Exchange heuristic
FFSP	Flexible flow-shop scheduling problem
First-in first-out	FIFO
FJSP	Flexible job shop scheduling problem
FSP	Flow-shop scheduling problem
GA	Genetic algorithm
GLS	Genetic local search
GLS	Guided local search
GS	Global selection
GT	Giffler and thompson
HFSP	Hybrid flow-shop scheduling
hGA	Hybrid genetic algorithm

IA	Immune algorithm
IE	Enhancement scheme
IGA	Integrated genetic algorithm
IP	Integer programming
JSP	Job shop scheduling problem
KBACO	Knowledge-based ant colony optimization algorithm
LB	Lower bound
LEGA	Learnable genetic architecture
LPT	Longest processing time
LS	Local selection
MA	Memetic approach
MADM	Multi-attribute decision making
MDT	Maximum deviation theory
MID	Mean ideal distance
MILP	Mixed integer linear programming
MIP	Mixed-integer programming model
MOGA	Multi-objective genetic algorithm
MOO	Multi-objective optimization
MOPSO	Multi-objective particle swarm optimization
MOSP	Multi-objective scheduling problem
MTTR	Mean-time-to-repair
NP-hard	Non-deterministic Polynomial-time hard
NSGA-II	Non-dominated sorting genetic algorithm ii
OGA	Order based genetic algorithm
PERT	Program evaluation and review technique
PMUT	Probability of mutation
PSO	Particle swarm optimization
PVNS	Parallel variable neighborhood search
QPSO	Quantum particle swarm
RAI	Recursive arc insertion
RAS	The rate of achievement to two objectives simultaneously
RCPSP	Resource constrained project scheduling problem
RKGA	Random keys genetic algorithm
RM	Robustness measure

RNGs	Random number generators
SA	Simulated annealing
SA	Simulated annealing
SA	Simulated annealing
SDE	Self-adaptive differential evolution
SJSP	Stochastic job-shop scheduling problem
SNS	Spread of non-dominance solutions
SPEA	Strength pareto evolutionary algorithm
SPT	Shortest processing time
SSDST	Sequence dependent setup times
TFN	Triangular fuzzy numbers
TS	Tabu search
TSP	Travel salesman problem
VND	Variable neighborhood descent

CHAPTER 1

Background and Motivation

1.1 Introduction

Scheduling is one of the important decision making processes in both manufacturing and service industries for improving organizational effectiveness and customer satisfaction. Scheduling deals with the allocation of operations on machines (i.e. a sequence of operations on machines) in such a manner that some performance goals such as flow time, tardiness, lateness and makespan can be minimized [1]. Effective scheduling has become a basic necessity for survival and business growth of a firm in the marketplace. Nowadays, the manufacturing industry is experiencing some new challenges such as global competition, shorter product life cycles, customized products and market demand changes etc. In order to sustain in the competitive environment, it is vital for the manufacturing companies to improve the performance of their production scheduling systems under increasing market fluctuations (e.g. rush orders and job cancellation) and internal uncertainties in the manufacturing process (e.g. machine breakdown, tool failure, and change of processing times). Scheduling is a process by which limited resources are assigned over time among parallel or sequential activities. Such situations are found routinely in factories, publishing houses, shipping, hospitals, airports etc Scheduling finds extensive applications in manufacturing, transportation, communication, health care, space exploration, education, network distribution etc. [2] Good scheduling algorithms can lower the production cost in a manufacturing process so as to enable the company to remain competitive. In general, intelligent scheduling methods are needed to assign activities to processors (machines) when faced with limited execution time and scarce resources. Organizations must meet the deadline committed to customers because failure to do so may result in a significant loss of goodwill. The organizations need to schedule activities in such a manner that available resources should be used in an efficient manner. There are many different performance measures to optimize a scheduling problem. One objective may be the minimization of the completion time of the last job and another may be the minimization of the number of jobs completed after their respective due dates [3].

Shop Scheduling problems are typical representatives of combinatorial optimization class of problems which means searching for an optimal solution in a finite set of potential solutions. A wide variety of scheduling problems have been identified and even wider range of solution methodologies has been proposed. At the very beginning, research has been focused on exact methods. Exact or complete algorithms guarantee to find an optimal solution for every finite size instance of a combinatorial optimization

problem in bounded time. The typical combinatorial problems like the shop scheduling problem are usually non-deterministic polynomial-time hard (NP-hard) i.e. no algorithm exist to solve those problems in polynomial time [4]. Therefore, exact algorithm needs unpredictably computation time in most cases leading to impractical computational burden for large scale application. The most common exact method for scheduling problem are branch and bound, branch and cut, Lagrangian relaxation and dynamic programming. Due to lack of computational resources and the need to solve large scale scheduling problems, it has been realized that exact methods are impractical and thus research focused on development of heuristic methods to obtain the approximate solution [5, 6, 7]. A heuristic method is based on a specified thumb rule for a particular problem.

1.2 Importance of scheduling in a manufacturing system

The current environment in manufacturing companies is characterized by massive competition faced by market and customers' requirement and expectations. These characters are increasing spontaneously high in terms of quality, cost and delivery time. Generally, the firm performance is built in two dimensions [8].

- Technological dimension
- Organizational dimension

The role of the technological dimension is to develop the inherent performance of marketed products in order to satisfy the requirement of quality and lower cost of the product. In this regard, it must be noted that the rapid technological growth for these products forced the companies to opt for mass production. This needs a flexible and progressive production system capable of adapting to market demand and needs quickly and efficiently.

An organizational dimension intends to performance improvement in terms of production cycle times, expected delivery date, inventory and work in process management etc. Therefore, companies must have powerful method and tools at their disposal for production planning and control.

To achieve these goals, an organization normally implements a number of functions including scheduling with variety of products, processes and production levels, production planning, material and capacity planning etc. for better coordination to increase productivity and minimize operation costs. A production schedule detects the control over the release of jobs to the shops, ensure required raw materials are ordered in time and find strategies for resource conflicts. A production schedule can determine

whether delivery promises can be met and identify time period available for preventive maintenance [9]. In manufacturing environment, all jobs or tasks are associated with a due date. These jobs have to be processed on the machines in a given order or sequence. Sometimes the processing of jobs may be delayed because certain machines are busy; preemptions may occur when high priority jobs arrive; unforeseen events such as machine breakdowns or uncertain processing times may occur. The release times, routings and processing times of the jobs are stochastic parameters and not known in advance [10] .

1.3 Classification of scheduling problems

Scheduling plays an important role in most manufacturing and service systems as well as in most information processing environments. Schedules are divided into two classes, feasible schedule and infeasible schedules. A feasible schedule is a schedule in which all tasks meet their deadlines with a specified constraint (resource availability constraints, precedence constraints etc.).The infeasible schedules are those schedules that violate some or all constraints. Scheduling also derives its importance from the two following different considerations:

- Ineffective scheduling results in poor utilization of available resources. A noticeable indication is the idleness of facilities, human resources and apparatus waiting for orders to be processed. As a result, the cost of production increases.
- Poor scheduling normally creates delays in the flow of some orders through the systems.

The major scheduling models are categorized by specifying the resource configuration and the nature of the tasks. For instance, a job may need processing on one machine or several machines. If it contains one machine, jobs are likely to be processed on a single stage whereas jobs may be processed on multiple machines in multiple stages. If all jobs to be scheduled are available at the beginning of the scheduling process, the problem is known as static scheduling. If the set of jobs to be processed is continuously changing over time, the problem is known as dynamic scheduling. The release times, routings and processing times of the jobs are stochastic parameters and not known in advance .Generally, static scheduling is easily controllable than dynamic scheduling and has been studied extensively. When all parameters are known with certainty, the scheduling model is called deterministic. On the other hand, the scheduling is called stochastic when uncertainty exists for any one of the scheduling parameters [11].

Basically, in scheduling problems, it is assumed that the setup times are negligible or as a part of the processing time. But if the set up time is considered then it is known as scheduling with sequence dependent setup times (SSDST). In some scheduling problems, buffer at some stages of the production process is considered and a job spends some time in the buffer between two consecutive operations. This important class of scheduling problems is characterized by the presence of no-wait or blocking constraints between consecutive operations of the jobs. A no-wait constraint occurs when two consecutive operations must be performed without any interruption.

1.4 Types of scheduling

Scheduling has a very wide area of application. Almost every service provider and manufacturer experiences a kind of scheduling problem. For instance, airports have landing and take-off sequencing problem, airline operator having timetabling and routing problems, a university must have class and examination scheduling, a manufacturer experiences several shop problem in order to meet customer demand [12]. The taxonomy of scheduling problems are as follows:

- Project Scheduling
- Single machine scheduling
- Flow shop scheduling
- Job shop scheduling
- Flexible flow shop scheduling
- Flexible job shop scheduling

1.4.1 Project scheduling

Project Scheduling mostly deals with the sequencing of activities subject to precedence constraints and allocation of resources to these activities in a project. The project scheduling problem is similar to parallel machine problem that has an infinite number of machines. The objective is to minimize the makespan. The methods used for project scheduling are critical path method (CPM) and program evaluation and review technique (PERT).CPM is used for projects with deterministic activity duration while PERT is used for projects with probabilistic activities. The project scheduling is known as resource constrained project scheduling problem (RCPSP) while it involves executing a group of activities limited by constraints. For accomplishment of each activity, a predefined amount of resources which are available in limited quantities per unit time is needed.

1.4.2 Single machine scheduling

When the jobs to be scheduled pass through a single machine or facility, the scheduling problems are called single machine scheduling problems. The single machine scheduling problem consists of 'n' jobs with a single operation on each of the jobs. The objective of single machine scheduling is to find the best sequence of the jobs such as to optimize the objective function.

1.4.3 Flow shop scheduling

The flow-shop scheduling problem (FSP) consists of 'm' machines and 'n' jobs. The scheduler's objective is to find an optimal sequence of 'n' jobs on 'm' machines. All 'm' machines are situated in a defined series. All 'n' jobs have to be processed on each machine. The routing of the jobs through the different machines is same for all jobs (unidirectional flow). Once a job is completed on one machine, it is placed into the queue of the next machine in series. Normally, jobs are removed from the queue on a first-in, first-out (FIFO) basis but this can be modified to fit the needs of the problem such as higher priority jobs could be bumped to the front of the queue. Figure 1.1 is an example of the flow-shop example where each job flows in an orderly fashion from one machine to the next. In the example, there are 'j' number of jobs and 'm' number of machines where 'J_1' is the first job and 'M_1' is the first machine.

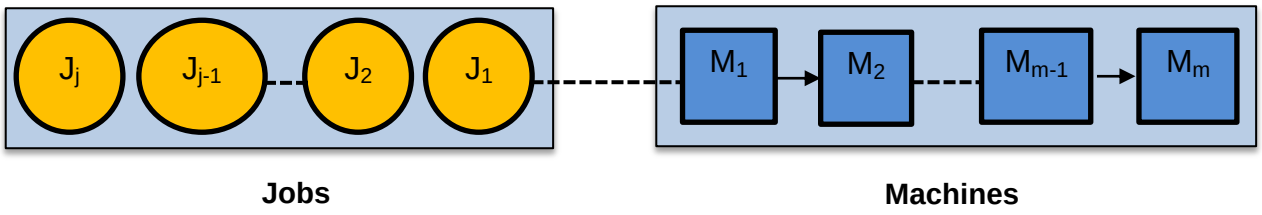


Figure 1.1 Generalized flow-shop problem.

1.4.4 Job shop scheduling

A classical job shop scheduling problem (JSP) deals with a set of 'n' jobs to be processed by a set of machines. Each job is processed on machines in a given order with a given processing time and each machine can process only one job at a time. The scheduler's objective is to find an optimal ordering of all the jobs with respect to their varied routing requirements through the machines. Each job must visit the machine in a sequence but the difference with the flow shop is that the sequence may be different for each job (multidirectional flow). Figure 1.2 shows an example of the job-shop problem where each job follows its own path through the various machines. The machines in the example are labeled as $M_{x,y}$ where the 'x' represents the job number and 'y' represents

the location of the machine with respect to the other machines. Machine $M_{1,3}$ be used for job 1 before $M_{1,4}$ as it is the third machine in the route and $M_{1,4}$ is the fourth machine. All jobs may not require the same number of machines. In order to show that each route may have a different number of machines, each route ends with a different variable for 'y'. Each row in the Figure 1.2 represents the ordering of a job with respect to the same 'm' number of machines.

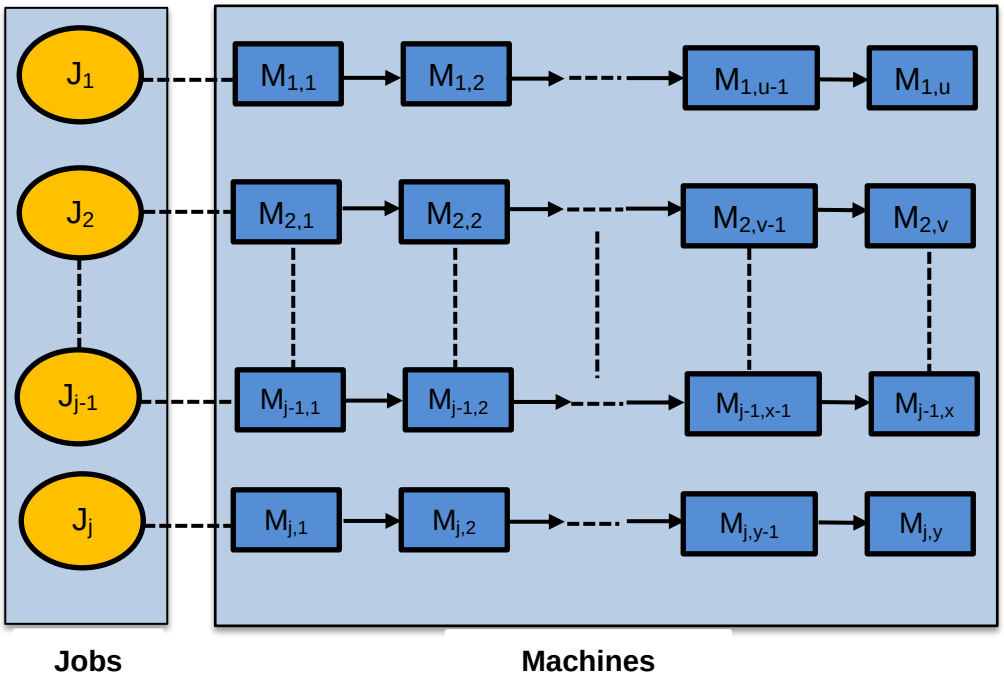


Figure 1.2 Generalized job-shop problem

1.4.5 Flexible flow shop scheduling

The flexible flow-shop scheduling problem (FFSP) is an extension of the flow-shop problem where parallel machines are combined with the flow-shop problem. Therefore, instead of m machines in series like flow-shop, there is a series of 'm' stages with each stage having one or more machines. The scheduler's objective is to find an optimal ordering through m stages for the 'n' jobs by taking advantage of the multiple machines in one or more stages. All the jobs still have to be processed by one machine in each stage [8]. Figure 1.3 represents an example of the flexible flow-shop problem where multiple machines can do the operation in order to limit bottlenecks in the process. The machines have a label, $M_{x,y}$ where x signifies the stage in which the machine belongs to and y is the machine number in that stage. Hence, $M_{1,h}$ is the 'h' machine in stage '1'. Note that the stages may have a different number of machines.

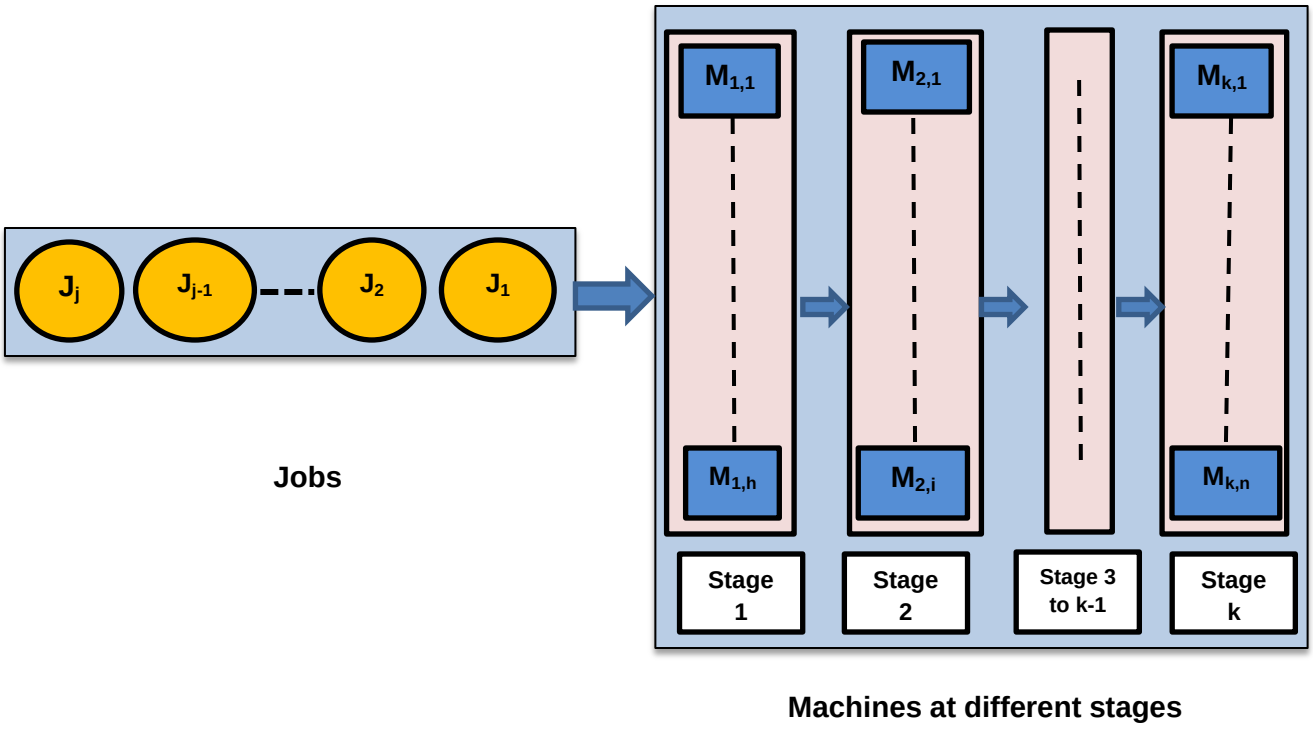


Figure 1.3 Diagram of the generalized flexible flow-shop problem

1.4.6 Flexible job shop scheduling

The flexible job shop scheduling problem (FJSP) is an extension of the job shop problem (JSP) where operations are allowed to be processed on any among a set of available machines at a facility. In the flexible job shop, parallel machines are combined with the job-shop problem having a total of 'P' possible work centers. Each work center consists of a set of 'm' machines from which one machine is chosen to perform the task or operation. FJSP is considered to be more difficult than the classical JSP because it contains an additional problem of assigning operations to machines. This model is particularly useful when it is employed to overcome bottlenecks by adding machines in parallel where slowdowns occur in the process. Figure 1.4 shows a diagram of the flexible job-shop where work centers have the parallel machines. The machines are labeled as $M_{x,y1,y2}$ where 'x' represents the job number, 'y_1' represents the work center number, and 'y_2' represents the number of machines in a particular work center. Notation $M_{4,3,5}$ represents the fifth machine in third stage of the route for job 4.

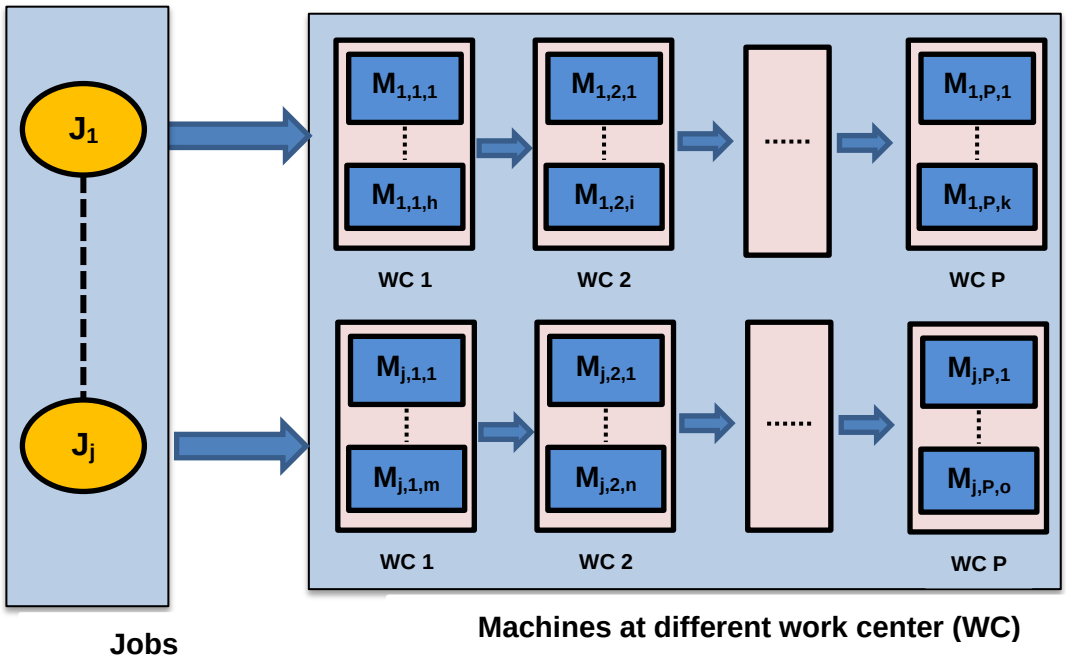


Figure 1.4 Diagram of the generalized flexible job-shop problem

1.5 Complexity of the scheduling problem

Computational complexity of a problem is the maximum number of computational steps needed to obtain an optimal solution. The notion of complexity refers to the computing effort required by a solution algorithm. Computing effort is described by order-of-magnitude notation. Suppose a particular algorithm is used to solve a problem of size 'n' (n denotes the amount of information needed to specify the problem). The number of computations required by the algorithm is typically bounded by a function of 'n'. If the order of magnitude of this function is polynomial as 'n' gets large then the algorithm is polynomial. For instance, if the function has order of magnitude '$n^{2'}$, denoted '$O(n^2)$' then the algorithm is polynomial. On the other hand, if the function is '$O(2^n)$' then the algorithm is non-polynomial (exponential) [13]. Based on complexity of the problem, all problems can be classified into two classes called 'P' and 'NP' in the literature. The class 'P' consists of the problems for which the execution time of the solution algorithm grows polynomially with the size of problem. The time taken to solve a problem belonging to the NP class grows exponentially. In actual practice, the algorithms are preferred whose execution time grows polynomially because it gets a solution in a reasonable time. Unfortunately, most of the practical scheduling problems belong to the non-deterministic polynomial-time hard (NP-hard) [14]. Based on this complexity concept, it is concluded that one may not be able to find optimal solutions with available techniques to solve large versions of an NP-hard problems which are applied to many scheduling problems.

Normally, these problems are too difficult to be solved exactly within a reasonable amount of time and heuristics become the method of choice.

1.6 Terminologies

While dealing with job attributes for scheduling problems, it is useful to distinguish between information that is known in advance and information that is generated as the result of scheduling decisions. Information that is known in advance serves as input to the scheduling process. The basic pieces of information that help to describe jobs in the scheduling problems are [6]:

Processing time (p_j) : The amount of processing time required by job j

Release date (r_j) : The time at which job j is available for processing

Due date (d_j) : The time at which the processing of job j is due to be Completed

Completion time (C_j) : The time at which the processing of job j is finished

Flow-time (F_j) : The time job j spends in the system: $F_j = C_j - r_j$

Lateness (L_j) : The amount of time by which the completion time of job j exceeds its due date $L_j = C_j - d_j$

These two quantities (flow time and lateness) reflect two kinds of service. Flow time measures the response of the system to individual demands for service and represents the interval a job waits between its arrival and its departure. (This interval is sometimes called the turnaround time). Lateness measures the conformity of the schedule to a given due date and takes on negative values whenever a job is completed early. Negative lateness represents earlier service than requested; positive lateness represents later service than requested.

Tardiness (T_j): The lateness of job j if it fails to meet its due date, or zero otherwise:

$$T_j = \max\{0, L_j\}$$

Schedules are generally evaluated 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

Total flow time: $\mathrm{F} = \sum_{j=1}^{n} \mathrm{F_j}$

Total tardiness: $\mathrm{T} = \sum_{j=1}^{n} \mathrm{T_j}$

Maximum flow time: $\mathrm{F_{Max}} = \max_{1 \le j \le n} \{\mathrm{F_j}\}$

Maximum tardiness: $T_{Max} = \underset{1 \le j \le n}{Max} \{T_j\}$

Number of tardy jobs: $U = \sum_{j=1}^{n} \delta(T_j)$ where $\delta(x) = 1$ if $x > 0$ and $\delta(x) = 0$ otherwise

Maximum completion time: $C_{Max} = \underset{1 \le j \le n}{Max} \{C_j\}$

1.7 Performance measures in scheduling

It is not easy to state objectives in scheduling as they are numerous, complex and often conflicting. A measure of performance is said to be regular if it is a non-decreasing function of job completion times and the scheduling objective is to minimize the performance measure. A large number of scheduling problems have been studied with regular performance measures. The most widely considered regular performance measures are [15]

- Makespan: the objective is to minimize the maximum completion time of the schedule.
- Mean flow time: the objective is to minimize the average time spent by a job in the system. Flow time is defined as the elapsed time since the job is ready to be processed until it has finished
- Total tardiness: the objective is to minimize the summed lateness of all jobs in the system. Lateness is defined as how much later a job has finished after its deadline.

Makespan and total flow time are related to maximizing system utilization and work in process inventory while the tardiness is related to job due dates. Scheduling with makespan criteria is very important in order to increase the productivity and maximum utilization of resources. In modern manufacturing and operations management, on time delivery is a significant factor towards the stress of competition on the markets i.e industry has to offer a great variety of different and individual products while customers are expecting ordered goods to be delivered on time. As lack of success in meeting the due dates can result in the loss of customer and market competitiveness. Hence, scheduling problems with due date related objectives have attracted increasing attention from managers and researchers. In today's competitiveness environment, cost of production must be reduced in order to survive in this dynamic environment which has been done by effective utilization of all the resources and production in shorter time to increase the productivity also simultaneously considering due dates of the job. Most of the research reported in the literature is focused on the single objective case of shop scheduling problems, in which the makespan is minimized. Some researchers have

investigated multi-objective perspective of scheduling problems but the amount of literature in this area is still scarce compared to the single objective case. In this dynamic and conflicting environment, industries have to achieve number of performance measures for survival and hence scheduling system with multi-objective performance measures have given due attention since 1980. Various researchers have considered multi-objective nature of scheduling problem but restricted to two or three criteria of performance measures. Despite the tremendous effort devoted to development of production scheduling techniques, few successful applications in solving real-world scheduling problems have been reported. Most of the research works tend to be based on highly unrealistic assumptions, implementation them is almost infeasible to deal with scheduling problems in real world manufacturing environments, which are complex, dynamic, and stochastic and is subjected to various disruptions due to a wide range of stochastic uncertainties. For example, resource shortages and machine breakdowns can delay a schedule's completion time. Among all the uncertain events, machine breakdown is one of the significant disruptions in shop scheduling problems. In addition to normal performance measures such as makespan, flow time, and tardiness, two more measures known as robustness (the schedule performance does not deteriorate in disruptions situation) and stability (the schedule which does not deviate the completion time of the unaffected operations from the original schedule in a disrupted situation) are considered in the uncertain environments [16].

1.8 Need for research

Despite the relative success of exact algorithms and heuristic methods, they are still incapable of solving medium and large instances and are too complex for real world problems. It is essential to study non-exact but efficient heuristics [17, 18, 19]. Therefore, efficient meta-heuristics procedures like tabu search (TS), ant colony optimization (ACO), artificial immune system (AIS), simulated annealing (SA), particle swarm optimization (PSO) and genetic algorithm (GA) have been proposed to find an approximate solution close to the optimum with considerably less computational time [20,21,22]. It has been found from the literature that the FFSP and FJSP problems are least explored to the field of research. As FFSP and FJSP are more complex problems than the FSP and JSP problems, encourage the researchers to apply the meta-heuristic techniques which will provide high quality solutions in a reasonable computational time.

The efficiency of a meta-heuristics algorithm depends on two goals such as exploration and exploitation. Exploration ensures every part of the solution domain is

searched enough to provide a global optimum solution. Exploitation concentrates the search effort around the best solutions found so far by searching the neighborhoods to reach at better solutions. PSO can update its particle's positions according to individual's memory (personal best) and swarm's information (global best) in an iteration leading to efficient exploration and exploitation capability. With the collective intelligence of the particles, the whole swarm can converge to an optimum or near-optimum solution. Such type of collective intelligence is hardly incorporated in GA. PSO is not only flexible but also possesses a well-balanced method than GA to improve and adjust to the global and local exploration and exploitation abilities within a short computation time. PSO is more computationally elegant than the GA as it uses less number of controlling parameters. These characteristics make PSO highly reasonable to be used for solving single objective and also multi-objective optimization problems. Due to the simple concept, easy implementation, and rapid convergence, PSO has gained much attention and been successfully applied to a wide range of applications such as power and voltage control, mass spring system, supply chain network and vehicle routing problems [23,24,25].

In this dissertation work, a novel particle swarm optimization (PSO) and quantum particle swarm (QPSO) optimization algorithm have been proposed for solving the single objective as well as multi-objective problems and also proposed a robust schedule in an uncertain situation for the flexible flow shop and flexible job shop scheduling. Chaotic numbers are used instead of random numbers to improve the solution diversity. In addition, mutation, a popular operator in genetic algorithm, is embedded in the standard PSO and QPSO algorithm to escape from local optima.

1.9 Research objectives

This research work aims at the development of artificial intelligence techniques using general proposed meta-heuristics to solve flexible flow shop and job shop scheduling problem

1. To develop an enhanced scheduling method embedding chaotic numbers and mutation operator for flexible flow shop and job shop scheduling problem with no-wait processing condition using meta-heuristic procedures, especially by particle swarm optimization and quantum particle swarm optimization for minimizing the makespan.
2. To design a multi-objective framework by considering the makespan and the robust measures simultaneously to generate the robust schedules that minimizes

the effect of machine breakdowns in the overall performance for the flexible flow shop and job shop scheduling problem.

3. To propose and implement a novel multi-objective particle swarm optimization (MOPSO) technique for solving the flexible flow shop and job shop scheduling problem with an objective to minimize makespan, mean flow time and mean tardiness simultaneously with the goal of finding approximations of the optimal Pareto front.

1.10 Organization of thesis

Seven chapters presented in this thesis are organized as follows:

1.10.1 Chapter 1: Background and motivation

To meet the above objectives, the thesis is organized into seven chapters including Chapter 1. This chapter introduces the concept of scheduling including basic applications and solution methodology. This chapter provides the justification, motivation and need for present research work.

1.10.2 Chapter 2: Literature review

The purpose of this chapter is to review related literature so as to provide background information on the issues to be considered in the thesis and to emphasize the relevance of the present study. Literature review provides a summary of the base knowledge already available about job scheduling. This chapter adopts an exploratory approach for identifying and examining a diverse range of issues in job scheduling. The chapter highlights the solution methodology and problems associated with various aspects of scheduling problem. Finally, the chapter is concluded by summarizing the heuristics/dispatching rules and meta-heuristic approaches proposed in the literature and possible literature gap so that relevance of the present study can be emphasized

1.10.3 Chapter 3: Flexible flow shop scheduling

This chapter briefly discusses the flexible flow shop scheduling. The evolutionary technique namely particle swarm optimization (PSO) and quantum particle swarm optimization (QPSO) has been adopted to determine the optimum solution. The benchmark instances are evaluated using makespan and compared with other solution approaches. The chaotic numbers are used instead of random numbers to improve the solution diversity. The mutation, a popular operator in genetic algorithm, is embedded in the standard PSO algorithm to escape from local optima.

1.10.4 Chapter 4: Flexible job shop scheduling

This chapter briefly discuss about the flexible job shop scheduling. The proposed particle swarm optimization (PSO) and quantum particle swarm optimization (QPSO) has been implemented to determine the optimum solution.

1.10.5 Chapter 5: Flexible flow shop and job shop scheduling with machine breakdown

This chapter addresses to produce a robust schedule for a flexible flow shop and job shop scheduling problem with random machine breakdown. A multi objective framework based on particle swarm optimization (PSO) and quantum particle swarm optimization (QPSO) is proposed to generate the robust schedule by minimize the makespan and the robust measure simultaneously. An experimental study and analysis of variance (ANOVA) is conducted to study the effect of different proposed robustness measures on the performance under uncertainty situation.

1.10.6 Chapter 6: Multi-objective flexible flow shop and job shop scheduling problem

In this chapter, a novel multi-objective particle swarm optimization (MOPSO) technique is proposed and implemented for solving the flexible flow shop scheduling problem (FFSP) and flexible job shop scheduling problem (FJSP) with an objective to minimize makespan, mean flow time and mean tardiness with the goal of finding approximations of the optimal Pareto front and is compared with non-dominated sorting genetic algorithm II (NSGA-II) in terms of four performance metrics. The Pareto-optimal solutions obtained through MOPSO have been ranked by the composite scores obtained through maximum deviation theory (MDT) to avoid subjective-ness and impreciseness in the decision making.

1.10.7 Chapter 7: Discussion and Conclusion

This chapter presents the summary of the results, recommendations and scope for future work in the direction of job scheduling. It also discusses the specific contributions made in this research work and the limitations there in. This chapter concludes the work covered in the thesis with implications of the findings and general discussions on the area of research.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Production scheduling concerns with allocation of finite manufacturing resources such as manpower, equipment, and tools to perform a collection of tasks with optimizing one or more objectives [1]. A variety of scheduling problems like flow shop, job shop, flexible flow shop, flexible job shop etc has been recognized from real-world manufacturing environment. However, most academic research mainly emphases on some classical ones (flow shop and job shop scheduling) [25, 26].The shop scheduling, one of the most classical and challenging scheduling problems, has interested the researchers from both academia and industry. The basic shop scheduling model comprises a set of jobs and machines and deals with determining an optimal or near optimal job sequence on each machine under some constraints. All the shop scheduling problems belong to the NP-hard class [10, 29]. These problems become much more difficult to solve when multiple performance measures and stochastic environment are considered.

In this direction, the current chapter highlights the development and problems associated with various aspects shop floor scheduling. Heuristic procedures for solving scheduling problems were introduced in mid 1950s. The literature survey begins with papers published after 1990 with maximum attention paid to last ten years. The search was restricted on those articles for which full text was available. Table 2.1 provides the source and number of citations from each source. The majority of the citations are found in peer-reviewed journals.

Table 2.1 Summary of publications referred

Source	Citation
Annals of Operations Research	2
Applied Mathematics and Computation	2
Applied Soft Computing	1
Computers and Industrial Engineering,	17
Computers and Operation Research	18
Computing	1
European Journal of Industrial Engineering	1
European Journal of Operational Research	25
IEEE Transactions on Evolutionary Computation	1
Expert Systems with Applications	5
Flexible Services and Manufacturing Journal	1
Future Generation Computer Systems	3
IEEE Transactions on Robotics and Automation	2

IEEE Transactions on Systems, Man, and Cybernetics	2
IEEE Transactions on Evolutionary Computation	2
IIE Transactions	2
International Journal of Advanced Manufacturing Technology	13
International Journal of Computational Intelligence Systems	1
International Journal of Production Economy	7
International Journal of Production Research	5
International Transactions in Operational Research	1
Journal of Complexity International	1
Journal of Intelligent Manufacturing	3
Journal of Scheduling	3
Journal of the Operational Research Society	1
Management Science	2
Materials Science Forum	1
Mathematical Methods of Operations Research	1
Mathematics and Computers in Simulation	1
Naval Research Logistics Quarterly	1
OMEGA: International Journal of Management Science	3
Operations Research	6
Operations Research Letters	2
Production Planning and Control	4
Books	4
Conference papers	4
Total	149

2.2 Classification of literature

The literature review gives enough confidence to identify a pertinent gap or methodological weaknesses in the existing literature to solve the research problem. The literature on scheduling can be broadly classified in two ways - one based on types of scheduling and other one - the way the solution methodology used for scheduling which is illustrated in Figure 2.1. Next sections provide brief discussion on these issues. Finally, the chapter is concluded by summarizing the advancement taken place in scheduling problem and possible literature gap so that relevance of the present study can be emphasized.

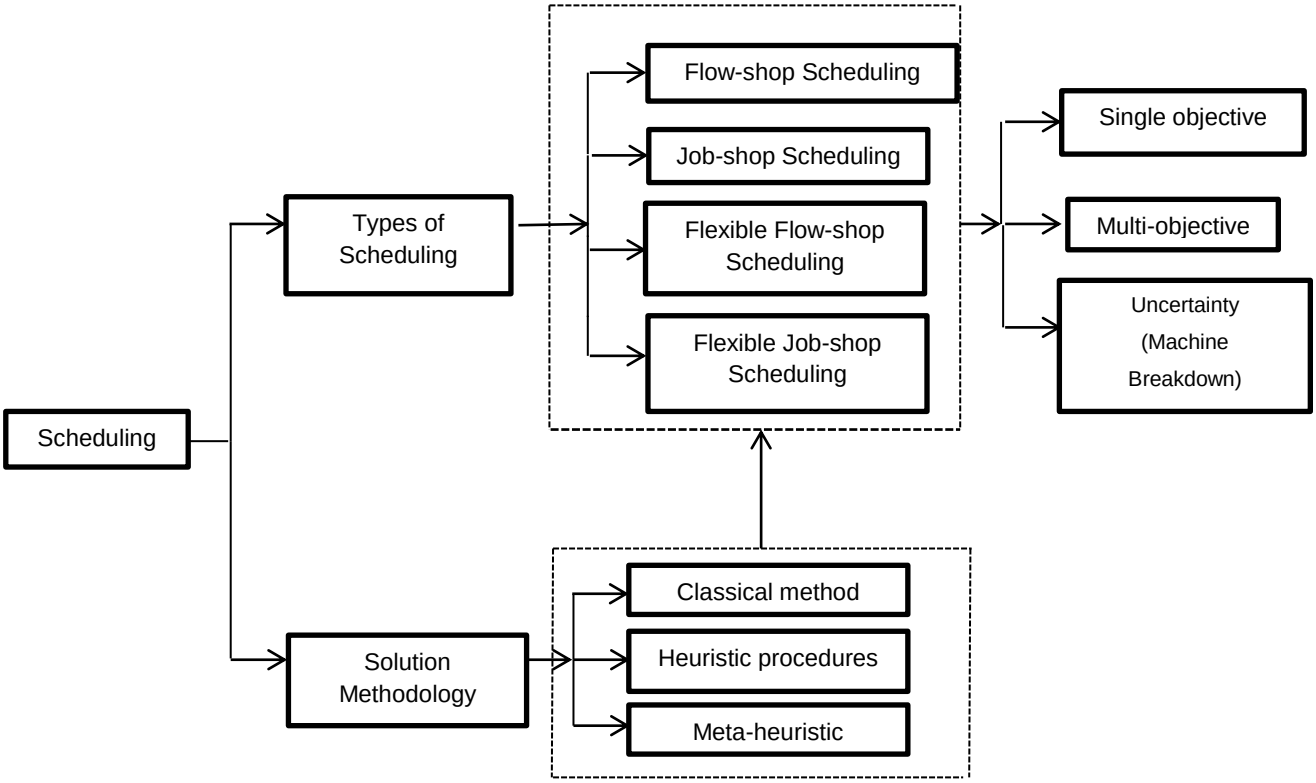


Figure 2.1 Taxonomic frameworks for Scheduling

2.3 Flow shop scheduling problem

The flow-shop scheduling problem (FSP) has been an interesting area of research for over last thirty years ever since Johnson [26] has anticipated the two stage scheduling problem with the makespan as an objective. The early research on flow-shop scheduling problems is mostly based on Johnson's rule which provides a procedure to obtain an optimal solution with two or three machines with certain characteristics. Palmer [27] has proposed a slope index based on the processing time to sequence the jobs on the available machines. The heuristic suggested by Campbell, Dudek and Smith (CDS) [28] is basically an extension of Johnson's algorithm. The CDS algorithm splits into a series of an equivalent two machine flow-shop problem for the 'm' machine problem and solves each equivalent problem by Johnson's rule. Gupta [29] has recommended another heuristic which is similar to Palmer's heuristic considering some exciting facts about optimality of Johnson's rule. NEH (Nawaz, Encore and Ham) heuristic is based on the assumption that a job should be given higher priority whose total processing time on all the machines is higher than job with low total processing time [30]. The NEH

algorithm does not transform the original m-machine problem into an artificial dummy two-machine problem. It generates the final sequence in a constructive way by accumulating a new job at every step and obtains the best solution. An improved heuristic for solving the flow-shop scheduling problem has been developed by Ho and Chang [31] and its performance with respect to makespan, mean utilization, and mean flow time is superior when compared with five well known heuristics. Rajendran and Chaudhuri [17] have proposed a heuristic algorithm to minimize flow time for a flow-shop scheduling problem using three heuristic criteria. The first criterion deals with the sum of idle times. The second criterion incorporates the sum of idle times and the waiting times. The third criterion includes the completion times of the partial schedule at various stages along with the above mentioned two criteria. Rajendran [32] has proposed an improved CDS algorithm. A heuristic preference relation is suggested and used as the basis to confine the search for possible enhancement in the multiple objectives. A proportionate flow-shop scheduling problem has been proposed in which the job processing times are inversely proportional to machine speeds by minimizing the maximum completion time [33]. Bulfin and M'Hallah [34] has proposed an exact algorithm to solve the two machine flow-shop scheduling problem with objective of weighted number of tardy jobs. Pranzo [35] has considered the two-machine batch scheduling flow-shop problem with sequence independent setup times and removal times. The study proposes a number of special cases of the problem and reduces the special cases to travel salesman problem (TSP).

Brown et al. [36] have recommended a non-polynomial time solution method and a heuristic for the no-wait flow-shop problem with sequence independent setup times and optimized the performance measures for both the makespan and total flow time. Blazewicz et al. [37] have investigated different solution procedures for the two machine flow-shop scheduling problem with a common due date and weighted late work criterion. Grabowski and Pempera [38] have addressed the no-wait flow-shop problem with makespan criterion and presented several variants of descending search and tabu search algorithms. The multi moves has been introduced to accelerate the convergence rate. Tabu search algorithm uses a dynamic tabu list to avoid to be trapped at a local optimum. Franca et al. [39] have presented a memetic approach (MA).The proposed MA algorithm uses an organized structured population as a ternary tree and local search scheme called RAI (recursive arc insertion). The experimental results indicate that the MA is superior to other algorithms but requires greater computational effort. Fink and Vob [40] have examined the application of different meta-heuristic methods to

solve the continuous flow-shop scheduling problem by minimizing the total completion time. An approach based on trade-off between solution quality and computational time is recommended. Ponnambalam et al. [41] have proposed a hybrid approach known as TSP-GA algorithm for flow-shop scheduling considering weighted sum of multiple objectives. The weights were randomly generated for each generation to enable a multi-directional search. The proposed algorithm was appraised by solving the benchmark problems available in the OR-Library. Lodree et al. [42] have developed a new scheduling heuristic for minimizing the number of tardy jobs in a dynamic flow-shop (job arrival or release dates are not known in advance). Ravindran et al. [43] have suggested three heuristic algorithms for solving the flow-shop scheduling problem to minimize the makespan and total flow time.

Ignall and Schrage [44] have proposed the first branch and bound (B&B) algorithms for permutation flow-shop problem with makespan minimization. The branch and bound algorithm for scheduling jobs with sequence dependent setup times on a single processor was suggested by Lockett and Muhlemann (1972) to optimize the total number of tool changes. It was computationally restrictive and suitable only for small sized problems. Hariri and Potts [45] have used B&B algorithm with at most fifteen jobs. Carlier et al. [46] have proposed two branch and bound algorithms for the permutation flow-shop problem. They used disjunctive graphs association with each operation on a machine which is a unique value of head and tail. The head is length of the longest path in the disjunctive graph from the source to the operation and tail is the length of the longest path from the current operation to the end. The branching rule was such that the first branching sequences a job at the beginning of the sequence and the second branching sequences a job at the end of the sequence. The node with the smallest value for the lower bound is selected as the branching node. Fry et al. [47] have suggested a branch and bound procedure to minimize mean absolute lateness. The branch and bound technique was used in conjunction with a one pass linear program. Blazewicz et al. [48] have assigned the two machine non-preemptive flow-shop scheduling problem with a common due date total and a weighted late work measure. A branch-and-bound algorithm for a two-machine flow-shop scheduling problem with deteriorating job. Gowrishankar et al. [49] have considered two types of problems. In the first case, m-machine flow shop scheduling with minimizing variance of completion times of jobs and second case with minimizing sum of squares of deviations of the job completion times from a common due date. A simple linear deterioration function was assumed and the

objective was to obtain a sequence that minimizes the makespan. Nowicki and Smutnicki [50] have addressed a new algorithm which uses some elements of the scatter search, the path relinking technique and some properties of neighborhoods to solve the flow-shop scheduling problem with the makespan criterion. Su and Lee [51] have considered the scheduling problem where a set of jobs are available for processing in a no-wait and separate setup two-machine flow-shop system with a single server. They compared the proposed branch-and-bound algorithm with method which was proposed by Aldowaisan [52] and the results are found to be encouraging.

A wide variety of meta-heuristic procedures like tabu search (TS), ant colony optimization (ACO), artificial immune system (AIS), simulated annealing (SA), particle swarm optimization (PSO) and genetic algorithm (GA) are used to solve such problems and generate approximate solutions close to the optimum with considerably less computational effort. Santos et al. [53] have improved the makespan of the multi-stage parallel flow-shop scheduling problem through fundamental adjustment of the exchange heuristic (EH). A hybrid approach of ordinal optimization and genetic algorithm called as order based genetic algorithm (OGA) for flow-shop scheduling problems was proposed by Wang et al. [54]. The simulated results illustrate that the OGA gives better solutions than GA, NEH and Blind search methods. They have also tested the various parameters of OGA and provided statistical results. A tabu search algorithm along with neural networks for permutation flow-shop scheduling problem was proposed by Solimanpur et al. [55].They have used the modified NEH algorithm proposed by Taillard [56] to generate the initial solution. They have used the insertion mechanism to generate the neighborhood structure since it was found to be more effective than a random swap mechanism [56]. Rajendran and Ziegler [57] have considered the problem of scheduling in permutation flow-shops by using ACO algorithms with the objective of minimizing the sum of the total flow time of jobs and makespan. The efficiency of the recommended ant colony optimization algorithm was assessed by considering the benchmark problems and upper bound values for makespan given by Taillard [56]. Shyu et al. [58] have developed the ACO algorithm to solve the two machine flow-shop scheduling problem with no waiting between operations including the set up time. Job processing times have been chosen randomly from the interval 0 to 100 and setup times also randomly chosen from different intervals 0-10, 0-50 and 0-100. Problem sizes vary from 50, 100, 150, 200 and 250 jobs. They have shown that the ACO algorithms outperform other algorithms. Lian et al. [59] have suggested a particle swarm optimization algorithm for solving the

permutation flow-shop scheduling problem with respect to minimization of makespan and computation experiments show that it is more efficient than GA. However, some problems cannot be solved to guarantee optimality. Pan et al. [60] have proposed a discrete particle swarm optimization (DPSO) algorithm for solving the no-wait flow-shop scheduling problem with both makespan and total flow time criteria. Solution quality was improved by hybridizing the DPSO algorithm with the variable neighborhood descent (VND) algorithm. Kuo et al.[61] have recommended a new hybrid particle swarm optimization model (HPSO) that combines the random-key encoding scheme (RK), individual enhancement scheme (IE) and particle swarm optimization (PSO) to solve the flow-shop scheduling problem (FSP) to obtain a sequence of jobs that minimizes makespan. The experimental results indicate that the flow-shop scheduling problem based on the proposed HPSO produces a better solution than the solutions based on GA. A hybrid genetic algorithm for the flow-shop scheduling problem was proposed by Tseng and Lin [62]. A modified version of NEH was used to generate the initial population and a new orthogonal array crossover was developed as the crossover operator of the genetic algorithm. Salmasi et al. [63] have established meta-heuristic algorithm based on ant colony optimization (ACO) and a lower bounding technique for flow-shop scheduling problem. They confirmed that the proposed ACO has a better performance than the other available meta-heuristics in the literature.

2.4 Job shop scheduling problem

Job-shop scheduling problem (JSP) was first proposed by Muth and Thompson [64]. In the last forty years, the JSP has become a standard scheduling problem closely related to industrial engineering and contributions have also been made by other research disciplines such as operations research, manufacture science, computer science and management science. The job-shop scheduling problem is a classical NP-hard problem, especially difficult to solve even in relatively small instances [65]. As an example, a particular instance having 10 machines and 10 jobs [65] remained unsolved for over 20 years until it has been solved by Carlier and Pinson [66]. Thus great deal of research has been made to study various job-shop scheduling problems in last four decades. Adams et al. [67] have proposed the shifting bottleneck (SB) which is a powerful heuristic for solving the JSP. In this method, the bottleneck machine (machine having maximum workload) is to be chosen for sequence. Each time a new machine has been sequenced, the sequence of each previously sequenced machine may be subjected to re-optimization. A non-linear mixed-integer programming model (MIP) has

been presented to formulate this problem. Branch and bound method is proposed by Singer and Pinedo [68] for solving JSP with due date criteria that minimizes the total weighted tardiness. Amaral et al. [69] have applied tabu search (TS) combined with dispatching rules to achieve an initial solution and searches new solutions in a neighborhood based on the critical paths of the jobs to solve the JSP with the objective of minimizing the total weighted tardiness.

Choi and Choi [70] have studied JSP with alternative operations and sequence-dependent setup times (SDST). A mixed integer program integrated with local-search scheme is proposed. Artigues and Roubellat [71] have proposed a polynomial insertion algorithm for multi-resource job-shop scheduling with sequence-dependent setup times for minimization of maximum lateness. First, they described the algorithm for pure JSP and then multi-resource requirements were introduced for the operations and finally, SDST was integrated in the multi-resource context. Low et al. [72] have determined the benefit of each lot-splitting situation in a job-shop environment. They have developed a mathematical programming approach with the objective of minimizing the sum material processing cost, setup time cost and inventory cost. Subramaniam et al. [73] have developed a framework to solve and optimize JSP problem with uncertain processing times in which imprecise processing times are modeled as triangular fuzzy numbers (TFN). Fandel et al. [74] have investigated an integrated job-shop production planning and scheduling problem. A new probabilistic model was introduced by Hongan et al. [75].

The conventional GA based on binary representation have been introduced to the job-shop scheduling problem [76]. Yamada and Nakano [76] have proposed a GA that uses problem-specific representation of solutions with crossover and mutation which are based on the Giffler and Thompson (GT) algorithm. Jaszkiewicz [77] has proposed genetic local search (GLS) which is a hybridization of GA and local search. The first ACO algorithm was proposed by Colorni et al. [78] to tackle a shop scheduling problem. The performance of ACO algorithm was unsatisfactory due to slow convergence, long computing time and falling into a local optimum easily. Steinhofel et al. [79] have presented simulated annealing based algorithms for the classical JSP problem where the objective is to minimize the makespan. Kolonko [80] has proposed a new approach that used a small population of SA embedded with the GA framework. Moreover, SA algorithm was used in three schemes i.e. pairwise exchange, insertion, and random insertion to solve job-shop scheduling problem. Zuo and Fan [81] have suggested an

immune algorithm for JSP. The antibody clone, antibody crossover, receptor editing, hyper mutation and niche technology has been used to retain the diversity of the population. Some benchmark problems were solved to verify the effectiveness of the recommended method. Chandrasekaran et al. [82] have recommended AIS algorithm to obtain optimal makespan values for different sized JSP problem. The algorithm was based on clonal selection and affinity maturation principles. A two phase mutation procedure and receptor editing has been used to generate new antibodies. Shuster [83] has studied the complexity of the job-shop scheduling problem with no-wait constraint and solved with a tabu search approach. Sha and Lin [22] have proposed a hybrid particle swarm optimization (HPSO) for job-shop scheduling problems and modified the representation of particle position, particle movement, and particle velocity to better performance and also applied tabu search to enhance solution quality. The computational results show that HPSO can produce better results than other methods. Essafi et al. [84] have considered the job-shop scheduling problem (JSP) with due dates and release dates to minimize the total weighted tardiness. A genetic algorithm integrated with a local search is adopted.

Baptiste et al.[85] have considered the JSP with the earliness and tardiness penalties. They have suggested two Lagrangian relaxations of the problem. The first one was based on the relaxation of precedence constraints while the second one was based on the relaxation of machine constraints. The results show that the relaxation of the resource constraints often leads to better lower bounds. Roshanaei et al. [86] have introduced a variable neighborhood search to solve sequence dependent setup times job shop scheduling problem. The meta-heuristic approach uses three different neighborhood search structures centered on insertion operator concept. Bozejko and Makuchowski [87] have presented a hybrid TS algorithm for a no-wait job-shop scheduling problem for minimizing the makespan. Pan and Huang [88] have proposed a hybrid genetic algorithm to solve the job-shop scheduling problem with no-wait constraint. In this algorithm, the new solutions are generated by transforming the chromosomes. Part of each chromosome is transformed into a travelling salesman problem (TSP) which is solved by heuristics to get a new sequence. This idea provides a better convergence for the algorithm. Jinwei et al. [89] have recommended a novel competitive co-evolutionary quantum genetic algorithm (CCQGA) for a stochastic job-shop scheduling problem (SJSP) with the objective to minimize the expected value of makespan. Three new strategies such as competitive hunter, cooperative surviving and

the big fish eating small fish are developed in population growth process. To increase the diversity of genes to avoid premature convergence, the suggested algorithm maintains the population size dynamically and accelerates the convergence speed.

Kachitvichyanukul and Sitthitham [90] have suggested a two-stage genetic algorithm (2SGA) for multi-objective job-shop scheduling problems with three measures i.e. makespan, total weighted earliness and total weighted tardiness. At first stage, parallel GA is applied to find the best solution of each individual objective function and then populations are combined at the second stage using the weighted aggregating objective function. The proposed algorithm can be used with one or two objectives without modification. Mati et al. [91] have suggested an efficient MA with a novel local search to solve the JSP. A systematic change of the neighborhood is accomplished to avoid trapping into local optima in the local search and two neighborhood structures are designed by exchanging and inserting based on the critical path. The objective of minimizing makespan is considered while satisfying a number of hard constraints.

2.5 Flexible flow shop scheduling

In the past, scheduling problem in a hybrid or flexible flow-shop has received attention of researchers because of its importance from both theoretical and practical points of view. This problem has been showed as an adequate model for the study of a great number of production systems specifically process industries such as glass, paper, metallurgy, wood, textile and aerospace. The flexible flow-shop scheduling problem is first proposed by Arthanari and Ramamurthy [92] in 1971 and solved by branch and bound algorithm. The problem has taken attention after 1994 while Gupta and Tunc [94] have considered a two-stage flow-shop scheduling problem when there is one machine at stage one and the number of identical machines in parallel at stage two is less than the total number of jobs. The setup and removal times of each job at each stage are separated from the processing times. They proposed a heuristic that was empirically tested to determine the effectiveness of finding an optimal solution. Guinet et al. [93] have proposed a heuristic for the makespan minimization problem in a two stage flexible flow-shop based on Johnson's rule. They compared the heuristic with the shortest processing time (SPT) and the longest processing time (LPT) dispatching rules. It is proved that the LPT rule gives good results for the makespan problem in a two-stage flexible flow-shop.

Liu and Chang [95] have exploited Lagrangian relaxation and proposed a search heuristic for the flexible flow-shop scheduling problem with sequence dependent setup

time, sequence dependent setup cost and non-zero release date to minimize the sum of setup times and costs. They have formulated the problem as a separable integer programming problem. Moursli and Pochet [5] have introduced a branch and bound algorithm to minimize makespan for hybrid flow-shop. Botta-Genoulaz [96] has used six new heuristics to solve the hybrid flow-shop scheduling (HFSP) problem with setup and removal time, precedence constraints and time lags. The goal is to minimize maximum lateness. Neron et al. [97] have presented the use of time-bound adjustment to enhance the efficiency of branch and bound procedures for solving the hybrid flow-shop scheduling problem.

Gupta et al.[98] have proposed heuristics to solve the HFSP considering variation of the processing times of the operations on some machines and a due date assignment cost. The objective is to minimize makespan. Su [99] has considered two-stage hybrid flow-shop. The first stage consists of a batch processor and the second stage consists of a single processor. Each batch processor can process a batch of jobs simultaneously. They have proposed a mixed integer linear programming (MILP) formulation to find the optimal solution and a heuristic algorithm is developed. The heuristic algorithm is divided into two stages: in 1^{st} stage, the jobs are allocated to batches such that the number of batches formed is minimized; and in 2^{nd} stage, the jobs are sequenced within a batch and the batches are later sequenced. The results from the heuristic are compared with MILP and it is observed that the heuristic performed consistently well with low CPU times.

Hmida et al. [100] have presented the depth-bounded discrepancy search (DDS) method to obtain near-optimal solutions with makespan of high quality. This method contains no idleness for the search tree expansion for the hybrid flow-shop scheduling (HFS). Again to improve the solutions of HFS problem, they have proposed a local search method known as climbing depth-bounded discrepancy search (CDDS) which is a hybridization of two existing discrepancy-based methods: DDS and climbing discrepancy search (CDS). Kurz and Askin [101] have considered the problem of flexible flow line scheduling with sequence-dependent setup times to minimize the makespan. They have proposed an integer programming (IP) model and a lower bound for the problem. They have also developed a random keys genetic algorithm (RKGA) because it is difficult to solve the problem by using the IP model. Engin et al. [20] have proposed an improved artificial immune system where a computational method based on clonal selection principle and affinity maturation mechanism of the immune response is used.

Low [102] has considered the problem of flexible flow-shop scheduling with unrelated parallel machines to minimize total flow time. The assumptions considered are independent setup and dependent removal times. A simulated annealing is proposed based on heuristic to solve the problem. Oguz et al. [103] have proposed a genetic algorithm for FFSP and integrated with a new crossover operator. First, it performed a preliminary test to set the best values of the control parameters i.e. population size, crossover rate and mutation rate. An extensive computational experiment was carried out to evaluate the performance of the proposed four versions of genetic algorithm in terms of the percentage deviation of the solution from the lower bound. Ruiz and Maroto [104] have recommended a genetic algorithm for a complex generalized flow-shop scheduling problem with sequence dependent setup times, unrelated parallel machines at each stage. They have proposed hybrid genetic algorithm which integrated with new characteristics and four new crossover operators. Janiak et al. [105] have suggested the problem of hybrid flow-shop for minimizing the summation of the total weighted earliness, the total weighted tardiness and the total weighted waiting time. Three constructive algorithms and three meta-heuristics based on tabu search and simulated annealing algorithms are proposed to solve the problem. Ying and Lin [106] have suggested an novel ant colony system (ACS) for solving multistage hybrid flow-shop scheduling problem with multiprocessor tasks. In the proposed algorithm, the same formula is used as classical ACO, but with a different starting solution procedure that affects the probability function so the quality of the solution is improved. Tseng et al. [107] have proposed multistage hybrid flow-shop scheduling problem with multiprocessor tasks. They have solved the problem by PSO with a new a velocity equation. They have verified the PSO algorithm with nine possible combinations of PSO with three velocity equations and three neighborhood topologies and compared with two existing genetic algorithms and an ant colony optimization algorithm. The proposed PSO algorithm outperforms all the existing algorithms for the same benchmark problems.

Kahraman et al. [108] have addressed the hybrid flow-shop (HFS) scheduling problems to minimize the makespan value. They proposed an efficient genetic algorithm based on a permutation representation of the n jobs. A direct coding approach was used i.e. a chromosome represents a schedule of the jobs directly. Allahverdi and Al-Anzi [109] have studied a two-stage assembly scheduling problem where there are ‘m’ machines on the first stage and an assembly machine at the second stage. In their model the setup times are treated as separate from the processing times. A

dominance relation was presented and proposed three heuristics i.e a hybrid tabu search, a self-adaptive differential evolution (SDE), and a new self-adaptive differential evolution (NSDE). Al-Anzi and Allahverdi [110] have considered the same problem of [109] where setup times was ignored and proposed some heuristics based on tabu search, particle swarm optimization (PSO), and self-adaptive differential evolution (SDE) along with the earliest due date (EDD) and Johnson heuristics to solve the problem. Computational experiments reveal that both PSO and SDE are superior to tabu search and PSO performs better than SDE. Mirsanei et al. [111] have considered hybrid flow-shop scheduling with parallel identical machines and makespan criterion. A novel simulated annealing (NSA) algorithm is proposed to obtain a reasonable schedule within an adequate computational time. The obtained results are compared with immune algorithm (IA) and random key genetic algorithm (RKGA) from the literatures. The results reveal that NSA outperforms both IA and RKGA.

2.6 Flexible job shop scheduling problem

Brandimarte [112] is the innovator in addressing flexible job-shop scheduling problem (FJSP). They have developed a polynomial algorithm for solving this problem with two jobs. Brandimarte [112] has applied hierarchical approach for FJSP based on decomposition and dispatching rule. First, routing sub-problem is solved and then sequencing sub-problem is solved using a TS algorithm. Saidi-Mehrabad et al. [113] have suggested a hierarchical approach. Brucker et al. [114] have suggested methods based TS algorithm to solve FJSP for both hierarchical and integrated approach. Dauzere-Peres and Paulli [115] have proposed a new neighborhood structure for the FJSP problem and recommended the TS algorithm for re-sequencing and rearranging the operation. Mastrolilli and Gambardella [116] have recommended two neighborhood functions incorporated with TS algorithm to find better performances than other existing meta-heuristics in terms of computation time and solution quality.

Kacem et al. [117] have suggested an evolutionary method for solving FJSP. Integrated approaches consider both assignment and sequencing sub-problems simultaneously. Usually, integrated approaches produce better solutions than hierarchical approaches but more difficult to solve and consumes more computational time. Xia and Wu [16] have presented a practical hierarchical solution approach by making use of PSO to assign operations on machines and simulated annealing algorithm to schedule operations on each machine. Zhang et al. [118] have proposed a multistage operation based GA to deal with the flexible job-shop scheduling problem

from a point view of dynamic programming. Fattahi et al. [119] have proposed a mathematical model and integrated two meta-heuristics approaches (SA and TS algorithms) for solving FJSP. Six different searching algorithms have been presented besides two established meta-heuristics considering both integrated and hierarchical approaches. Ho et al. [120] have developed an architecture known as learnable genetic architecture (LEGA) for learning and evolving solutions for the FJSP. The architecture identifies a population generator module that produces the initial population of schedules and also trains the schemata learning module. A large range of benchmark data taken from literature are used to analyze the efficacy of LEGA.

Pezzella et al. [121] have presented a genetic algorithm (GA) that incorporates different approaches for creating the initial population, selecting the individuals for reproduction and reproducing new individuals. The computational study indicated that the integration of certain strategies in a genetic framework leads to results comparable to those obtained by the best-known algorithm. Gao et al. [122] have employed a hybrid GA and variable neighborhood descent (VND) algorithm for FJSP. VND involves two local search procedures: local search of moving one operation and local search of moving two operations. Xing et al.[123] have proposed a knowledge-based ant colony optimization algorithm (KBACO) which is integration between ant colony optimization (ACO) model and knowledge model to solve the FJSP. In the KBACO algorithm, knowledge model learns some available knowledge from the optimization of ACO and then applies the existing knowledge to guide the current heuristic searching. Bagheri et al. [124] have employed an artificial immune system (AIS) algorithm to solve the flexible job-shop scheduling problem. The AIS algorithm used different strategies for producing the initial population and selecting the individuals for reproduction. Different mutation operators are also utilized for reproducing new individuals. Yazdani et al. [125] have developed a parallel variable neighborhood search (PVNS) algorithm for solving FJSP. Parallelization in the presented optimization method increases the diversification and the exploration in the search space. Defersha and Chen [126] have developed a parallel GA to minimize the makespan in a complex flexible job-shop scheduling which includes sequence dependent setup times, machine release dates and time lag requirements. Hmida et al. [127] have recommended a new discrepancy-based method known as climbing depth-bound discrepancy search (CDDS). They have used a block concept of Jurisch [128] to find the neighborhood structure. The computational experiments revealed that proposed method outperforms the GA of Pezzella et al. [121] and the TS of

Mastrolilli and Gambardella [116] for all instances. Bozejko et al. [87] have proposed a parallel double-level meta-heuristic approach for the flexible job-shop scheduling problem based on two methods implemented i.e. tabu search and population-based approach. Additionally, they have implemented two new major modules: the machine selection module refers to execute sequentially and the operation scheduling module to execute in parallel. Zhang et al. [129] have recommended an effective GA for solving the FJSP to minimize makespan. In the proposed algorithm, global selection (GS) and local selection (LS) has been designed to generate high-quality initial population in the initialization stage. An improved chromosome representation was used to conveniently represent a solution of the FJSP and different strategies for crossover and mutation operator were adopted. Computational results proved that the proposed genetic algorithm is effective and efficient for solving flexible job-shop scheduling problem.

2.7 Scheduling based on machine breakdown

Efforts on scheduling problems normally consider a static environment with a fixed number of jobs, deterministic processing times and no unexpected events that would influence the job processing when the schedule is executed [130, 131]. A two-machine flow-shop scheduling problem with an availability constraint was proposed by Lee [132]. He has developed a pseudo-polynomial dynamic programming algorithm to solve the problem optimally and also developed two O(n log n) time heuristic algorithms with an error bound analysis. Allahverdi and Mittenthal [133] have considered dual-criteria scheduling on a two-machine flow-shop subject to random breakdowns with respect to both makespan and maximum lateness objective functions. They provided an elimination criterion in a two-machine flow-shop when both machines are subjected to random breakdowns. They proved that the longest processing time and the shortest processing time orders are optimal with respect to both criteria in a two machine ordered flow-shop when the first or the second machine respectively suffers stochastic breakdowns. Mehta and Uzsoy [134] have generated a predictive schedule by using the available information on uncertainties. The effect of disruptions on planned activities was measured by the difference between the planned completion times of jobs in the predictive schedule and their realized completion times. Completion time deviations were reduced by inserting additional idle time into the predictive schedule i.e. by under-capacity scheduling. The amount of additional idle time inserted based on the structure of the predictive schedule and the nature and frequency of the disruptions that is expected to occur. Thus, the completion times of jobs in the predictive schedule depend on the schedule and the

amount of additional idle time inserted. Holthaus [135] has investigated simulation-based analysis of dispatching rules in job-shop scheduling taking into account machine breakdown. Results reveal that different levels of breakdown parameters have significant impact on performance of scheduling rules. Sabuncuoglu and baylz [136] have studied the reactive scheduling problems in a stochastic manufacturing environment and tested on several scheduling policies under machine breakdowns in a classical job-shop system. In addition, they have measured the effect of system size and type of work allocation (uniform and bottleneck) on the system performance. The performance of the system was measured for the makespan and mean tardiness criteria. The robust and flexible solutions for flexible job-shop scheduling problems was proposed by Jensen[137]. A robustness measure has been defined and investigated by using a genetic algorithm to obtain robust and flexible schedules with a low makespan. These schedules were demonstrated to perform significantly better in rescheduling after a breakdown than ordinary schedules.

A robustness framework has been introduced by Allaoui et al. [138] to minimize makespan of a stochastic FFS problem under machine breakdown. Rangsaritratsamee et al. [139] have suggested a rescheduling method with local search genetic algorithm for a job-shop scheduling with dynamically arriving jobs. The proposed algorithm considers the efficiency by preserving the makespan, tardiness and stability by minimizing the jobs starting time deviations simultaneously. Here, the rescheduling takes place at definite time intervals using all available jobs at the rescheduling moment. Kasap et al.[140] have investigated optimal sequencing policies for the expected makespan problem on a single machine subject to random breakdowns where jobs have to be reprocessed in their entirety if preemptions occur because of breakdowns. They identified a class of uptime distributions under which LPT minimizes expected makespan. Suwa and Sandoh [141] have proposed a new when-to-schedule policy in reactive scheduling which considers time of schedule revision based on the concept of a control limit policy. Schedule revision is carried out based on a cumulative job delay and can be measured to determine suitable timing of schedule revision.

Goren and Sabuncuoglu [142] have defined two robustness measures and three stable measures. Then, a dominance rule and two lower bounds for one of the robustness measures are used in a branch-and bound algorithm to solve the problem exactly. A beam search heuristic is also proposed to solve large problems for all five measures. Gholami et al. [131] have proposed a heuristic to solve FFS scheduling

problems with sequence-dependent setups and stochastic machine breakdown. This method employs the random key genetic algorithm (GA) to identify the optimal solution. They have proposed a simulator in which the event-driven policy and the right-shift heuristic approach is incorporated into the GA to evaluate the expected makespan. The robustness of the algorithm is analyzed using the Taguchi parameter design. The number of jobs, the number of stages, the mean-time-between-failure and the population size are found to have a significant impact on the robustness of the algorithm.

Yahyaoui et al. [143] have suggested a new shifting method for job-shop scheduling problems with machine unavailability. In their research, six rules were put forward to determine when and how to right shift by determining the different starting times for each operation in order to minimize the makespan. Shifting one operation to the right will affect not only the succeeding operations of the same job but also the operations on the same machine. Zandieh and Adibi [144] have suggested a scheduling method based on variable neighborhood search (VNS) to solve a dynamic job-shop scheduling problem that considers random machine breakdown. In their method, an event-driven policy is selected. To enhance the efficiency and effectiveness of the scheduling method, an artificial neural network with a back propagation error learning algorithm is used to update parameters of the VNS at any rescheduling point according to the problem condition. The problem of scheduling stochastic job-shop scheduling subjected to breakdown was proposed by Lei [145]. An efficient genetic algorithm (EGA) is used to solve JSP with exponential processing time and non-resumable jobs. The objective is to minimize the stochastic makespan. A novel random key representation is suggested to represent the schedule of the problem and a discrete event-driven decoding method applied to build the schedule and handle breakdown. Al-Hinai and Mekkawy [146] have defined a number of bi-objective measures combining the robustness and stability of the predicted scheduling. Consequently, a two-stage hybrid genetic algorithm is proposed to generate the predictive scheduling. Hasan et al. [147] have proposed an improved local search technique, shifted gap reduction (SGR), which improves the performance of GAs when solving relatively difficult test problems. The new algorithm for JSP with machine unavailability or breakdown is modified considering two scenarios of machine unavailability.

2.8 Multi-objective scheduling

Although the single-objective FFSP and FJSP have been widely investigated, the research on the multi-objective FFSP and FJSP is still relatively limited. In such multi-

objective scheduling problems, it is common to obtain a set of pareto-optimal or efficient solutions such that all solution obtained are the best solution i.e. the solutions in the set do not dominate each other. Ponnambalam et al. [148] have suggested a multi-objective genetic algorithm (MOGA) to derive the optimal machine-wise priority dispatching rules to resolve the conflict among the contending jobs in the Giffler and Thompson (GT) procedure applied for job-shop scheduling problems. The performance criterion considered is the weighed sum of the multiple objectives minimization of makespan, minimization of total idle time of machines and minimization of total tardiness. The weights assigned for combining the objectives into a scalar fitness function are not constant. The weights are specified randomly for each evaluation. This leads to the multidirectional search in the proposed multi-objective genetic algorithm. Parsopoulos and Vrahatis [149] were the first to propose the particle swarm optimization method in multi-objective optimization problems. Kacem et al. [117] have proposed a localization approach to solve the resource assignment problem and an evolutionary approach controlled by the assignment model to solve the mono-objective and multi-objective FJSP. Chang et al. [150] have proposed the gradual priority weighting approach to search the pareto optimal solution for multi objective FSP. The proposed approaches search the feasible solution space from the first objective at the beginning and towards the other objective step by step. They considered the multi-objective flow-shop scheduling problem by considering makespan, total flow time, total tardiness and maximum tardiness as the performance measures. The effectiveness and efficiency of gradual-priority weighting (GPW) approach is compared with the variable weight approach and the proposed method performs quite effectively and efficiently.

Coello et al.[151] have presented an approach in which Pareto dominance is combined into particle swarm optimization in order to allow the heuristic to handle problems with several objective functions. Xia et al. [152] have recommended a practical hierarchical solution approach for solving multi-objective FJSP. The proposed approach use particle swarm optimization (PSO) to allocate the operations on machines and simulated annealing (SA) algorithm to schedule operations on each machine. The objective is to minimize makespan, the total workload of machines, and the workload of the critical machine. Suresh and Mohanasundaram [153] have applied pareto archived simulated annealing to the multi-objective job-shop scheduling problem in which the related objectives are minimization of makespan and mean of flow time. Lei and Wu [154] have proposed a crowding measure based multi-objective evolutionary algorithm

(CMOEA) which makes use of the crowding measure to adjust the external population and assign different fitness for individuals. The comparison between CMOEA and strength pareto evolutionary algorithm (SPEA) indicates that CMOEA performs better in job-shop scheduling with two objectives including minimization of makespan and total tardiness. Gao et al. [155] have addressed the FJSP problem with three objectives such as minimization of makespan, maximal machine workload and total workload. The authors have developed a new genetic algorithm hybridized with an innovative local search procedure for the problem. Advanced crossover and mutation operators are proposed to adapt to the special chromosome structures and the characteristics of the problem. The bottleneck shifting works over two kinds of effective neighborhood which use interchange of operation sequences and assignment of new machines for operations on the critical path. The performance of the proposed method was tested by numerical experiments on a large number of representative problems. Ho et al. [156] have studied a hybrid evolutionary algorithm combined with a guided local search and an external pareto archive set. They have proposed an efficient approach for solving the multiple-objective flexible job-shop scheduling by combining evolutionary algorithm and guided local search (GLS). Instead of applying random local search to find neighboring solutions, they introduced a GLS procedure to accelerate the process of convergence to pareto-optimal solutions. The main improvement of this combination is to diversify the population toward the pareto front. A branch and bound algorithm was also proposed to obtain the lower bounds of multiple-objective solutions. Experimental results indicate that the multiple-objective pareto-optimal solutions of the algorithm dominate previous designs for solving the same benchmarks while incurring less computational time.

Try et al.[157] have solved multi-objective flexible job-shop scheduling problems using dispatching rules discovered through genetic programming. They have evaluated and employed suitable parameters and operators for evolving composite dispatching rules using genetic programming with an aim towards greater scalability and flexibility. Experimental results show that composite dispatching rules generated by genetic programming framework outperforms the single dispatching rules and composite dispatching rules selected from literature over five large validation sets with respect to minimum makespan, mean tardiness and mean flow time as objectives. Yagmahan and Yenisey[158] have considered the flow-shop scheduling problem with multi-objectives of makespan, total flow time and total machine idle time. Ant colony optimization (ACO) algorithm was proposed to solve this problem. The proposed algorithm was compared

with solution performance obtained by the existing multi-objective techniques. Lei [159] have presented a particle swarm optimization for multi-objective job-shop scheduling problem in order to simultaneously minimize makespan and total tardiness of jobs. Job-shop scheduling is converted into a continuous optimization problem by constructing the corresponding relation between real vector and the chromosome obtained by using priority rule-based representation method. They have also designed a Pareto archive particle swarm optimization in which the global best position selection is combined with the crowding measure-based archive maintenance. Sha and Lin [22] have presented PSO for multi-objective job-shop scheduling problem. The original PSO was used to solve continuous optimization problems. Due to the discrete solution spaces of scheduling optimization problems, they modified the particle position representation, particle movement, and particle velocity in their study. The modified PSO was used to solve various benchmark problems. Zhang et al. [160] have proposed a new method based on multi-objective particle swarm optimization to deal with the flexible job-shop scheduling problems with multiple objectives such as minimizing completion time, total machine workload and the biggest machine workload. This algorithm adopts linear weighting method to change multi-objective optimization problem into the single objective optimization problem and introduces random and uniform design method to produce weight coefficient. Wang et al. [161] have presented a multi-objective genetic algorithm (MOGA) based on immune and entropy principle to solve the multi-objective FJSP. In this improved MOGA, the fitness scheme based on Pareto-optimality was applied and the immune and entropy principle was used to keep the diversity of individuals and overcome the problem of premature convergence. Efficient crossover and mutation operators were proposed to adapt to the special chromosome structure.

2.9 Discussions

This chapter provides the insight into various past developments and improvements in the area of scheduling. Based on established shared themes, it is divided into six main sections i.e. flow-shop scheduling, job-shop scheduling, flexible flow-shop scheduling, flexible job-shop scheduling, scheduling under machine breakdown situation and multi-objective scheduling. Figure 2.2 provides the breakdown of the number of citations based on the types of scheduling. The Figure 2.2 reveals that most of the research is concentrated on the flow-shop and job-shop scheduling. However, the researchers are attracted slowly towards the FFSP and FJSP to propose suitable methods to reduce the computational burden and obtain the near optimal solution in a reasonable time.

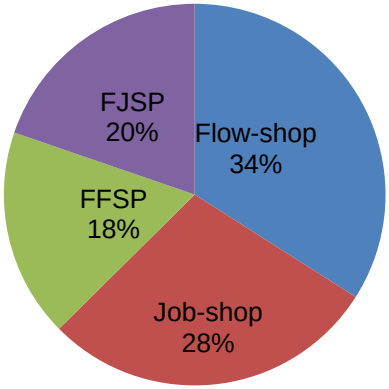


Figure 2.2 Percentage of paper surveyed based on types of scheduling

Reviewed literature is classified among the different solution methodology and techniques adopted by the authors. Figure 2.3 shows a pie chart with this distribution. It illustrates that 51% of the studied literature propose meta-heuristic approach whereas 26% propose heuristic methods. 23% of the studied literature use classical methods for solving scheduling problems. Most of the articles focus on efficient meta-heuristics that can be used for solving real-life problems at reasonable computational effort.

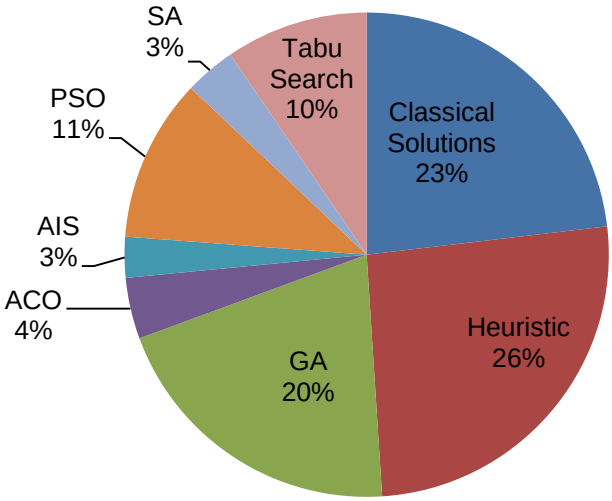


Figure 2.3 Percentage of paper surveyed based on solution methodology

Figure 2.4 illustrates that the literature is heavily biased towards the single objective criterion with a 78% of the references. It is striking to see from all surveyed papers that only a 10% deals with the machine breakdown situation criterion and 12% deals with the multi-objective scheduling.

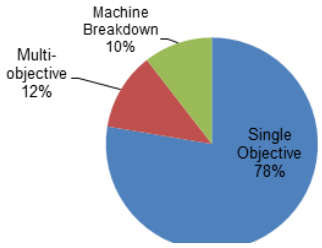


Figure 2.4 Percentage of paper surveyed based on types of objectives

2.10 Conclusions

After a comprehensive study made on the existing literature, it is concluded that the scheduling problem is a rich and promising field of research with application in manufacturing and industrial engineering. The analysis of scheduling problem in 21st century shows a change of interest in FSP and JSP topics studied by researchers and focus their attention on extended FSP and JSP models i.e. flexible flow-shop and flexible job-shop scheduling with new heuristic, optimization algorithms for the both single and multiple objectives. With regard to scheduling under machine breakdown scenario, adequate research effort has not been addressed to any possible combinations of robust measure with different fundamental scheduling approaches. Thus, the potential research opportunity is identified and an attempt has been made to propose a scheduling framework combining with the robust measure to deal with machine breakdown scenario in an FFSP and FJSP. In the last decade, several meta-heuristic algorithms have been developed which is based on the nature inspired analogy. However, despite of having several attractive features, it has been observed that most of these algorithms do not always perform as per expectations. The achievement of most of the meta-heuristics optimization algorithms depends on exploration and exploitation capability of the algorithm. The exploration and exploitation capability can be achieved by using local search methods or global search approaches or an integration of both global and local search strategies.

The present research work proposes an efficient PSO algorithm which has gained much attention and been successfully applied to a wide range of applications. Hence, in this work, an attempt has been made to proposed and analyzes the improvement of PSO algorithm with other search techniques and presents a comparison of the proposed PSO and QPSO, which is a new version of PSO, with the well-known algorithm from the literature. The next chapter briefly discusses the FFSP. The evolutionary techniques namely PSO and QPSO have been proposed to determine the optimum schedule in FFSP.

CHAPTER 3

FLEXIBLE FLOW SHOP SCHEDULING

3.1 Introduction

In the manufacturing environment, scheduling is primarily related to the problem of finding successive assignment of limited resources to a number of jobs which is optimal in terms of certain performance measures such as flow time, tardiness, lateness, and makespan [94]. A flexible flow shop problem (FFSP), an extended version of the simple flow shop, consists of two or more production stages in series and there exists one or more parallel machines at each stage. Usually, the flexible flow shop problem is denoted as $\left(\mathrm{FH_m}, \left(\mathrm{PM^{(j)}}\right)_{j=1}^{m} || \mathrm{C_{max}}\right)$ in which flexible flow shop or hybrid flow shop is denoted as $\mathrm{FH_m}$ with number of stages m and $\left(\mathrm{PM^{(j)}}\right)_{j=1}^{m}$ represents that one or more number of identical parallel machines exists in all stages j=1,.....,m [162]. FFPS has been successfully applied in a wide variety of industries to solve a wide range scheduling problems. The industries include the electronics [163, 164, 165], textile [166], production of concrete [167], manufacturing of photographic film [168,169], and others [170; 104; 171, 172, 173, 174]. FFSP has also been applied in non-manufacturing areas like civil engineering [175], internet service architectures [175] and container handling systems [177].

In this chapter, scheduling of a multistage flexible flow shop with parallel identical machines in each stage is considered with the objective to minimize the makespan (C_{max}) i.e., the completion time of all jobs in the last stage. A set of n jobs, i =1,2,3…..n, needs to be processed in a production system with m production stages and at each stage, a set of identical parallel machines M_j exists as illustrated in the Figure 3.1. Job i requires a processing time P_{ij} in stage j. It is assumed that each job is available at time zero and pre-emption is not allowed. The jobs are assumed to be independent of each other and each job is processed as a whole i.e. a job cannot be divided.

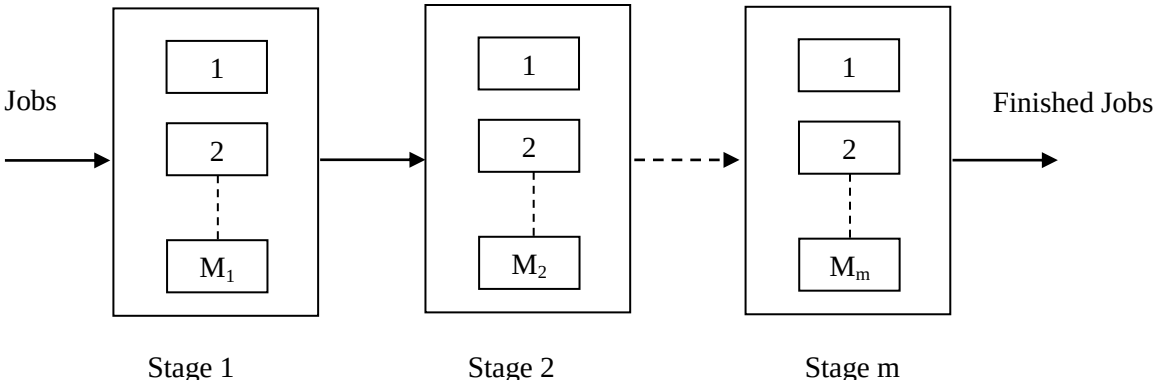


Figure 3.1 A flexible flow shop environment

The FFSP scheduling problem can be formulated mathematically as Mixed Integer Linear Programming (MILP) problem as follows [178]. The symbols used are as follows:

m Number of stages

n Number of jobs to be scheduled

M_j Number of machines at stage j, j=1, 2…M_j

N_{jk} The number of jobs which are processed on machine k at stage j, $j = 1,2, \ldots . m$; $k \in (1,2, \ldots ., M_j)$

P_{ijk} The processing time when job i is processed on machine k at stage j, i = 1,2,…N_{jk}, j =1,2,..m, k =1,2,..M_j

C_{ijk} The time when job i is finished on machine k at stage j

S_{ijk} The time when job i starts to be processed on machine k at stage j

Y_{ijk} = 1, if job i is processed on machine k at stage j,

= 0, otherwise

h_{jk} The h^{th} job which is processed on machine k at stage j, $h = 1,2, \ldots ., N_{jk}$

The constraints of FFSP are as follows:

The constraints of assigning

Each job can be assigned to one machine at every stage i.e.

$$\sum_{k=1}^{M_j} Y_{ijk} = 1 \quad i = 1,2, \ldots , n ; \; j = 1,2, \ldots , m \tag{3.1}$$

For each stage, the number of the jobs assigned to machines must be n

$$\sum_{k=1}^{M_j} N_{jk} = n \qquad j = 1,2, \ldots , m \tag{3.2}$$

The constraints of time: a job cannot be processed at next stage until it has finished its processing at the current stage. This is expressed as

$$C_{ijk} \leq S_{i(j+1)\,r} ; \; i = 1,2, \ldots , n ; \; j = 1,2, \ldots , m ; \; k = 1,2, \ldots . M_j ; r = 1,2, \ldots M_{j+1} \tag{3.3}$$

For any job, its finishing time is determined by its processing time and starting time on a machine at a stage. It can be given by

$$C_{ijk} = S_{ijk} + P_{ijk} \quad i = 1,2, \ldots , n ; j = 1,2, \ldots , m ; \; k = 1,2 \ldots . M_j \tag{3.4}$$

For a machine at any stage, it can process next job only after it has finished the current one. The mathematical expression is given as

$$C_{h_{jk}\,jk} \leq S_{(h+1)_{jk}jk} ; \; j = 1,2, \ldots , m ; \; k = 1,2, \ldots . M_j ; \; h = 1,2, \ldots . N_{jk} - 1 \tag{3.5}$$

Makespan of the schedule must be always equal or greater than the completion time of last job at last stage.

$$C_{max} = \max\{C_{imk}\} \; i = 1,2, \ldots , N_{mk} ; \; k = 1,2, \ldots , M_m \tag{3.6}$$

For example, four jobs need to be scheduled in two processing stages with three machines at each stage then total number of decision variables becomes 24 (4 jobs × 2 stages × 3 machines = 24). The number of constraints according to the equation 3.1 is 8 (4 jobs × 2stage = 8) , according to the equation 3.2 it is 2 (2 stages), according to the equation 3.3 it is 36 ((4 jobs × 3 machines)stage 1 × (3 machines) stage 2 = 36), according to the equation 3.4 it is 24 (4 jobs × 2 stages × 3 machines = 24), according to the equation 3.5 it is 18 (3 jobs (n-1) × 2 stages × 3 machines =18). From the above instance, total eighty eight numbers of constraints are found for such a small problem. Thus, scheduling of jobs in a flexible flow shops is considered as NP-hard problem because increase in problem size leads to increase in number of constraints resulting in finding solution in polynomial time. Therefore, classical approach like integer programing for solving scheduling problems requires exponential computation time in most cases leading to impractical computational burden for large scale application.

For such problems, it is not always possible to find an optimal solution in a reasonable time. Gupta [94] has shown that even the two-stage flow shop scheduling problem with parallel processors to minimize makespan is a non-deterministic polynomial-time hard (NP-hard) problem. The above formulation can be solved by branch and bound (B&B) method for general FFSP problem with any number of stages and any number of parallel machines per stage [4]. Despite the relative success of exact algorithms, they are still incapable of solving medium and large size problems and too complex for real world problems. Therefore, it is essential to look for non-exact but efficient meta-heuristics that can be used for solving real-life problems at reasonable computational effort. To address this issue, a variety of heuristic procedures such as dispatching rules and local search and meta-heuristics procedures like tabu search (TS), simulated annealing (SA), particle swarm optimization (PSO) and genetic algorithm (GA) are used to solve such problems and generate approximate solutions close to the optimum with considerably less computational time [4,16,57,177,179]. Most meta-heuristics proposed for the FFSP use a simple strategy of restricting the search to the space of job permutations. The idea is to find a permutation of the n jobs and builds a schedule by assigning jobs onto the machines according to this ordering.

PSO is an effective algorithm which gives high quality solutions in a reasonable computational time and consists of less number of parameters to be adjusted as compared to the other evolutionary meta-heuristics like GA [152]. Due to the simple concept, easy implementation, and quick convergence, PSO has gained much attention

and been successfully applied to a wide range of applications such as job scheduling, power and voltage control, mass spring system, supply chain network and vehicle routing problems [21, 22, 23, 24, 25]. Recently, a new variant of PSO, called quantum-behaved particle swarm optimization (QPSO), has been proposed in order to improve the global search ability of the original PSO [180]. PSO has an inherent drawback of getting trapped at local optimum due to large reduction in velocity values as iteration proceeds and poses difficulty in reaching at best solution. However, this drawback can be effectively addressed using quantum-behaved particle swarm optimization due to its advanced global search ability of the original PSO. The iterative equations of QPSO is different from that of PSO in that it needs no velocity vectors for particles, needs fewer parameters to be adjusted and can be executed easily. It has been proved that such iterative equations leads QPSO to be global convergent [181]. The stagnation of search in PSO can also be avoided by introducing diversity in the solution through the use of mutation, an important operator used in genetic algorithm. Mutation operator is an integral part of evolutionary computation techniques preventing loss of diversity in a population of solutions which allows a greater region of the search space to be covered. Mutation operators introduce new individuals into a population by creating a variation of a current individual; thus adding variability into the population and preventing stagnation of the search in local optima. It has been demonstrated that optimization algorithms using chaotic sequence can carry out overall searches at higher speeds than stochastic searches that depend on probabilities due to the non-repetition of chaotic sequence. In this context, several optimization algorithms using chaotic sequences have been proposed for solving various problems (182,183). Application of chaotic sequences instead of random sequences in PSO is a powerful strategy to obtain a diversified population of particles and improve the PSO's performance in preventing premature convergence.

In this chapter, the search mechanism of the particle swarm optimization (PSO) and quantum particle swarm optimization (QPSO) is explored to solve FFSP. The proposed approach uses PSO and QPSO to assign jobs on machines at each stage and schedule job sequence on each machine in the corresponding stages. The objective considered in this chapter is to minimize makespan. Chaotic numbers are used instead of random numbers to improve the solution diversity [184]. In addition, mutation, a popular operator in genetic algorithm, is embedded in the standard PSO and QPSO algorithm to escape from local optima.

3.2 Particle swarm optimization

Particle swarm optimization (PSO) algorithm, originally introduced by Kennedy and Eberhart [185], is a population based evolutionary computation technique. It is motivated by the behavior of organisms such as bird flocking and fish schooling. 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 g_{best} model in the literature. On the other hand, based on the local variant so called the p_{best} model, each particle moves towards its best previous position and towards the best particle in its restricted neighborhood. PSO is basically characterized as a simple heuristic of well-balanced mechanism with flexibility to progress and adjust to both global and local exploration capabilities. Compared with GA, all the particles tend to converge to the best solution quickly even in the local version in most cases. PSO does not require that the optimization problem be differentiable as is required by classical optimization methods such as gradient descent and quasi-Newton methods. PSO can, therefore, also be used on optimization problems that are partially irregular and noisy. In PSO, the initial population is generated randomly and parameters are initialized. After evaluation of the fitness function, the PSO algorithm repeats the following steps iteratively:

- Personal best (best value of each individual so far) is updated if a better value is discovered.
- Then, the velocities of all the particles are updated based on the experiences of personal best and the global best in order to update the position of each particle with the velocities currently updated.
- Permutation is determined through an encoding scheme so that evaluation is again performed to compute the fitness of the particles in the swarm.

After finding the personal best and global best values, velocities and positions of each particle are updated using equations. 3.7 and 3.8 respectively.

$$v_{ij}^{t} = w^{t-1}v_{ij}^{t-1} + c_1 r_1\left(p_{ij}^{t-1} - x_{ij}^{t-1}\right) + c_2 r_2\left(g_{ij}^{t-1} - x_{ij}^{t-1}\right) \quad (3.7)$$

$$x_{ij}^{t} = x_{ij}^{t-1} + v_{ij}^{t} \quad (3.8)$$

Where v_{ij}^{t} represents velocity of particle i at iteration t with respect to j^{th} dimension (j =1,2,……n). p_{ij}^{t} represents the position value of the i^{th} personal best with respect to the j^{th} dimension.g_{ij}^{t} represents the global best (gbest) i.e the best of pbest among all the particles. x_{ij}^{t} is the position value of the i^{th} particle with respect to j^{th} dimension. c_1 and c_2 are positive acceleration parameters which provide the correct balance between exploration and exploitation, and are called the cognitive parameter and the social parameter, respectively. r_1 and r_2 are the random numbers provide a stochastic characteristic for the particles velocities in order to simulate the real behavior of the birds in a flock. The inertia weight parameter 'w' is a control parameter which is used to control the impact of the previous history of velocities on the current velocity of each particle. Hence, the parameter 'w' regulates the trade-off between global and local exploration ability of the swarm. The recommended value of the inertia weight w is to set it to a large value for the initial stages, in order to enhance the global exploration of the search space, and gradually decrease it to get more refined solutions facilitating the local exploration in the last stages. In general, the inertia weight factor is set according to the following equation 3.9

$$w = w_{max} - \frac{w_{max} - w_{min}}{t_{max}} \times t \quad (3.9)$$

where w_{min}, w_{max} are initial and final weights, 't' is the current iteration number and t_{max} is the maximum number of iterations.

The pseudo-code for the particle swarm optimization

Initialize the parameters, including swarm size, maximum number of iteration, $w_{min}, w_{max}, c_1, c_2$

While (termination condition i.e. maximum Iteration)

Do

t=0

Initialize particle's position and velocity stochastically;

Evaluate each particle's fitness, i.e. the objective function;

Initialize p_{best} position;

Initialize g_{best} position with the particle with lowest fitness in the swarm;

t= t+1;

Update the velocity of the particles by the equation (3.7);

Update the position of the particles by the equation (3.8);

Evaluate each particle's fitness, i.e. the objective function;

Find the new g_{best} and p_{best} value by comparison;

Update g_{best} of the swarm and p_{best} of each particle;

end do

end

3.3 Quantum behaved particle swarm optimization

The main disadvantage of the classical PSO algorithm may be that it does not guarantee global convergence because it is trapped into local optima although it converges fast. The reason being that the velocity vectors assume very small values as iterations proceed. The PSO algorithm has a risk of trapping at local optima and loses its exploration-exploitation ability. Clerc and Kennedy [181] have showed that although PSO is capable of finding a reasonable quality solution much faster than other evolutionary algorithms but it cannot improve the quality of the solution as the number of generations is increased. If p_{best} and g_{best} of a particle are very close to each other then it becomes inactive in the swarm. In other words, when $\left(p_{ij}^{t-1} - x_{ij}^{t-1}\right)$ and $\left(g_{ij}^{t-1} - x_{ij}^{t-1}\right)$ are both small in equation 3.7 and at the same time v_{ij}^{t} has a small value then this particle lose its exploration ability. This could happen at early stages for the gbest particle and as a consequence the PSO is trapped in local minima. To avoid the drawbacks of original PSO, QPSO was proposed stimulated by analysis of the convergence of the PSO. In the quantum PSO, the state of a particle is described by wave function $\Psi(x,t)$ instead of velocity. The dynamic behavior of the particle is broadly divergent from that of the particle in classical PSO systems in that the exact values of 'x' and 'v' cannot be determined simultaneously. We can only learn the probability of the particle's appearing in position x from probability density function $|\Psi(x,t)|^2$, the form of which depends on the potential field the particle lies in and then the probability distribution function of the particle's position can be calculated through the probability density function. Employing the Monte-Carlo method, the particle position is updated according to the following equations,

$$X_{i,(t+1)}^{j} = P_{i,(t+1)}^{j} - \beta * \left(M_{best_t}^{j} - X_{i,t}^{j}\right) * \ln\left(1/u\right) \quad \text{if } k \geq 0.5 \tag{3.10}$$

$$X_{i,(t+1)}^{j} = P_{i,(t+1)}^{j} + \beta * \left(M_{best_t}^{j} - X_{i,t}^{j}\right) * \ln\left(1/u\right) \quad \text{if } k < 0.5 \tag{3.11}$$

$$P_{i,(t+1)}^{j} = \theta * p_{best_{i,t}}^{j} + (1-\theta) * g_{best_t}^{j} \tag{3.12}$$

$$M_{best_t}^{j} = \frac{1}{N}\sum_{i=1}^{N} p_{best_{i,t}}^{j} \quad (3.13)$$

Where P_i is the local attractor, $p_{best_{i,t}}^{j}$ are the best positions which the of particle i^{th} at iteration t with respect to j^{th} dimension has achieved so far and $g_{best_t}^{j}$ the best position of all particles in current generation. $M_{best_t}^{j}$ is the mean best position which is defined as the mean of all the best positions of the population in current generation, k, u, θ are random number distributed uniformly on [0,1]. The parameter β in equation 3.10 and 3.11 is called contraction expansion (CE) coefficient which can be tuned to control the convergence speed of the particle. The starting value of β =1 is used to initially accommodate a more global search and is dynamically reduced to β = 0.4. The idea is to terminate the QPSO algorithm with a better local search. The β value is adaptively allocated as per the equation 3.14

$$\beta = \beta_{max} - [\{(\beta_{max} - \beta_{min})/t_{max}\} * t] \quad (3.14)$$

where β_{max} is the initial contraction expansion factor value, β_{min} is the final contraction expansion factor value , 't' is the current iteration number and t_{max} is the maximum number of iterations.

The pseudo code for the search procedure of the QPSO is as follows.

Initialize the population size, the current position and the dimensions of the particles;

While (termination condition i.e maximum Iteration)

Do

t= t+1;

Compute the mean best position M_{best} by equation (3.13);

Select a value of β by equation (3.14);

for i=1 to population size

for j=1 to dimensions of the particles

θ =rand (0,1);

$P_{i,(t+1)}^{j} = \theta * p_{best_{i,t}}^{j} + (1-\theta) * g_{best_t}^{j}$;

u =rand(0,1);

k =rand(0,1);

If $k \geq 0.5$

$$X_{i,(t+1)}^{j} = P_{i,(t+1)}^{j} - \beta * \left(M_{best_t}^{j} - X_{i,t}^{j}\right) * \ln\left(1/u\right);$$

else

$$X_{i,(t+1)}^{j} = P_{i,(t+1)}^{j} + \beta * \left(M_{best_t}^{j} - X_{i,t}^{j}\right) * \ln\left(1/u\right);$$

end if

end for

Evaluate the fitness value of $X_{i,(t+1)}$ that is the objective function

Update the $p_{best_{i,t}}$ and $g_{best_{i,t}}$

end for

end do

end

3.4 Problem representation of flexible flow shop scheduling

For solution representation of flexible flow shop scheduling problem, a real number encoding is used. The fractional part of the number is used to sort the jobs assigned to each machine whereas the integer part represents the machine number to which the job is assigned. The position of the particle represented by a real number is used for encoding. The dimension of the particle is equal to the number of jobs to be processed. Suppose a FFSP has four jobs to be processed in three processing stages. Stage one, two and three are having two, three and four machines respectively. Real numbers between [1, 1+Mj] (j =1, 2, 3) are generated for each stage where M_j is the number of machines in j^{th} stage. The processing times of the jobs on different stages are given as, $P=\begin{bmatrix}7 & 4 & 2\\ 2 & 7 & 3\\ 5 & 3 & 1\\ 4 & 5 & 2\end{bmatrix}$. The initial positions of the particle are generated as four real numbers within the range of 1 to 3 for the 1^{st} stage. The real part of the position values is used for ensuring machine number on which the job is to be processed and fractional part is used to sequence the jobs on the same machine. For stage 1, the position values of four jobs are giving as [1.05, 2.33, 1.45, and 2.67]. At stage one, job 1 and job 3 are assigned to machine 1 and job 2 and job 4 are assigned to machine 2. The process order of jobs to be scheduled on the same machine depends on the value of fractional parts.

The job is sequenced according to the ascending order of the fractional part which is processed by the same machine. The order of jobs to be scheduled on machine1 is job 3 followed by job 1 because the fractional part of job 3 is greater than the fractional part of job 1. At stage j (j >1), if two jobs are assigned to the same machine, the jobs are assigned according to the completion time in the (j-1) stage to schedule its processed sequence. In other words, the job which completes first in the former stage will be processed first. If the completion time of the former stage is same then values of fractional parts of the particle positions are compared. The job whose value of fractional

parts is smaller will be processed first. If the values are also equal then the job processing sequence is randomly chosen. The encoding scheme is given in Figure 3.2 and the Gantt chart for a possible is shown in Figure 3.3

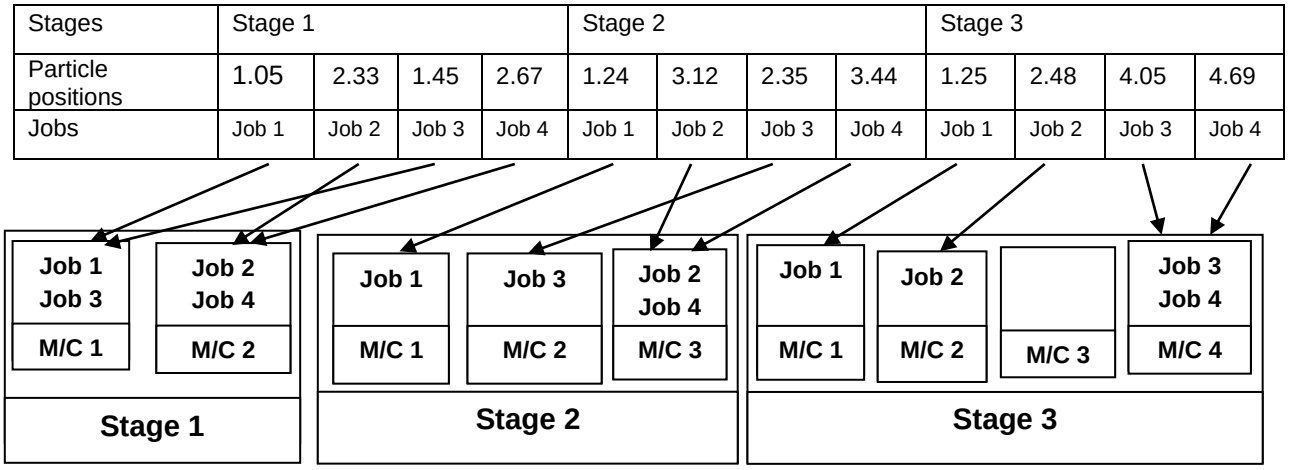


Figure 3.2 Problem representations for the example problem

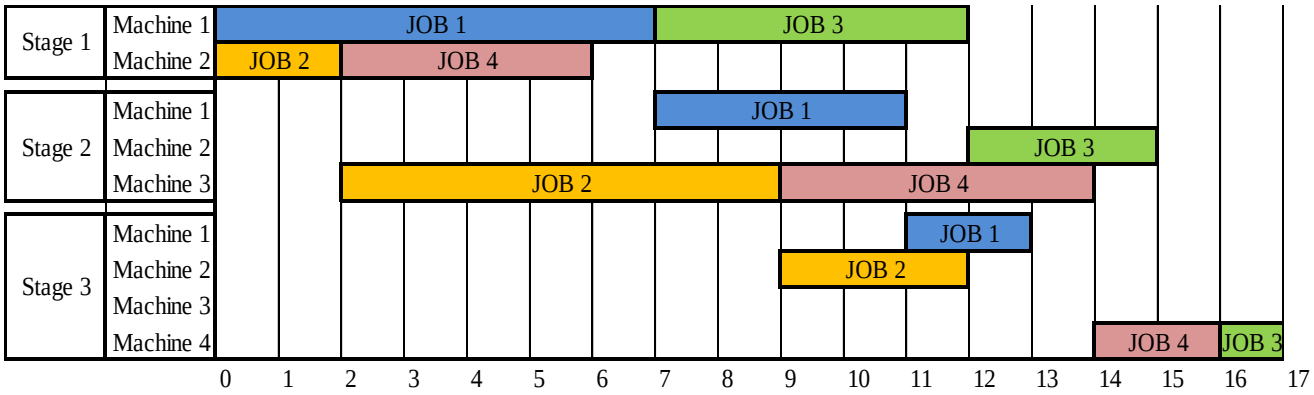


Figure 3.3 Gantt chart for the example problem

3.5 Chaotic numbers

Various versions of PSO have been presented in last few years. Most PSO algorithms use uniform distribution to generate random numbers. Chaos is a kind of characteristic of nonlinear systems, which is a bounded unstable dynamic behavior that exhibits sensitive dependence on initial conditions and includes infinite unstable periodic motions [186,187]. Recently, chaotic sequences have been adopted instead of random sequences to obtain good results in many applications like machine loading problem in flexible manufacturing systems [184]. The purpose of chaotic sequences as a substitute of random sequences in PSO is a powerful strategy to diversify the PSO population and

improve its performance in preventing premature convergence to local minima. Various chaotic time series sequences such as logistic map, tent map, Henon map, Ikeda map, Chua's system, Lorenz's systems, and Lozi's map are available [188,189,190].Recently, several attempts were made for PSO using chaos methods based on logistic map to overcome the drawbacks of PSO technique of premature convergence [183,191,192,193,194].

In most of the stochastic search algorithms, random number generators (RNGs) have been widely used. RNGs are known for slow convergence and have an inherent characteristic of sticking to a solution. To overcome this difficulty, chaotic generators have recently been used instead of RNGs. The following equation is used to replace RNGs [182].

$$N(t)= R^* N(t-1)*(1-N(t-1)) \quad (3.15)$$

The logistic map illustrated in equation (3.15) is one of the simplest dynamic systems which demonstrates chaotic behavior where N(t) is the value of chaotic variable in t^{th} iteration and R shows the bifurcation parameter of the system [186]. In Figure 3.4, a comparison is made between chaotic numbers and random numbers. The above formula was coded in Matlab 7 to generate two hundred chaotic numbers between 0 and 1. Here, R and N are initialized to 4 and 0.1 respectively while rand () function is used to generate two hundred random numbers. The function rand () returns a random value between 0 and RAND_MAX, where RAND_MAX is maximum value that can be stored by an integer variable. The numbers were divided by RAND_MAX to get two hundred random numbers between 0 and 1. From Figure 3.4, it can be observed that chaotic numbers have higher degree of disorder which facilitates high diversity in the particles and thus helps the algorithm to converge rapidly towards the solution.

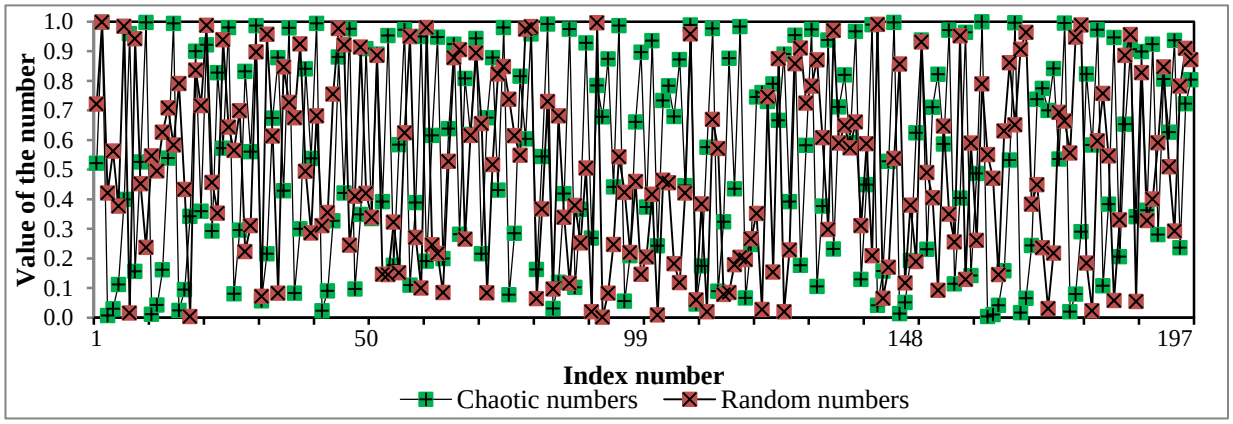


Figure 3.4 Comparison between random numbers and chaotic numbers

3.6 Mutation strategy

Particle swarm optimization typically converges relatively rapidly in the first part of the search and then slows down or stops. This behavior has been attributed to the loss of diversity in the population and a number of researchers have suggested methods to overcome this drawback with varying degrees of success [195]. Looking at the positions of the particles when the swarm had stagnated, it is clear that the points were very tightly clustered and the velocities were almost zero. The points were often not that far from the global optimum but the updating equations, due to the almost zero velocity, were unable to generate new solutions which might lead the swarm out of this state. This behavior can also lead to the whole swarm being trapped in a local optimum from which it becomes impossible to escape. As mutation is capable of introducing diversity in the search procedure, two types mutation have attracted the researchers - mutation of global best and mutation based on sharing information from neighbors. Because the global best individual attracts all members of the swarm, it is possible to lead the swarm away from a current location by mutating a single individual if the mutated individual becomes the new global best. This mechanism potentially provides a means both escaping local optima and speeding up the search.

In this chapter, a mutation operator is introduced which mutates some particles selected randomly from the swarm. Mutation is not carried out each time, the mutation process will begin if the number of iteration is less than the product of maximum number of iteration and probability of mutation then only the mutation is performed on the position of the particle. Given a particle, a randomly chosen variable, say $\mathrm{m_p}$, is mutated to assume a value $\mathrm{m'_p}$ as given by following equation.

$$\mathrm{m'_p} = \begin{cases} \mathrm{m_p} + \Delta(\mathrm{y}, \mathrm{UB} - \mathrm{m_p}) & \text{if flip} = 0 \\ \mathrm{m_p} - \Delta(\mathrm{y}, \mathrm{m_p} - \mathrm{LB}) & \text{if flip} = 1 \end{cases} \tag{3.16}$$

when flip denotes the random event of returning 0 or 1. UB and LB denote the upper and lower bound of the variable $\mathrm{m_p}$ respectively. The function $\Delta(\mathrm{y}, \mathrm{x})$ returns a value in the range [0, x] such that the probability of $\Delta(\mathrm{y}, \mathrm{x})$ being close to 0 increases as y increases.

$$\Delta(\mathrm{y}, \mathrm{x}) = \mathrm{x} \times \left(1 - \mathrm{r}^{\left(1 - \frac{\mathrm{t}}{\mathrm{t_{max}}}\right)^{\mathrm{b}}}\right) \tag{3.17}$$

where r is the random number generated in the range [0, 1], $\mathrm{t_{max}}$ is the maximum number of iterations and t is the number of iteration. The parameter b determines the degree of dependence of mutation on the iteration number. There are two reasons for

adopting this method: (i) if there is no premature convergence (ii) The algorithm will not increase computational overhead much.

3.7 Proposed particle swarm optimization algorithm for FFSP

The proposed methodology introduces an improved algorithm by combining PSO with chaotic numbers and mutation operator is embedded to escape from local optima and to improve the solution diversity. Various steps involved in the proposed PSO algorithm are listed out below.

Step 1. Initialize the parameters such as population size, maximum iteration, decrement factor, inertia weight, social and cognitive parameters.

Step 2. Input number of jobs, number of stages, and number of machines at each stage, and processing times.

Step 3. Generate the initial position values of the particle: $x_{ij}^{t} = x_{min} + (x_{max} - x_{min}) \times N(0,1)$ where $x_{min} = 1.0$, $x_{max} = 1+M_j$ and N(0,1) is a chaotic number between 0 and 1. Generate initial velocities of the particle $v_{ij}^{t} = v_{min} + (v_{max} - v_{min}) \times N(0,1)$ where $v_{min} = -4.0$, $v_{max} = 4.0$ and N(0,1) is a chaotic number between 0 and 1 where M_j is the number of machines in j^{th} stage

Step 4. Get the schedule using encoding scheme as mentioned in Section 3.4.

Step 5. Evaluate each particle's fitness (makespan).

Step 6. Find out the personal best (p_{best}) and global best (g_{best}).

Step 7. If (t< (t_{max} * PMUT), then perform mutation on x_{ij}^{t}.
(PMUT is the probability of mutation)

Step 8. Update velocity, position and inertia weight by using equations (3.7), (3.8) and (3.9). All the random number used in equation 3.7 is replaced by chaotic number.

Step 9. Terminate if maximum number of iterations is reached and store the g_{best} value. Otherwise, go to Step 3.

Step 10. End

3.8 Proposed quantum-behaved particle swarm optimization algorithm for FFSP

The proposed methodology introduces an algorithm by combining QPSO with chaotic numbers and mutation operator is embedded to escape from local optima and to improve the solution diversity. Various steps involved in the proposed QPSO algorithm are listed out below

Step 1. Initialize the parameters such as population size, maximum iteration, k, u, θ with chaotic number between 0 and 1.

Step 2. Input number of jobs, number of stages, and number of machines at each stage, and processing times.

Step 3. Initialization of swarm positions: Generate the initial position values of the particle : $x_{ij}^{t} = x_{min} + (x_{max} - x_{min}) \times N(0,1)$ where x_{min} = 1.0, x_{max} =1+M_j and N(0,1) is a chaotic number between 0 and 1.

Step 4. Get the schedule using encoding scheme as mentioned in section 3.4.

Step 5. Evaluate each particle's fitness (makespan).

Step 6. Comparison to p_{best} (personal best): Compare each particle's fitness with the particle's p_{best}.

Step 7. Comparison to g_{best} (global best): Compare the fitness with the population's overall previous best.

Step 8. If (t< (t_{max} * PMUT), then perform mutation on x_{ij}^{t}.
(PMUT is the probability of mutation)

Step 9. Then calculate the mean value of the best position (M_{best}) using Eq. 3.13 and generate the Contraction-Expansion factor using equation. 3.14.

Step 10. Update the positions of all particles according to equations.3.10 and 3.11

Step 11. Repeat the cycle: loop to Step 4 till the stop criterion is met or terminate if maximum number of iterations is reached and store the g_{best} value.

Step 12. END

3.9 Results and discussions

The computational study aims to analyze the performance of the proposed PSO and QPSO to minimize the makespan for the flexible flow shop scheduling problems. The algorithm was implemented in Matlab 7 on a Pentium IV running at 2 GHz on the Windows XP operating system. The benchmark problems are taken from Carlier and Neron's (2000) [196].The Carlier and Neron's [196] problem sizes vary from 10 jobs×5 stages to 15 jobs×10 stages. The processing times of jobs are uniformly distributed between 3 to 20. Three characteristics are used to represent a problem, which are the number of jobs, the number of stages, and the number of identical machines at each stage. For instance, the notation of j10c5a2 means a 10-job, 5-stage problem. The letters j and c denote job and stage, respectively. Letter a defines the structure of the machine distribution at the stages. The last number 2 is the instance index for a specific type.

Figure 3.5 illustrates the comparison of the performance of the random and chaotic numbers for an instance j15c10a1. It is conceivable to note that QPSO algorithm with

chaotic numbers improves the makespan and that the best makespan, equal to 236, is reached after 88 iterations. It can be observed from Figure 3.5 that the solution converges towards the best value faster when the chaotic numbers are used. This is because of higher degree of disorderness of the chaotic numbers which facilitates high diversity in the particles and helps the algorithm to converge rapidly towards the solution.

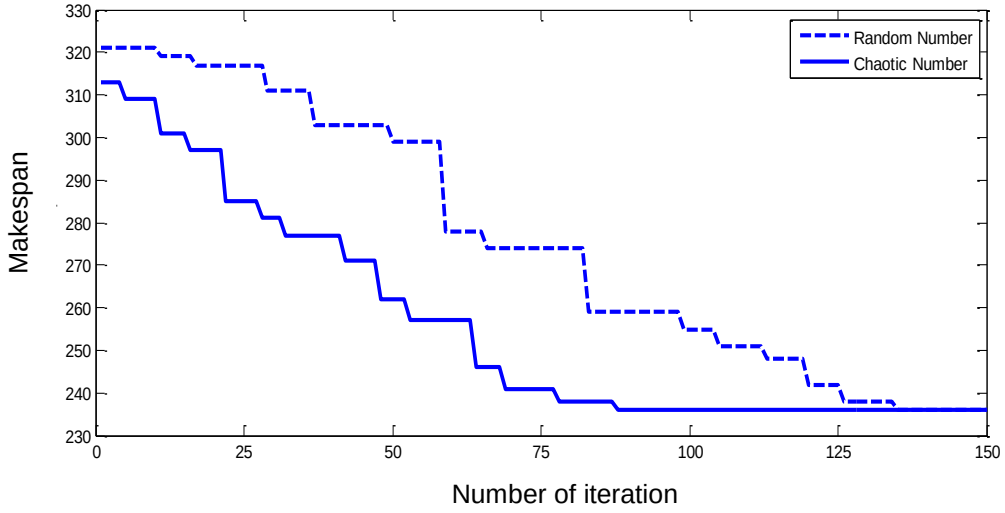


Figure 3.5 The convergence curve of the problem j15c10 a1

In order to validate the performance of the proposed methods, the results are compared with the earlier studies of Kahraman et al [108], Neron et al's[97],Engin and Doyen [20] and Alaykiran et al. [19]. The Table 1 represents the comparison of the best makespan value obtained by our proposed PSO and QPSO with the GA model proposed by Kahraman et al [108], artificial immune system (AIS) model proposed by Engin and Doyen's [20], branch and bound (B&B) model proposed by Neron et al's [97] and ant colony optimization model (ACO) proposed by Alaykiran et al. [19].

In the Table 3.1 the first columns symbolize the name of the problem. The second column shows the Lower Bound of the makespan for the problem. The third and fourth column refers to the best makespan result from the proposed PSO and QPSO algorithm. The fifth column up to the eight one represent the best makespan resulted from GA, AIS, ACO, B&B respectively. Relative deviation criterion is used to compare the results of the proposed QPSO with those of the above five mentioned algorithms. The results are represented in terms of percentage deviation (PD) of the solution from the lower bound (LB) [108].Percentage Deviation (%PD) is defined in the following relation.

$$\%\mathrm{PD} = \frac{\mathrm{Best}\ C_{\max} - \mathrm{lowerbound}}{\mathrm{lowerbound}} \times 100 \qquad (3.18)$$

Table 3.1 The computational results

Problem	LB	Proposed PSO		Proposed QPSO		GA [108]		AIS [20]		ACO [19]		B&B [97]	
		C_{max}	% PD	C_{max}	% PD	C_{max}	% PD	C_{max}	% PD	C_{max}	% PD	C_{max}	% PD
j10c5a2	88	88	0	88	0	88	0	88	0	88	0	88	0
j10c5a3	117	117	0	117	0	117	0	117	0	117	0	117	0
j10c5a4	121	121	0	121	0	121	0	121	0	121	0	121	0
j10c5a5	122	122	0	122	0	122	0	122	0	124	1.639	122	0
j10c5a6	110	110	0	110	0	110	0	110	0	110	0	110	0
j10c5b1	130	130	0	130	0	130	0	130	0	131	0.769	130	0
j10c5b2	107	107	0	107	0	107	0	107	0	107	0	107	0
j10c5b3	109	109	0	109	0	109	0	109	0	109	0	109	0
j10c5b4	122	122	0	122	0	122	0	122	0	124	1.639	122	0
j10c5b5	153	153	0	153	0	153	0	153	0	153	0	153	0
j10c5b6	115	115	0	115	0	115	0	115	0	115	0	115	0
j10c5c1	68	68	0	68	0	68	0	68	0	68	0	68	0
j10c5c2	74	74	0	74	0	74	0	74	0	76	2.703	74	0
j10c5c3	71	71	0	71	0	71	0	72	1.408	72	1.408	71	0
j10c5c4	66	66	0	66	0	66	0	66	0	66	0	66	0
j10c5c5	78	78	0	78	0	78	0	78	0	78	0	78	0
j10c5c6	69	69	0	69	0	69	0	69	0	69	0	69	0
j10c5d1	66	66	0	66	0	66	0	66	0	NA		66	0
j10c5d2	73	73	0	73	0	73	0	73	0	NA		73	0
j10c5d3	64	64	0	64	0	64	0	64	0	NA		64	0
j10c5d4	70	70	0	70	0	70	0	70	0	NA		70	0
j10c5d5	66	66	0	66	0	66	0	66	0	NA		66	0
j10c5d6	62	62	0	62	0	62	0	62	0	NA		62	0
j10c10a1	139	139	0	139	0	139	0	139	0	NA		139	0
j10c10a2	158	158	0	158	0	158	0	158	0	NA		158	0
j10c10a3	148	148	0	148	0	148	0	148	0	NA		148	0
j10c10a4	149	149	0	149	0	149	0	149	0	NA		149	0
j10c10a5	148	148	0	148	0	148	0	148	0	NA		148	0
j10c10a6	146	146	0	146	0	146	0	146	0	NA		146	0
j10c10b1	163	163	0	163	0	163	0	163	0	163	0	163	0
j10c10b2	157	157	0	157	0	157	0	157	0	157	0	157	0
j10c10b3	169	169	0	169	0	169	0	169	0	169	0	169	0
j10c10b4	159	159	0	159	0	159	0	159	0	159	0	159	0
j10c10b5	165	165	0	165	0	165	0	165	0	165	0	165	0
j10c10b6	165	165	0	165	0	165	0	165	0	165	0	165	0
j10c10c1	113	115	1.77	115	1.77	115	1.77	115	1.77	118	4.425	127	12.39
j10c10c2	116	117	0.862	116	0	117	0.862	119	2.586	117	0.862	116	0
j10c10c3	98	113	15.31	108	10.2	116	18.37	116	18.37	108	10.2	133	35.71
j10c10c4	103	112	8.738	110	6.796	120	16.5	120	16.5	112	8.738	135	31.07
j10c10c5	121	125	3.306	122	0.826	125	3.306	126	4.132	126	4.132	145	19.83
j10c10c6	97	103	6.186	102	5.155	106	9.278	106	9.278	102	5.155	112	15.46
j15c5a1	178	178	0	178	0	178	0	178	0	178	0	178	0
j15c5a2	165	165	0	165	0	165	0	165	0	165	0	165	0
j15c5a3	130	130	0	130	0	130	0	130	0	132	1.538	130	0
j15c5a4	156	156	0	156	0	156	0	156	0	156	0	156	0
j15c5a5	164	164	0	164	0	164	0	164	0	166	1.22	164	0
j15c5a6	178	178	0	178	0	178	0	178	0	178	0	178	0
j15c5b1	170	170	0	170	0	170	0	170	0	170	0	170	0
j15c5b2	152	152	0	152	0	152	0	152	0	152	0	152	0
j15c5b3	157	157	0	157	0	157	0	157	0	157	0	157	0
j15c5b4	147	147	0	147	0	147	0	147	0	149	1.361	147	0
j15c5b5	166	166	0	166	0	166	0	166	0	166	0	166	0
j5c5b6	175	175	0	175	0	175	0	175	0	176	0.571	175	0
j15c5c1	85	85	0	85	0	85	0	85	0	85	0	85	0
j15c5c2	90	90	0	90	0	91	1.111	91	1.111	90	0	90	0

j15c5c3	87	87	0	87	0	87	0	87	0	87	0	87	0
j15c5c4	89	89	0	89	0	89	0	89	0	89	0	90	1.124
j15c5c5	73	74	1.37	73	0	75	2.74	74	1.37	73	0	84	15.07
j15c5c6	91	91	0	91	0	91	0	91	0	91	0	91	0
j15c5d1	167	167	0	167	0	167	0	167	0	167	0	167	0
j15c5d2	82	84	2.439	82	0	84	2.439	84	2.439	86	4.878	85	3.659
j15c5d3	77	83	7.792	80	3.896	83	7.792	83	7.792	83	7.792	96	24.68
j15c5d4	61	81	32.79	77	26.23	84	37.7	84	37.7	84	37.7	101	65.57
j15c5d5	67	80	19.4	80	19.4	80	19.4	80	19.4	80	19.4	97	44.78
j15c5d6	79	82	3.797	81	2.532	82	3.797	82	3.797	79	0	87	10.13
j15c10a1	236	236	0	236	0	236	0	236	0	236	0	236	0
j15c10a2	200	200	0	200	0	200	0	200	0	200	0	200	0
j15c10a3	198	198	0	198	0	198	0	198	0	198	0	198	0
j15c10a4	225	225	0	225	0	225	0	225	0	228	1.333	225	0
j15c10a5	182	182	0	182	0	182	0	182	0	182	0	183	0.549
j15c10a6	200	200	0	200	0	200	0	200	0	200	0	200	0
j15c10b1	222	222	0	222	0	222	0	222	0	222	0	222	0
j15c10b2	187	190	1.604	189	1.07	187	0	187	0	188	0.535	187	0
j15c10b3	222	222	0	222	0	222	0	222	0	224	0.901	222	0
j15c10b4	221	221	0	221	0	221	0	221	0	221	0	221	0
j15c10b5	200	200	0	200	0	200	0	200	0	NA		200	0
j15c10b6	219	219	0	219	0	219	0	219	0	NA		219	0
Average percentage deviation (APD)			1.368		1.012		1.624		1.659		1.443		3.634

For seventy seven benchmark problems solved here, it is observed from the Table 1 that most of the makespan values by the proposed PSO and QPSO algorithm have smaller PD over GA, AIS, ACO and B&B. Again it is noticeable that proposed QPSO performs better than the proposed PSO algorithm. Average percentage deviation (APD) of the proposed QPSO is also compared with proposed PSO, GA, AIS, ACO and B&B for benchmark problems of flexible flow shop scheduling problem. The Average percentage deviation (APD) is defined as follow.

$$\mathrm{APD} = \frac{\sum_{L=1}^{I} \mathrm{PD(L)}}{I} \tag{3.19}$$

Where I is the total number of problems and L stands for index of the problem. The APD for proposed PSO, GA, AIS, ACO and B&B are 1.368, 1.624, 1.659, 1.443 and 3.634 respectively. The APD for proposed QPSO is obtained as 1.012 which happens to be superior as compared to other algorithms. The performance of B&B is worst among all the algorithms.

The improvement rate for APD using QPSO is defined as follows:

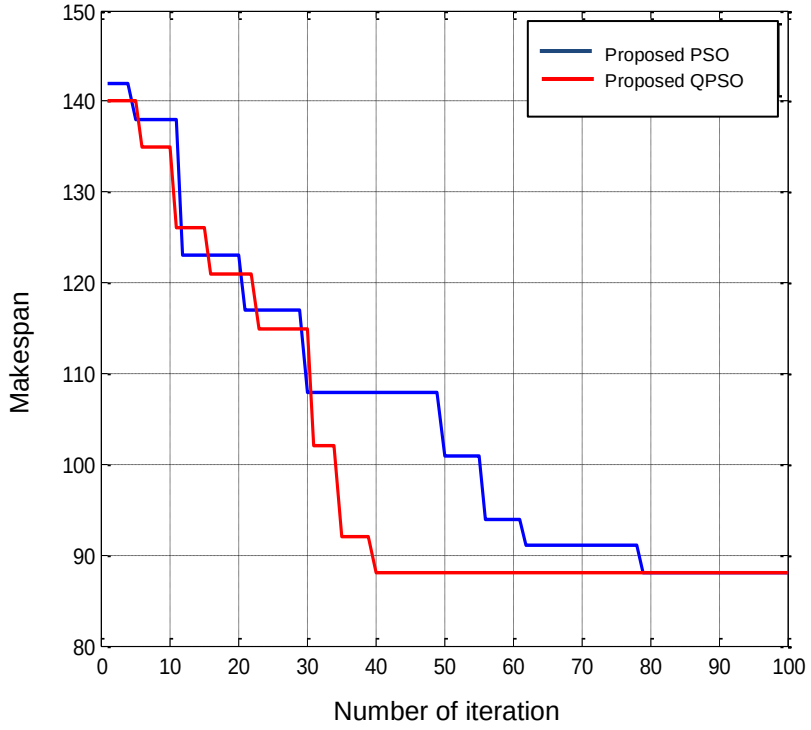
$$\text{Improvement rate } (\%) = \frac{(\mathrm{APD_{PSO,GA,AIS,B\&B}} - \mathrm{APD_{QPSO}})}{\mathrm{APD_{PSO,GA,AIS,B\&B}}} \tag{3.20}$$

The improvement rate of average percentage deviation (APD) of proposed QPSO is found to be 26.0812 % with respect to proposed PSO, 37.73% with respect to GA, 38.95% with respect to AIS, 29.946% with respect to ACO and 72.18 % with respect to

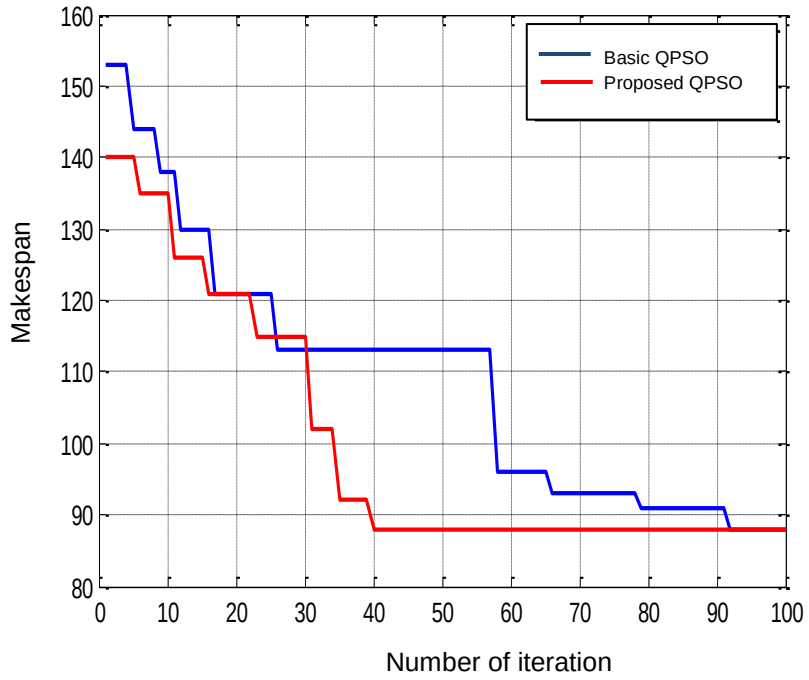
branch and bound (B&B) for seventy seven benchmark problems considered in the study.

The convergence curve is drawn for different data set to compare the proposed QPSO and proposed PSO algorithm. The Figure 3.6 (a) to 3.9 (a) shows the convergence of the problems 10jobs 5stages, 10jobs 10stages, 15jobs 5stages, 15jobs 10stages. It is conceivable to note that proposed QPSO algorithm perform better than the proposed PSO algorithm as it converges as a faster rate as compared to proposed PSO algorithm. The Figure 3.6 (b) to 3.9 (b) illustrates the performance of basic QPSO and the proposed QPSO for the benchmark problems. It can be observed from Figure 3.6 (b) to 3.9 (b) that the solution converges towards the best value faster when the chaotic numbers are used for initialization of the particle position and mutation is used to avoid premature convergence.

.

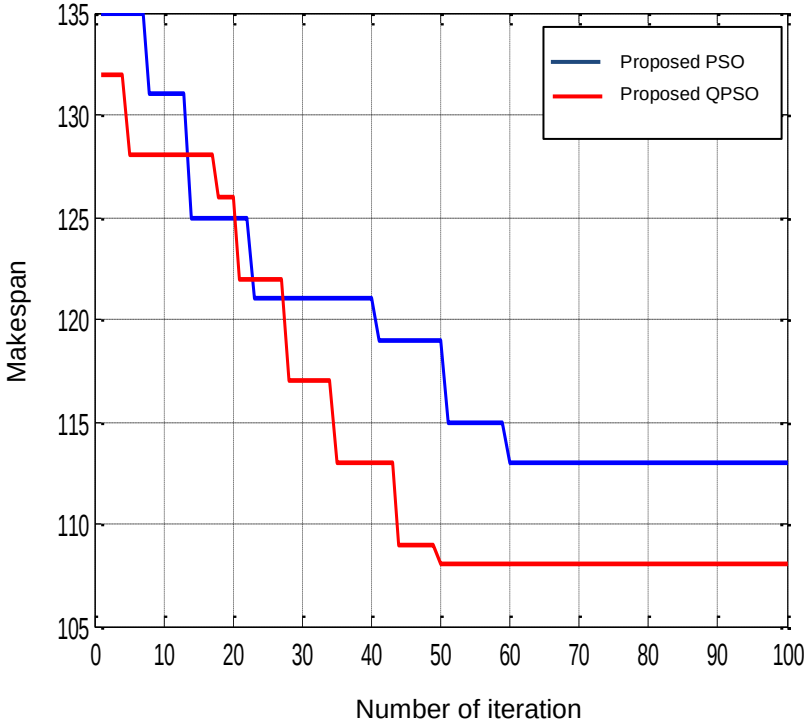


(a)

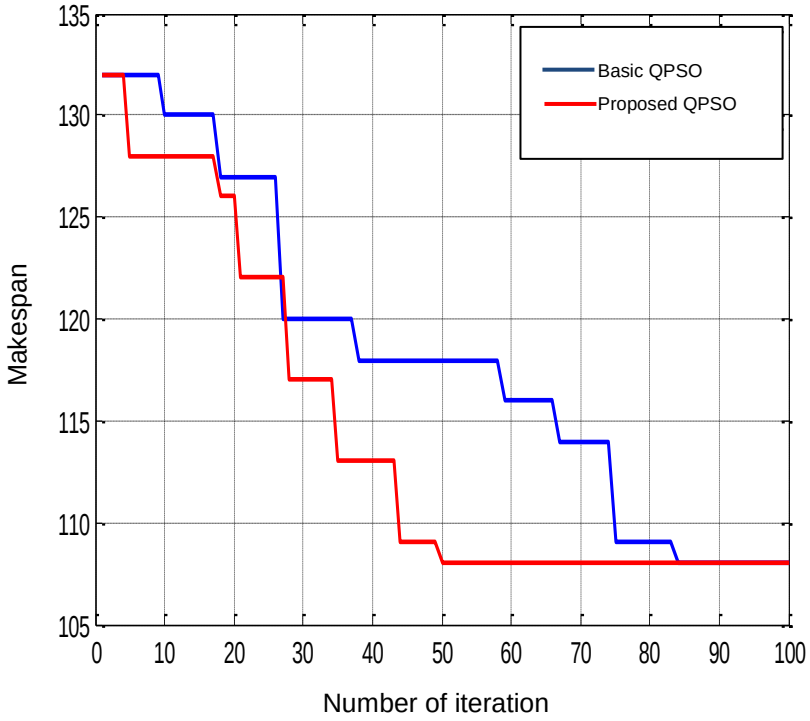


(b)

Figure 3.6 The convergence curve for 10 jobs 5stages. (Problem j10c5a2)

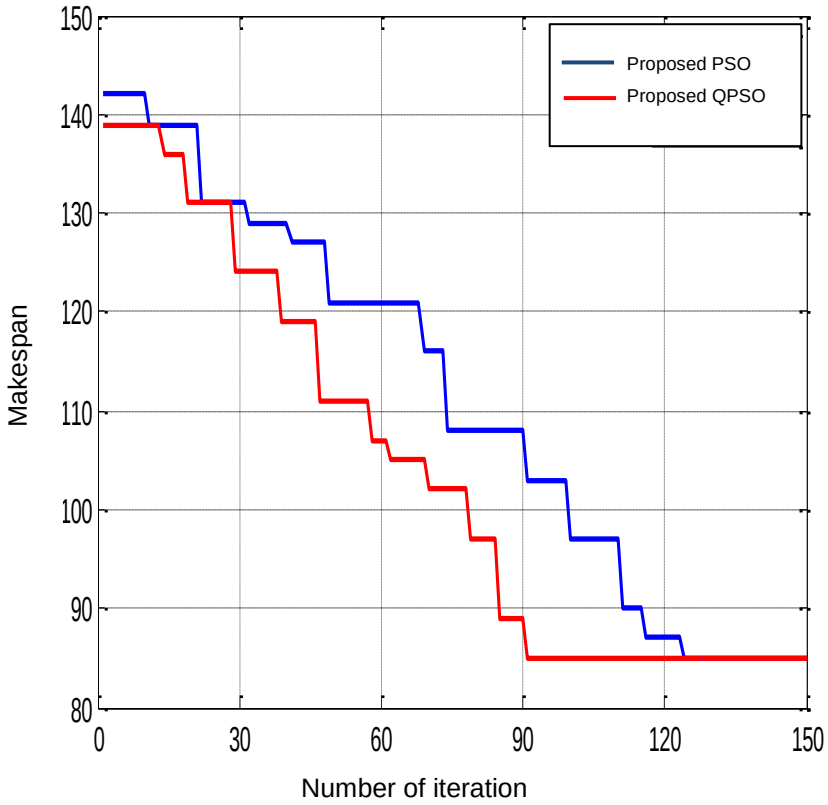


(a)

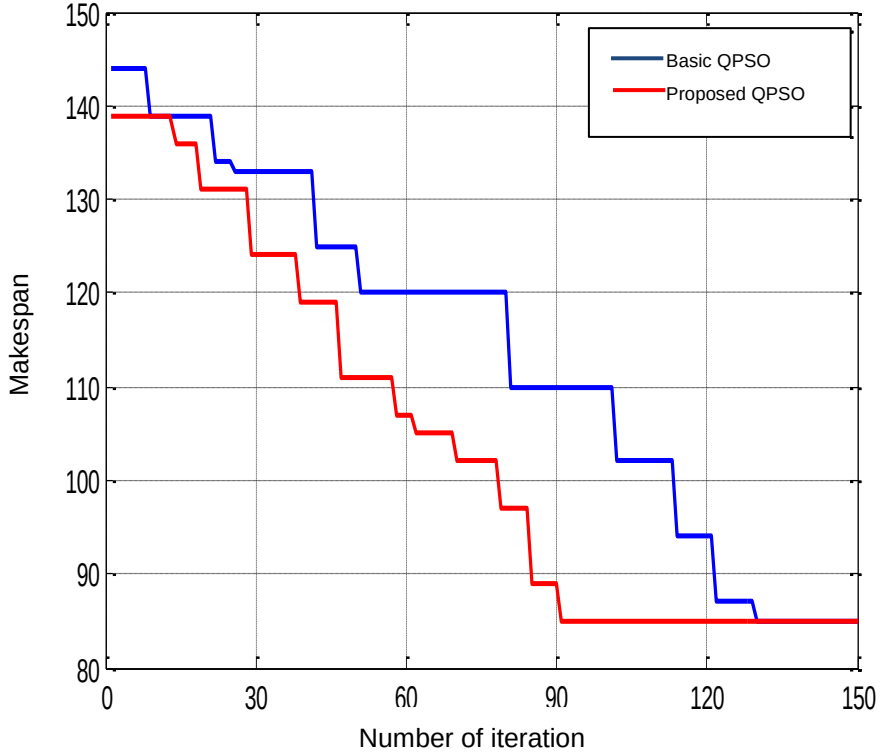


(b)

Figure 3.7 The convergence curve for 10 jobs 10stages. (Problem j10c10c3)

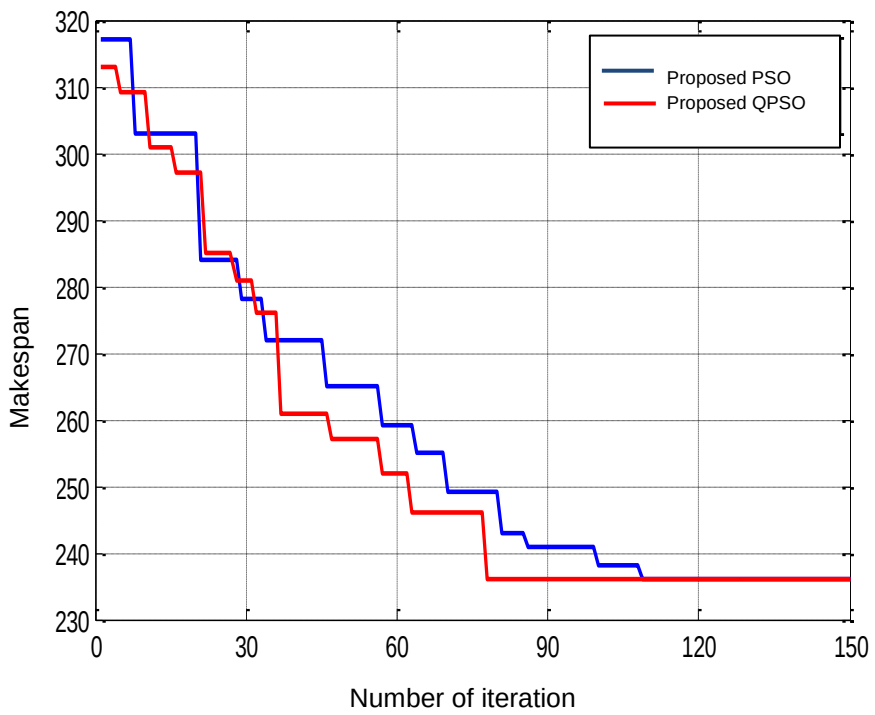


(a)

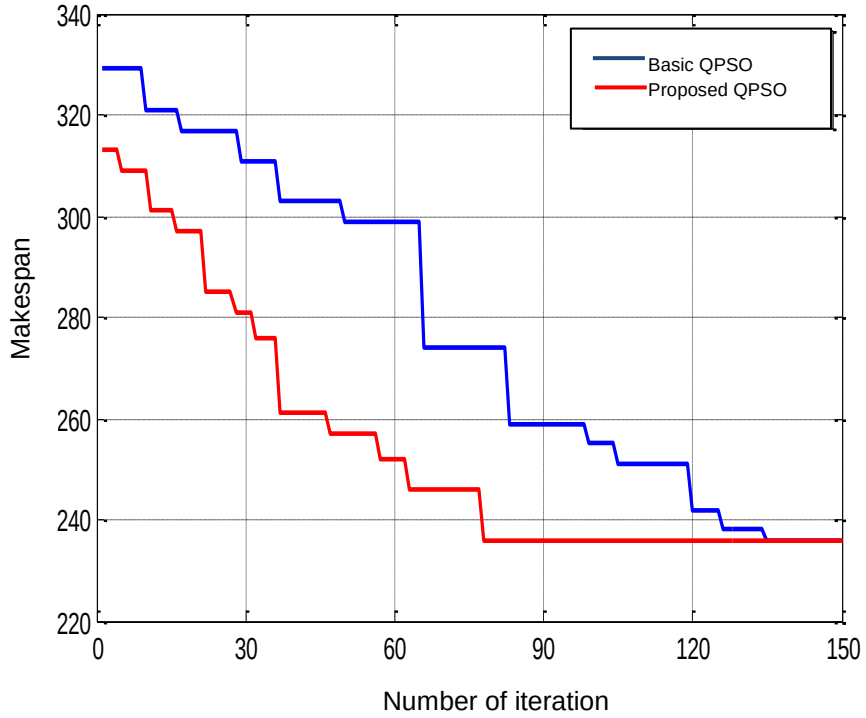


(b)

Figure 3.8 The convergence curve for 15 jobs 5stages. (Problem j15c5c1)



(a)



(b)

Figure 3.9 The convergence curve for 15 jobs 10stages. (Problem j15c10a1)

3.10 Conclusions

In this chapter, flexible flow shop scheduling problem which is NP-hard is considered and an efficient quantum particle swarm optimization on the value of an optimal scheduled as developed. A mutation operator used in genetic algorithm is introduced in PSO and QPSO for the solution to improve the solution diversity, which to accelerate convergence rate strategy and improve solution diversity. For preventing the premature convergence to local minima of the algorithm is improved through the use of chaotic numbers (Logistic map) instead of random numbers. Thereby, computational efforts can be reduced by a large extent. The proposed approach is tested on a set of seventy seven instances taken from the literature given by Carlier and Neron's [196].The comparison was made with the flexible flow shop scheduling problems with proposed PSO and proposed QPSO, the GA model proposed by Kahraman et al [108], AIS model proposed by Engin and Doyen's [20], B&B model proposed by Neron et al's [97] and ACO proposed by Alaykiran et al. [19]. The obtained results are encouraging in that the proposed QPSO algorithm gives smaller percentage deviation from lower bound over proposed PSO, GA, AIS, ACO and B&B. The improvement rate of average percentage deviation (APD) of proposed QPSO is found to be 26.0812 % with respect to proposed PSO, 37.73% with respect to GA, 38.95% with respect to AIS, 29.946% with respect to ACO and 72.18 % with respect to branch and bound (B&B) for seventy seven benchmark problems considered in the study. The advantage of QPSO algorithm lies in the fact that it requires less number of parameters to be controlled at arrive at good solutions as compared to other population based methods. The QPSO have been effectively tackled through use of mutation inspired from genetic algorithm to make it more efficient. The chaotic number used in the work provides solution diversity and reduces computational burden. The proposed QPSO approach is a good problem solving technique for a flexible flow shop scheduling problem. The next chapter briefly discusses on the flexible job shop scheduling. The proposed particle swarm optimization (PSO) and quantum particle swarm optimization (QPSO) have been implemented to determine the optimum schedule.

CHAPTER 4

FLEXIBLE JOB SHOP SCHEDULING

4.1 Introduction

A classical job-shop scheduling problem (JSP) deals with a set of n jobs to be processed by a set of machines. Each job is processed on machines in a given order with a given processing time and each machine can process only one job at a time. In contrast, the flexible job-shop scheduling problem (FJSP) is a generalization of the classical JSP, where operations are allowed to be processed on any among a set of available machines. In fact, the FJSP mainly presents two difficulties. The first one is to assign each operation to a machine out of a set of capable machines, and the second one is to sequence the assigned operations on all machines. The FJSP is visualized as more complex version of the JSS problem [115,117]. Therefore, it is treated as a strongly NP-hard combinatorial problem and poses difficulty in solving by exact methods as computation time increases unexpectedly [65]. Instead of searching an optimal solution with high computational effort, it is prudent to use an efficient heuristic method to generate approximate solutions close to the optimum with considerably less computational time. In recent years, a large variety of heuristic and meta-heuristics have been extensively applied to solve the challenging FJSP because of its computational complexity. The heuristic procedures such as dispatching rules, local search, and meta-heuristic procedures are used to solve such problems and generate approximate solutions close to the optimum with considerably less computational time. These methods can be classified into two main categories: hierarchical approach and integrated approach. In hierarchical approaches, the assignment of operations to machines and the sequencing of operations on the machines are treated separately. In effect, hierarchical approach is based on the idea of decomposing the original problem in order to reduce complexity. Brandimarte [112] has applied hierarchical approach for FJSP based on decomposition. First, the routing sub-problem is solved using dispatching rules and then the sequencing sub-problem is solved by using TS algorithm. Kacem et al. [117] have proposed a localization approach to solve the resource assignment problem and an evolutionary approach controlled by the assignment model for the FJSP. Integrated approaches consider both assignment and sequencing sub-problems simultaneously. Usually, integrated approaches produce better solutions than hierarchical approaches but more difficult to solve [118].

The flexible job shop problem is to organize the execution of ‘n’ jobs on any among a set of available ‘m’ machines at a facility. Flexible job shop has ‘p’ possible work centers and each work center consists of a set of m machines in parallel from which one

machine is chosen to perform the task or operation. All jobs and machines are available at time zero and a machine can only execute one operation at a given time. Preemption is not allowed i.e. each operation must be completed without interruption once it starts. The FJSP is machine dependent because the performance of each operation on each allowable machine has a different processing time. The objective of the problem is to assign each operation to an appropriate machine and to sequence the operations on the machines in order to minimize the makespan which is the time required completing all the jobs. The FJSP scheduling problem can be formulated mathematically as mixed integer linear programming (MILP) problem as follows [119]. The symbols used are as follows:

n: Total number of jobs

m: Total number of machines

O_{ij}: The j^{th} operation of job i

M_{ij} : Set of capable machine is assigned to operation O_{ij}

J_i : Total number of operations of job i

P_{ijk}: Processing time of O_{ijk} if performed on machine k

M : A large number (either or constraint)

C_{ij} : Completion time of operation O_{ij}

S_{ijk}: Start time of operation O_{ij} on machine k

C_{ijk}: Completion time of operation O_{ij} on machine k

C_i: Completion time of job i

C_{max}: Makespan

i, h : Index of jobs, $i, h = 1,2, \dots, n$

k: Index of machines, $k = 1,2, \dots, m$

j, g : Index of operations, $j, g = 1,2, \dots, J_i$

Decision Variables:-

$$X_{ijk} : \begin{cases} 1, & \text{if } O_{ij} \text{ is performed on machine } k \\ 0, & \text{otherwise} \end{cases}$$

$$Z_{ijhgk} : \begin{cases} 1, & \text{if operation } O_{ij} \text{ precedes operation } O_{hg} \text{ on machine } k \\ 0, & \text{otherwise} \end{cases}$$

Objective:

$$C_{max} \geq C_i \quad \forall i \tag{4.1}$$

Subject to:

$$C_i \geq \sum_{k \in M_{ij}} C_{ijk} \qquad \forall i, j = J_i \tag{4.2}$$

$$S_{ijk} + C_{ijk} \leq X_{ijk} * M \qquad \forall i, j, k \in M_{ij} \tag{4.3}$$

$$C_{ijk} \geq S_{ijk} + P_{ijk} - (1 - X_{ijk}) * M \qquad \forall i, j, k \in M_{ij} \tag{4.4}$$

$$S_{ijk} \geq C_{hgk} - (Z_{ijhgk}) * M \qquad \forall i \leq h, \forall j, \forall k \in M_{ij} \cap M_{hg} \tag{4.5}$$

$$S_{ijk} \geq C_{hgk} - (1 - Z_{ijhgk}) * M \qquad \forall i \leq h, \forall j, \forall k \in M_{ij} \cap M_{hg} \tag{4.6}$$

$$\sum_{k \in M_{ij}} S_{ijk} \geq \sum_{k \in M_{ij}} C_{ij-1k} \qquad \forall i, \forall j = 2, \dots, J_i \tag{4.7}$$

$$\sum_{k \in M_{ij}} X_{ijk} = 1 \qquad \forall i, j \tag{4.8}$$

$$S_{ijk} \geq 0 \ , \ C_{ijk} \geq 0 \qquad \forall i, j, k \tag{4.9}$$

$$C_{max} \geq C_i \qquad \forall i \tag{4.10}$$

Constraints set (4.2) determine the completion times of the jobs. Constraints set (4.3) and (4.4) imposes that the difference between the starting and the completion times is equal in the least to the processing time on machine k. Constraints sets (4.5) and (4.6) ensure that operation O_{ij} and operation O_{hg} cannot be done at the same time on any machine in the set $M_{ij} \cap M_{hg}$. Constraint set (4.7) ensures that the precedence relationships between the operations of a job are not violated, i.e. the operation O_{ij} is not started before the operation O_{ij-1} has been completed. A constraint set (4.8) ensures that an operation is performed on one and only one machine. Constraint set (4.9) set the starting and completion times of it on machine k equal to zero if operation O_{ij} is not assigned to machine k and constraints (4.10) determine the makespan.

Due to the complexity of FJSP, exact techniques, such as branch and bound [12, 197] and dynamic programming [198,199] are only appropriate to small scale problems. For example, two jobs having two operations each need to be scheduled with three machines then total number of decision variables becomes 36. The number of constraints according to the equation 4.2 is 18 (3 machine^2operation per job× 2job = 18) , according to the equation 4.3 it is 12 (2operation per job × 2job× 3 machine=12), according to the equation 4.4 it is 12 (2operation per job ×2job×3 machine=12), according to the equation 4.5 it is 24 ((2 operation per job × 2job) × 2 job× 3 machines = 24), according to the equation 4.6 it is ((2 operation per job × 2job) × 2 job× 3 machines = 24), according to the equation 4.7 it is 12 (2operation per job ×2job×3 machine=12),

according to the equation 4.8 it is 12 (2operation per job ×2job×3 machine=12), according to the equation 4.9 it is ((2 operation per job ×2job) × 2 job× 3 machines = 24). From the above instance, total of 126 numbers of constraints are found for such a small problem. So the number of constraints will increase exponentially as the number of problem size increases. Therefore, integer programing need exponential computation time in most cases which leads to impractical computational burden for large scale application. Therefore, scheduling of jobs in a flexible job shops is considered as NP-hard problem.

Most of them fail to obtain good solutions solving large scale problems because of the huge memory and lengthy computational time required. Despite the relative success of exact algorithms, they are still incapable of solving medium and large instances and are too complex for real world problems. It is essential to study non-exact but efficient heuristics. Therefore, efficient heuristic algorithms have been proposed to find an approximate solution. Hence, a variety of heuristic procedures such as dispatching rules, local search and meta-heuristics procedures are used to solve such problems and to generate approximate solutions close to the optimum with considerably less computational time.

This chapter presents a novel PSO and QPSO combined with chaotic numbers and mutation operator for solving flexible job shop scheduling problem. The application of chaotic sequences based on chaotic logistic mapping instead of random sequences in PSO and QPSO, which is a powerful strategy to diversify the initial population and improve the algorithm's performance by preventing the premature convergence to local minima of the algorithm. The mutation operator is used to improve the solution diversity, which to accelerate convergence rate strategy. Thus, the possibility of exploring a global minimum in problems with many local optima is increased. The search will continue until a termination criterion is satisfied. The proposed approach uses PSO and QPSO to assign the operations of each job on available capable machines and sequence the operations on each machine. The objective considered in this chapter is to minimize makespan.

4.2 Problem representation of flexible job shop scheduling

In this chapter, a real number encoding system is proposed. The integer part is used to assign the operations of each job to the machine and fractional part is used to sequence of the operations on each machine. The position of the each particle is represented by a real number. The value of integer part allocate as a priority level for

each operation which is used to select the machine for the operation. First sequencing of available machines for an operation according to the increasing order of processing time is carried out. If tie occurs, the machine having lower number is given the priority. Priority levels for all machines are generated for processing all the operations of each job [16]. As an instance, a problem is to execute three jobs on four machines. Table 4.1 represents data including jobs, operations, and processing times on different machines. Table 4.2 shows the order of priority or priority level i.e 1, 2, 3, 4 of machines corresponding to each operation.

Table 4.1 Example problem of FJSP

Job	Operations	Machine 1	Machine 2	Machine 3	Machine 4
Job 1	$O_{1,1}$	9	5	4	3
	$O_{1,2}$	7	8	9	5
	$O_{1,3}$	5	8	8	3
Job 2	$O_{2,1}$	4	6	5	8
	$O_{2,2}$	5	4	6	2
Job 3	$O_{3,1}$	3	8	6	3
	$O_{3,2}$	5	5	2	2

Table 4.2 Priority order

Job	Operations	Priority 1	Priority 2	Priority 3	Priority 4
Job 1	O1,1	M4	M3	M2	M1
	O1,2	M4	M1	M2	M3
	O1,3	M4	M1	M2	M3
Job 2	O2,1	M1	M3	M2	M4
	O2,2	M4	M2	M1	M3
Job 3	O3,1	M1	M4	M3	M2
	O3,2	M3	M4	M1	M2

Table 4.3 represents the stochastic particle position representation. Initial particle's positions in the swarm are generated using logistic map function. The maximum position (X_{max}) of the particle is taken as the maximum value of priority level (mpl) i.e. the number of machines available. The position of the particle must be a positive integer as each particle position value represents priority level for each operation. Hence, it lies in the range [1, mpl]. For example, the 1st position is 2.25 and the integer value is 2. Therefore, operation $O_{1,1}$ is assigned to machine 3 as per the priority order in Table 4. 2. The process order of operations to be scheduled on the same machine depends on the value of fractional parts. The operations are sequenced according to the ascending order of the fractional part which is processed by the same machine. For instance, operations $O_{1,2}$ and $O_{3,2}$ are assigned to machine 2. The sequence of operations to be

scheduled on machine 2 is operation ($O_{3,2}$) followed by operation($O_{1,2}$) because the fractional part of the particle position for $O_{3,2}$ is greater than fractional part of the particle position for $O_{1,2}$. If the value of fractional parts is equal then the operation processing sequence is randomly chosen.

Table 4.3 Stochastic particle position representation

Operation	$O_{1,1}$	$O_{1,2}$	$O_{1,3}$	$O_{2,1}$	$O_{2,2}$	$O_{3,1}$	$O_{3,2}$
Particle positions	2.252	3.643	1.121	2.442	3.143	2.052	4.824
priority level							
Processing machine	M3	M2	M4	M3	M1	M4	M2

4.3 Proposed particle swarm optimization algorithm for FJSP

The proposed methodology introduces an improved algorithm by combining PSO with chaotic numbers and mutation operator is embedded to escape from local optima and to improve the solution diversity. Various steps involved in the proposed PSO algorithm to solve flexible job shop are listed out below.

Step 1. Initialize the parameters such as population size, maximum iteration, decrement factor, inertia weight, social and cognitive parameters.

Step 2. Input number of jobs, number of stages, and number of machines at each stage, and processing times.

Step 3. Generate the initial position values of the particle: $x_{ij}^{t} = x_{min} + (x_{max} - x_{min}) \times N(0,1)$ where x_{min} = 1.0, x_{max} = mpl and N (0,1) is a chaotic number between 0 and 1. Generate initial velocities of the particle $v_{ij}^{t} = v_{min} + (v_{max} - v_{min}) \times N(0,1)$ where v_{min} = -4.0, v_{max} = 4.0 and N (0,1) is a chaotic number between 0 and 1.

Step 4. Get the schedule using encoding scheme as mentioned in Section 4.2.

Step 5. Evaluate each particle's fitness (makespan).

Step 6. Find out the personal best (p_{best}) and global best (g_{best}).

Step 7. If (t< (t_{max}× PMUT), then perform mutation on x_{ij}^{t}.
(t_{max} is the maximum number of iterations and PMUT is the probability of mutation)

Step 8. Update velocity, position and inertia weight by using equations (3.7), (3.8) and (3.9). All the random number used in 3.7 is replaced by chaotic number.

Step 9. Terminate if maximum number of iterations is reached and store the g_{best} value. Otherwise, go to Step 3.

Step 10. End

4.4 Proposed quantum-behaved particle swarm optimization algorithm for FJSP

Various steps involved in the proposed QPSO algorithm to solve flexible job shop are listed out below.

Step 1. Initialization of swarm positions: Generate the initial position values of the particle: $x_{ij}^{t} = x_{min} + (x_{max} - x_{min}) \times N(0,1)$ where x_{min} = 1.0, x_{max} = mpl and N(0,1) is a chaotic number between 0 and 1.

Step 2. Input number of jobs, number of operation of each job, number of machines, and processing times.

Step 3. Generate the k , u, θ with chaotic number between 0 and 1.

Step 4. Get the schedule using encoding scheme as mentioned in Section 4.2.

Step 5. Evaluate each particle's fitness (makespan).

Step 6. Comparison to p_{best} (personal best): Compare each particle's fitness with the particle's p_{best}.

Step 7. Comparison to g_{best} (global best): Compare the fitness with the population's overall previous best.

Step 8.. If (t< (t_{max} × PMUT), then perform mutation on x_{ij}^{t}.
(t_{max} is the maximum number of iterations and PMUT is the probability of mutation)

Step 9. Then calculate the mean value of the best position (M_{best}) using equations 3.13 and generate the Contraction-Expansion factor using equation 3.14.

Step 10. Update the positions of all particles according to equations 3.10 and 3.11.

Step 11. Repeat the cycle: loop to Step 4 till the stop criterion is met or terminate if maximum number of iterations is reached and store the g_{best} value.

Step 12. END

4.5 Results and discussions

The computational study aims to analyze the performance of proposed PSO and QPSO to minimize the makespan for the flexible job shop scheduling problems. The algorithms were implemented in Matlab 7 on a Pentium IV running at 2 GHz on the Windows XP operating system. The proposed algorithms are tested on three sets of problem instances from Kacem et al. [117], Brandimarte [112] and Dauzere-peres [115] (DP data). Kacem et al. [117]'s data set contains three instances ranging from 8×8 to 15×10 whose scale (n × m, n: number of jobs, m: number of machines). Brandimarte [112]'s (BR) data set contains a set of 10 problems. The number of jobs ranges from 10 to 20, the number of machine ranges from 4 to 15 and the number of operations for each job ranges from 5 to 15. These two data sets are the most commonly adopted benchmark instances in the literature on FJSP. The DP data set is a set of 18 problems. The number of jobs ranges from 10 to 20, the number of machine ranges from 5 to 10 and the number of operations for each job ranges from 15 to 25.

The first data set considered for analysis belongs to Kacem data [117]. Table 4.4 compares the results of the proposed PSO and QPSO with approach by localization and classical GA (AL + CGA) [117], PSO + SA [152], Multistage-base GA (MGA) [118], [parallel variable neighborhood search algorithm (PVNS) [125] and hybrid genetic algorithm (hGA)[122]. The first column characterizes the size of the problem, in which 'n' stands for the number of jobs and m symbolize the number of given machines in the problem. The second column up to the sixth one represent the best makespan resulted from AL + CGA, PSO + SA, MGA, PVNS and hGA, respectively. The seventh and eighth column signifies the best makespan obtained from five runs of PSO and QPO algorithm. Gantt chart of the obtained solution for problem 10×10, 15x10 is illustrated in Figure 4.1 and 4.2. Table 4.4 illustrate that the results obtained by the proposed QPSO algorithm is superior than or equal to the cited algorithm from the literature. The computational results validate this algorithm's effectiveness.

Table 4.4 Makespan of Kacem instances

Problem size n x m	Proposed PSO	Proposed QPSO	AL + CGA [117]	PSO + SA [152]	MGA [118]	PVNS [125]	hGA [122]
8 x 8	14	14	15	15	15	14	14
10 x 10	7	7	7	7	7	7	7
15 x 10	11	11	24	12	NA	12	11

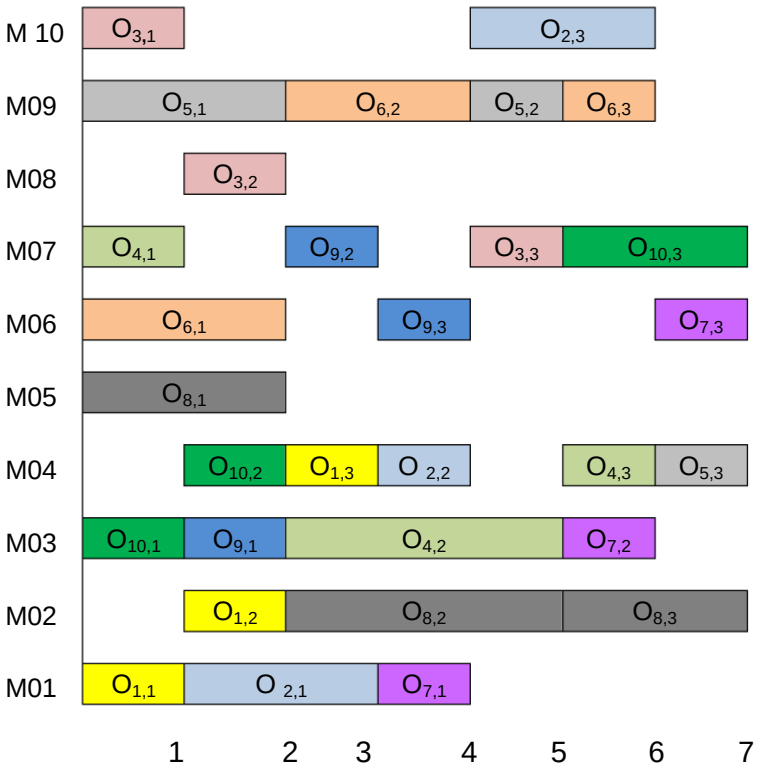


Figure 4.1 Gantt chart obtained by QPSO (Problem 10 x 10 from Kacem's instance)

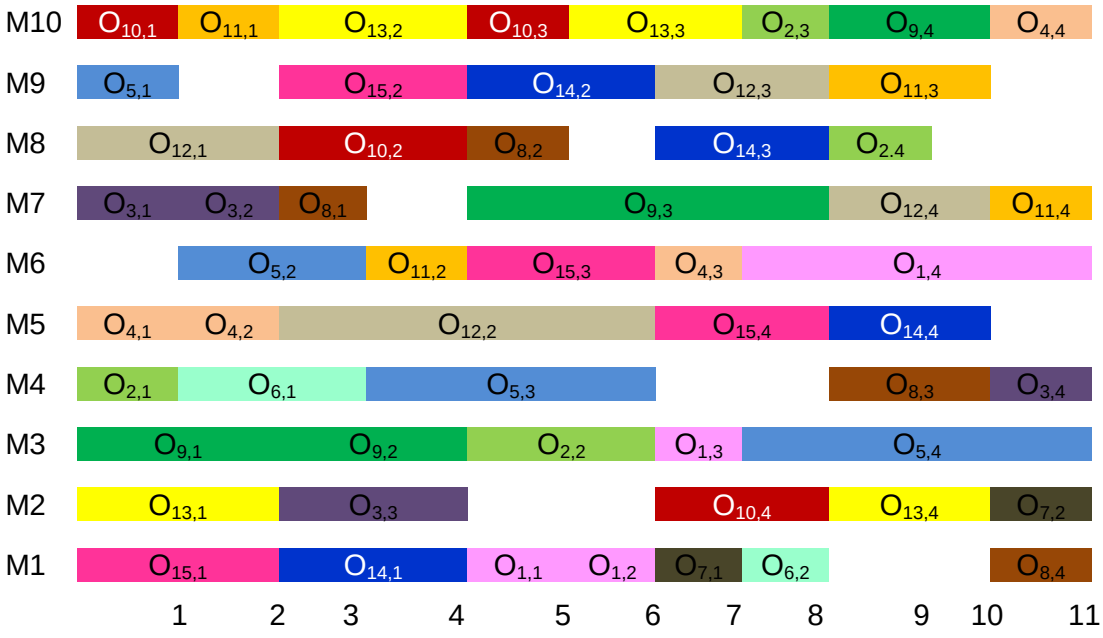


Figure 4.2 Gantt chart obtained by QPSO (Problem 15 x 10 from Kacem's instance)

In Table 4.5, comparison of makespan obtained by proposed PSO and QPSO to the results of the GA [121], PVNS[125] , integrated genetic algorithm [200] and knowledge based ant colony optimization (KBACO) [123] on ten FJSP instances from Brandimarte data set is made. The first and second columns symbolize the name and size of the problem respectively. The third column indicates to the value of lower bound (LB) .The fourth and fifth column refers to the best makespan result from proposed PSO and QPSO algorithm. The sixth column up to the ninth one represent the best makespan resulted from GA, PVSN, integrated genetic algorithm and KB ACO respectively. Relative percentage deviation criterion is used to compare the results of the proposed PSO and QPSO with those of the above four mentioned algorithms.

The results are represented in terms of percentage deviation (%PD) of the solution from the lower bound (LB) [Equation 3.18].

Table 4.5 Results of the BR data instances

Problem	n x m	LB	Proposed PSO		Proposed QPSO		GA [121]		PVNS [125]		Integrated GA [200]		KB ACO [123]	
			C_{max}	%PD	C_{max}	% PD	C_{max}	% PD	C_{max}	% PD	C_{max}	% PD	C_{max}	% PD
Mk 01	10 x 6	36	39	8.33	37	2.78	40	11.11	40	11.11	40	11.11	39	8.33
Mk 02	10 x 6	24	26	8.33	26	8.33	26	8.33	26	8.33	27	12.5	29	20.83
Mk 03	15 x 8	204	204	0.00	204	0.0	204	0.00	204	0.00	204	0.00	204	0.00
Mk 04	15 x 8	48	60	25.00	60	25	60	25.00	60	25.00	60	25	65	35.42
Mk 05	15 x 4	168	173	2.98	173	2.98	173	2.98	173	2.98	173	2.97	173	2.98
Mk 06	10x15	33	63	90.91	64	93.94	63	90.91	60	81.82	62	87.88	67	103.1
Mk 07	20 x 5	133	139	4.51	139	4.51	139	4.51	141	6.02	139	4.51	144	8.27
Mk 08	20x10	523	523	0.00	523	0.00	523	0.00	523	0.00	523	0.00	523	0.00
Mk 09	20x10	299	307	2.68	307	2.68	311	4.01	307	2.68	309	3.34	311	4.01
Mk 10	20x15	165	203	23.03	202	22.42	212	28.48	208	26.06	206	24.85	229	38.79

It is observed from the Table 4.5 that most of the makespan values by the proposed PSO and QPSO algorithm have smaller PD over GA, PVNS, Integrated GA and KBACO. Again it is noticeable that proposed QPSO performs better than the proposed PSO algorithm.

The APD [Equation 3.19] for proposed PSO, GA, PVNS, Integrated GA and KBACO are found to be 16.58, 17.53, 16.39, 17.22 and 22.16 respectively. APD for QPSO is obtained as 16.26 which happen to be superior as compared to other algorithms. The performance of KBACO is worst among all the algorithms.

The improvement rate for APD using QPSO is defined as follows:

$$\text{Improvement rate } (\%) = \frac{(\text{APD}_{\text{PSO,GA,PVSN,IGA,KBACO}} - \text{APD}_{\text{QPSO}})}{\text{APD}_{\text{PSO,GA,AIS,B\&B}}} \quad (4.13)$$

The improvement rate in terms of APD is 1.889 % with respect to proposed PSO, 7.534% with respect to GA, 0.825% with respect to parallel variable neighborhood search (PVNS) algorithm, 5.5366 % with respect to integrated genetic algorithm (IGA) and 26.63% with respect to knowledge based ant colony optimization (KBACO) for the Brandimarte data set.

Table 4.6 illustrates the comparisons the computational results of Dauzere-peres data set. The first and second columns symbolize the name and size of the problem respectively. The third column indicates to the value of lower bound (LB) and upper bound (UB). The fourth and fifth column refers to the best makespan results from proposed PSO and QPSO algorithm. The sixthth to seventh column represents the best makespan results from TS [116], hybrid genetic algorithm (hGA) [122].Table 4.6 indicates that TS produces better result than proposed QPSO in one case, hGA does not give better solution than proposed QPSO . It is identified from the Table 4.6 that most of the makespan values by the proposed PSO and QPSO algorithm have smaller PD over TS and hGA.

The APD for the proposed PSO proposed QPSO, TS and hGA are found to be 1.563, 1.44784, 2.01 and 2.124 respectively. For the Dauzere-peres data set, the proposed QPSO has an improvement rate (in APD) of 7.369 % with respect to proposed PSO, 27.936% with respect to TS, and 31.836 % with respect to hGA.

The convergence curve is drawn for 11a of Dauzere-peres data set in Figure 4.3. Figure 4.3 illustrates the comparison of the random and chaotic numbers. It is conceivable to note that QPSO algorithm with chaotic numbers improves the makespan, and that the best makespan, equal to 2050, is reached after 41 iterations. It can be observed from Figure 4.3 that the solution converges towards the best value faster when the chaotic numbers are used. This is because of higher degree of disorderness of the chaotic numbers which facilitates high diversity in the particles and helps the algorithm to converge rapidly towards the solution.

Table 4.6 Results of the Dauzere-peres instances

Problem	n x m	LB,UB	Proposed PSO		Proposed QPSO		TS [116]		hGA [122]	
			C_{max}	%PD	C_{max}	%PD	C_{max}	%PD	C_{max}	%PD
1a	10x5	2,505	2505	0	2505	0	2518	0.519	2518	0.519
2a	10x5	2,228	2230	0.090	2230	0.090	2231	0.135	2231	0.135
3a	10x5	2,228	2229	0.045	2229	0.045	2229	0.045	2229	0.045
4a	10x5	2,503	2506	0.120	2503	0.000	2503	0.000	2515	0.479
5a	10x5	2,189	2210	0.959	2207	0.822	2216	1.233	2217	1.279
6a	10x5	2,162	2174	0.555	2170	0.370	2203	1.896	2196	1.573
7a	15x8	2,187	2271	3.841	2264	3.521	2283	4.390	2307	5.487
8a	15x8	2,061	2073	0.582	2073	0.582	2069	0.388	2073	0.582
9a	15x8	2,061	2066	0.243	2066	0.243	2066	0.243	2066	0.243
10a	15x8	2,178	2207	1.331	2205	1.240	2291	5.188	2315	6.290
11a	15x8	2,017	2064	2.330	2050	1.636	2063	2.281	2071	2.677
12a	15x8	1,969	2021	2.641	2019	2.539	2034	3.301	2030	3.098
13a	20x10	2,161	2253	4.257	2253	4.257	2260	4.581	2257	4.442
14a	20x10	2,161	2167	0.278	2167	0.278	2167	0.278	2167	0.278
15a	20x10	2,161	2165	0.185	2165	0.185	2167	0.278	2165	0.185
16a	20x10	2,148	2258	5.121	2252	4.842	2255	4.981	2256	5.028
17a	20x10	2,088	2136	2.299	2134	2.203	2141	2.538	2140	2.490
18a	20x10	2,057	2124	3.257	2123	3.209	2137	3.889	2127	3.403

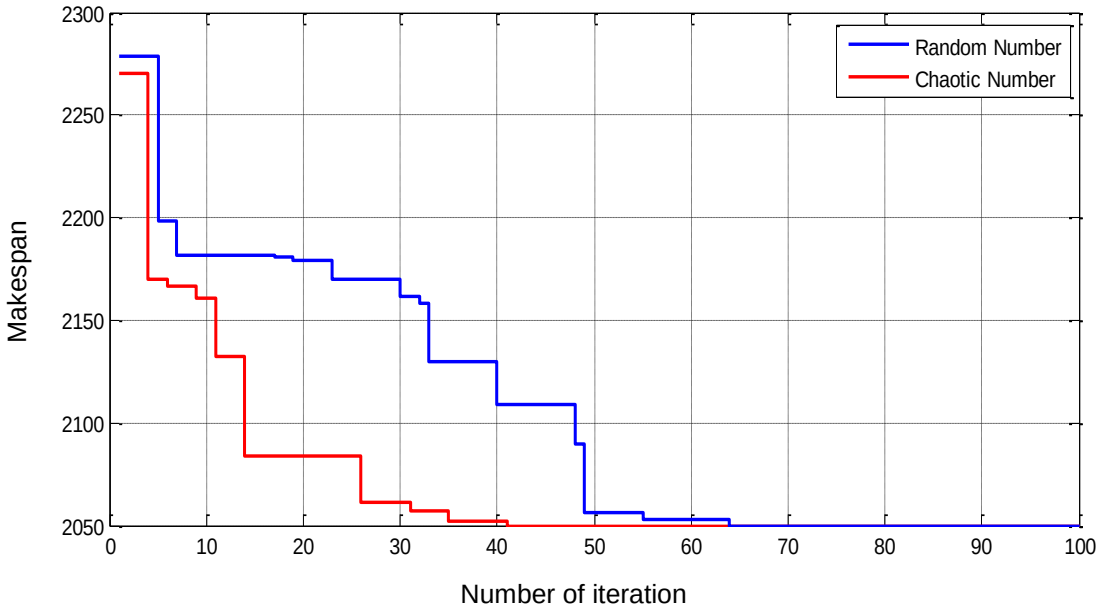


Figure 4.3 The convergence curve of the QPSO algorithm

(Problem 11a from DP data set)

The comparison between the proposed QPSO algorithm and the results obtained by other algorithms is made in Tables 4.5 and Table 4.6. The proposed algorithm has outperformed almost all the benchmark instances. Therefore, it is concluded from the computational results that the proposed QPSO provides better performance than those testified by other algorithms.

4.6 Conclusions

In this charter, flexible job shop scheduling problem which is NP-hard was considered and an efficient PSO and QPSO to find near-optimal schedules has been used. The proposed QPSO approach is a found to be a good problem solving technique for scheduling problem. The algorithm is applied three sets of problem instances from Kacem et al. [117], Brandimarte [112] and Dauzere-peres [115]. It has been validated that the results obtained by proposed QPSO has an improvement rate of 7.369 % with respect to proposed PSO, 27.936% with respect to TS, 31.836 % with respect to hybrid genetic algorithm (hGA) for the DP data set in terms of APD. The improvement rate in terms of APD is 1.889 % with respect to proposed PSO, 7.534% with respect to GA, 0.825% with respect to parallel variable neighborhood search (PVNS) algorithm, 5.5366 % with respect to integrated genetic algorithm (IGA) and 26.63% with respect to knowledge based ant colony optimization (KBACO) for the Brandimarte data set. For the Dauzere-peres data set, the proposed QPSO has an improvement rate (in APD) of 7.369 % with respect to proposed PSO, 27.936% with respect to TS, and 31.836 % with respect to hGA. The results indicate that QPSO produces either better solutions or same as compared to best known solutions in the literature. It has been demonstrated that QPSO outperforms other well-known algorithms at least for benchmark instances considered in the study. The study can be extended in future to consider more number of problems, proposed QPSO with local search techniques to reduce computational burden, and consider machine set up time. The next chapter addresses to produce a robust schedule for a flexible flow shop and job shop scheduling problem with random machine breakdown. Multi-objective framework based on particle swarm optimization (PSO) and quantum particle swarm optimization (QPSO) is proposed to generate the robust schedule by minimizing the makespan and the robust measures simultaneously.

CHAPTER 5

FLEXIBLE FLOW SHOP AND JOB SHOP SCHEDULING WITH MACHINE BREAKDOWN

5.1 Introduction

The research area of production scheduling has received extensive attention from both the academic and the industries over the last decade. However, most of the shop scheduling research assumes that scheduling parameters are known and deterministic. The real world manufacturing environments is quite dynamic in nature and schedules are subjected to various disruptions due to a wide range of stochastic uncertainties [132,133]. For example, resource shortages and machine breakdowns can delay a schedule's completion time. It is essentially important for the manufacturing firms to improve the performance of production scheduling systems that can address internal uncertainties such as machine breakdown, tool failure, and change in processing times. The schedules must meet the deadline committed to customers because failure to do so may result in a significant loss of goodwill. Also scheduling of activities should be efficient enough to use the available resources in an effective manner. To address these issues, shop scheduling has motivated the researchers in both academia and industry and a variety of scheduling problems have been acknowledged from real-world manufacturing environments [140,144]. FFSP and FJSP problems become much more difficult to solve if uncertainties are considered. The uncertainties in manufacturing context can be attributed due to resource and job [201,202]. The uncertainties due to resource may be listed as machine breakdown, unavailability or failure of tools, operator illness, loading limits, delay in the arrival or shortage of materials, defective material etc. Uncertainties due to job may be listed as rush jobs, due date changes, job cancellation, changes in job priority, early or late arrival of jobs, changes in job processing time etc.

Among all the uncertain events, machine breakdown is one of the significant disruptions in shop scheduling problems. In this study, the flexible flow shop scheduling problem (FFSP) and flexible job shop scheduling problem (FJSP) is dealt considering stochastic machine breakdown as the uncertainty. In addition to normal performance measures such as makespan, flow time, and tardiness, two more measures known as robustness and stability are considered in the uncertain environments [203,204,205]. Uncertainty has two kinds of major negative impacts on initial schedules. First, it degrades schedule performance measured in terms of deviation of makespan between the realized schedule (with disruptions) and the predictive schedule (without disruptions)[]. This kind of effect is known as robustness. If the schedule performance does not deteriorate in disruptions situation then the schedule is termed as a robust one. Secondly, unexpected disruptions cause inconsistency. This kind of effect is known as

stability. The schedule which does not deviate the completion time of the unaffected operations from the original schedule in a disrupted situation then it is called stable [142].

In this chapter, robustness of flexible flow shop and job shop scheduling under the stochastic machine breakdown is evaluated. The proposed PSO and QPSO which is described in section 3.7 and 3.8 is used to obtain a schedule that reduces the effect of machine breakdowns in the overall performance (makespan). Three popular robustness measures are tested in a multi objective framework. An experimental study and analysis of variance (ANOVA) is conducted to study the effect of different proposed robustness measures on the performance under uncertainty situation so that decision makers can determine trade-offs between makespan and robustness for their schedules Exhaustive experimental study is conducted to study the effect of different proposed robustness measures on the generated schedules using benchmark problems.

5.2 Machine breakdown formulation

The performance of a schedule is usually challenged with disturbances and unpredicted events. In real world manufacturing environment, a shop floor may be affected by various uncertain factors. Flexible flow shop environment with random machine breakdowns during schedule execution is considered here.

5.2.1 Machine breakdown formulation for FFSP

A shop floor may have number of machine breakdowns during a scheduling. Machine breakdown in flexible flow shop is commonly described in terms of (a) identify in which stage which machine to break down, (b) the repair time, and (c) the time of breakdown [146].

As the time of machine failure is not predictable, a probability distribution can identify on the basis of historical data. Assume the probability of the stage in which the machine break down occurs is subjected to the ratio between the individual stage busy times with the total busy time of all stages.

$$S_m = \frac{BTS_m}{BTS_{Total}} \tag{5.1}$$

where S_m: The probability of stage m to fail, BTS_K: the busy time of stage m, and BTS_{Total} : the total busy time of all stages.

Assume the probability of machine break down is subjected to the ratio between the individual machine busy times with the total busy time of all machines.

$$P_K = \frac{BT_K}{BT_{Total}} \tag{5.2}$$

where P_K: The probability of machine k to fail in stage S_m, BT_K : the busy time of machine k in stage S_m, and BT_{Total} : the total busy time of all machines in stage S_m.

The machine breakdown level is categorized by two parameters: (i) two levels of machine breakdown duration (repair time) i.e. low level and high level and (ii) the interval of machine breakdown occurrence time i.e. early and late. The machine breakdown time and the breakdown duration (repair time) are generated by uniform distributions.

$$RT_K = [\beta_1 BT_K , \beta_2 BT_K] \quad (5.3)$$

RTK: Machine breakdown duration (repair time), where β_1 and β_2 are disruption level coefficient between [0, 1]

$$BT_K = [\propto_1 BT_K , \propto_2 BT_K] \quad (5.4)$$

BTK: Machine breakdown occurrence time, where $\propto_1$ and $\propto_2$ are coefficient between [0, 1]

5.2.2 Machine breakdown formulation for FJSP

As the breakdowns occur randomly, we have proposed robust schedule a simulator to simulate the situations that allow the machines to disrupt the operations due to the random breakdowns in flexible job shop scheduling problem. Machine breakdown in flexible job shop is commonly described in terms of (a) identify which machine to break down, (b) the repair time, and (c) the time of breakdown.

$$P_K = \frac{BT_K}{BT_{Total}} \quad (5.5)$$

where P_K: The probability of machine k to fail, BT_K : the busy time of machine k, and BT_{Total} : the total busy time of all machines.

The machine breakdown time and the breakdown duration (repair time) are generated by uniform distributions as per equation 5.3 and 5.4.

In this work, we have generated two scenarios for machine breakdown i.e. low disruption level and high disruption level. The value of β between 0.1 and 0.15 create the disruption level to a relatively low level and these values between 0.35 and 0.4 raises the disruption level to be from 35% to 40% of the machine's busy time. Similarly, the values of α ensure that the breakdown occurrence time. If α is between 0 and 0.5, the breakdown occurs during the first half of the scheduling. While the values of α is between 0.5 and 1, the breakdown occurs during the second half of the scheduling. When sudden breakdown occurs during the process, it is required to re-optimize the affected operations from the time of machine breakdown. If any operation is incomplete

due to a machine breakdown, the incomplete operation is resumed after the repair is complete.

5.3 Multi-objective optimization for robust schedule

The scheduling problem becomes a multi-objective nature when makespan and robustness are simultaneously considered. In this article, makespan is considered as the primary objective for FFSP and FJSP with random breakdowns and the robustness measure as the secondary objective. These objectives are linearly combined (scalarization method) to form a bi-objective function.

$$Z = w_1 \times (F_1) + w_2 \times (F_2) \quad (5.6)$$

where, w_1, w_2 is the weight co-efficient and $F_1$is the primary objective i.e. makespan and the F_2 is the second objective i.e. the robust measure.

Three types of robustness measures (RMs) are taken from the [142,146]. A comparison of performance of the robustness measure is performed within a multi objective optimization framework.

$$\text{RM } 1 = \frac{\frac{1}{N}\sum_{i=1}^{N} E[M_R(s)] - M_P(s)}{M_P(s)} \quad (5.7)$$

where N is the sample size and $E[M_R(s)]$ is the expected makespan of a realized schedule 'S' i.e. with a specific uncertainty condition and $M_P(s)$ is the makespan of a predictive schedule i.e. deterministic makespan of schedule S.

$$\text{RM2} = E[\sum_{i=1}^{n} \max(0, C_{iR} - d_i] \quad (5.8)$$

RM2 is the expected realized total tardiness. C_{iR} is the expected completion time of job i in the realized schedule. d_i is the is the due date of job i and n is total number of job.

$$\text{RM3} = E[\sum_{i=1}^{n} C_{iR}\ ] \quad (5.9)$$

RM 3 is the expected realized total flow time.

5.4 The proposed PSO algorithm and approach in machine breakdown

The proposed PSO algorithm is used to obtain a schedule in an uncertainty condition. The procedure for implementing the PSO is given by the following steps

Step 1. Initialize the parameters such as population size, maximum iteration, decrement factor, inertia weight, social and cognitive parameters.

Step 2. Input number of jobs, number of stages, and number of machines at each stage, and processing times.

Step 3. Generate the initial position values of the particle: $x_{ij}^{t} = x_{min} + (x_{max} - x_{min}) \times N(0,1)$ where x_{min} = 1.0, x_{max} = 1+M_j and N(0,1) is a chaotic number between 0 and 1. Generate initial velocities of the particle $v_{ij}^{t} = v_{min} + (v_{max} - v_{min}) \times$

$N(0,1)$ where v_{min} = -4.0, v_{max} = 4.0 and N (0,1) is a chaotic number between 0 and 1.where M_j is the number of machines in j^{th} stage

Step 4. Get the schedule using encoding scheme.

4.1. Decode the particles's position.

4.2. For all the operation of the jobs

(a) Assign operations to machine according to the particle position as described in section 3.4 for the flexible flow shop scheduling problem and section 4.2 for the flexible job shop.

(b) Run the breakdown disruption algorithm according section 5.2.1 for the flexible flow shop scheduling problem and section 5.2.2 for the flexible job shop scheduling problem.

Step 5. Evaluate each particle's fitness (makespan).

Step 6. Find out the personal best (p_{best}) and global best (g_{best}).

Step 7. If (t< (t_{max} * PMUT), then perform mutation on x_{ij}^{t}.

(t_{max} is the maximum number of iterations and PMUT is the probability of mutation)

Step 8. Update velocity, position and inertia weight by using equations 3.7, 3.8 and 3.9. All the random number used in equation 3.7 is replaced by chaotic number.

Step 9. Terminate if maximum number of iterations is reached and store the g_{best} value. Otherwise, go to Step 3.

Step 10. End

5.5 The proposed QPSO algorithm and approach in machine breakdown

The proposed QPSO algorithm is used to obtain a schedule in an uncertainty condition. The procedure for implementing the QPSO is given by the following.

Step 1. Initialization of swarm positions: Generate the initial position values of the particle.

Step 2. Input number of jobs, number of operation of each job, number of machines, and processing times.

Step 3. Generate the k , u, θ with chaotic number between 0 and 1.

Step 4. Get the schedule using encoding scheme.

4.1. Decode the particles's position.

4.2. For all the operation of the jobs

(a) Assign operations to machine according to the particle position as described in section 3.4 for the flexible flow shop scheduling problem and section 4.2 for the flexible job shop.

(b) Run the breakdown disruption algorithm according section 5.2.1 for the flexible flow shop scheduling problem and section 5.2.2 for the flexible job shop scheduling problem.

Step 5. Evaluate each particle's fitness (makespan).

Step 6. Comparison to p_{best} (personal best): Compare each particle's fitness with the particle's $p_{best.}$

Step 7. Comparison to g_{best} (global best): Compare the fitness with the population's overall previous best.

Step 8. If (t< (t_{max} * PMUT), then perform mutation on x_{ij}^{t}.

Step 9. Then calculate the mean value of the best position (M_{best}) using Equation. 3.13 and generate the Contraction-Expansion factor using Equation. 3.14.

Step 10. Update the positions of all particles according to Equations. 3.10 and 3.11.

Step 11. Repeat the cycle: loop to Step 4 till the stop criterion is met or terminate if maximum number of iterations is reached and store the g_{best} value.

Step 12. END

5.6 Results and discussions

The computational study aims to analyze the performance of proposed PSO and QPSO to minimize the makespan for the FFSP, FJSP problems and the analysis of variance (ANOVA) test is conducted to analyze robustness measures in terms of solution quality in a random machine breakdown environment. Multi objectives are linearly combined and tested by the proposed PSO and QPSO algorithm.

5.6.1 Result analysis of FFSP in an uncertainty condition (Machine breakdown)

To evaluate the performance of the schedule, a comparative study is made by using different robust measures with different disturbance scenario for the flexible flow shop scheduling problems. Usually, a scheduler requires a schedule that has a minimum makespan when released to the shop floor and if a disruption occurs, same schedule has the minimum realized makespan after rescheduling is implemented.

In this section, performance of robustness measures within a multi objective framework is analyzed by implementing the proposed PSO and QPSO algorithms with different robustness measures. An experimental study is conducted to investigate the solution quality of algorithms with robustness measure. The four factors such as break

down (BD), robustness measure (RM), algorithm (ALGORITHM) and problem instances (TEST CASE) factors and their levels are illustrated in Table 5.1. A full factorial experiment design having thirty six experimental runs (2×3×2×3) has been selected to gather sufficient information on model behavior with less number of experiments. The test cases are taken from [196]. Each test case is subjected to the two different levels of the breakdown (high level and low level) and each time is solved using one of the three robustness measures. Schedule is subjected to four hundred random machine breakdowns in each test in order to draw more accurate responses. Under each treatment, multi objective performance measure (Z) is computed. Analysis of variance (ANOVA) is performed on the response, Z, using the commercial statistical software Minitab 14.

Table 5.1 Factors and their levels for FFSP

Factors	Index of levels	Levels
BD	1	Low level breakdown
	2	High level breakdown
RM	1	RM1
	2	RM2
	3	RM3
ALGORITHM	1	PSO
	2	QPSO
TEST CASE	1	j10c5a3
	2	j10c10a3
	3	j15c5a3

Table 5.2 ANOVA table for FFSP

Source	DF	Seq SS	Adj SS	Adj MS	F	P
BD	1	0.0000381	0.0000508	0.0000508	0.6	0
ALGORITHM	1	0.0009509	0.0012478	0.0012478	14.85	0.438
RM	2	0.0442662	0.0398747	0.0199373	237.21	0
TEST CASE	2	0.0802305	0.0828654	0.0414327	492.95	0
BD*ALGORITHM	1	0.0003729	0.0000866	0.0000866	1.03	0.312
BD*RM	2	0.0044058	0.0039122	0.0019561	23.27	0
BD*TEST CASE	2	0.0014688	0.001214	0.000607	7.22	0.001
ALGORITHM*RM	2	0.0012129	0.0017625	0.0008812	10.48	0
ALGORITHM*TEST CASE	2	0.0011523	0.0012076	0.0006038	7.18	0.001
RM*TEST CASE	4	0.0522343	0.0522343	0.0130586	155.36	0
Error	137	0.011515	0.011515	0.0000841		
Total	156	0.1978477				

The ANOVA shown in Table 5.2 is used for testing statistical significance of factors. It can be concluded that effect of the robustness measure, test case, break down type and the interaction of break down type and robustness measure are significant at significant level of 0.05. The Figures 5.1 and 5.2 illustrate that the breakdown type (BD) and robustness measure (RM) has a significant impact on the schedule in a multi-objective framework. As indicated in Figure 5.2, better schedule is obtained with the RM 3 and QPSO algorithm.

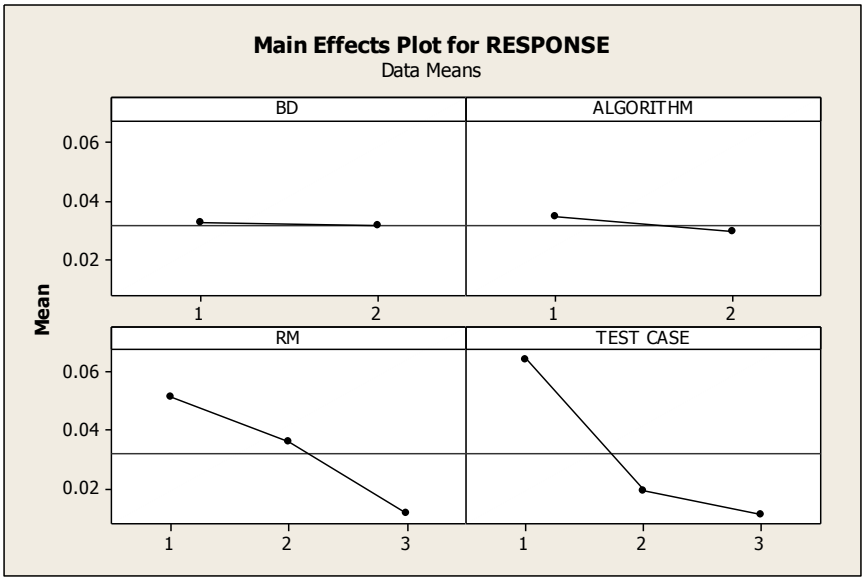


Figure 5.1 Main effects plot (data mean) for Response for FFSP

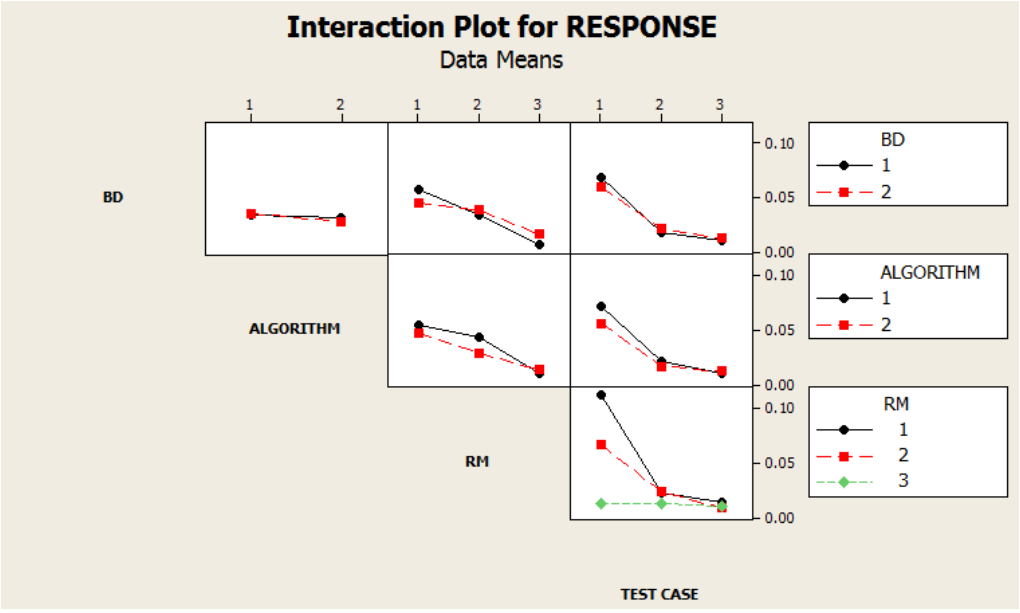


Figure 5.2. Interaction Plot (data means) for Response for FFSP

The proposed PSO and QPSO algorithm is tested in a multi objective framework according to equation 5.6 i.e. the primary objective as makespan and the second objective as the robust measure (RM3) on two different breakdown scenarios in the flexible flow shop scheduling, is analyzed on benchmark problem instances from Carlier and Neron [196].Table 5.3 compares the results of the proposed QPSO with proposed PSO. The first column symbolizes the name of the problem. The second and third column represents the makespan results from proposed PSO and proposed QPSO during a low disruption level (BD1), fifth and sixth column represents the makespan results during a high disruption level (BD2). The fourth and seventh column represents the improvement rate. It is conceivable to note that QPSO algorithm improves the makespan over the PSO algorithm during the machine breakdown scenarios.

$$\text{Improvement rate } (\%) = \frac{(\text{Makespan}_{\text{PSO}} - \text{Makespan}_{\text{QPSO}})}{\text{Makespan}_{\text{PSO}}} \quad (5.10)$$

Table 5.3 Comparison results for FFSP at two different machine breakdown scenarios

Problem	BD 1		Improvement rate (%)	BD 2		Improvement rate (%)
	Proposed PSO	Proposed QPSO		Proposed PSO	Proposed QPSO	
j10c5a2	105	97	7.619	118	110	6.780
j10c5a3	132	126	4.545	154	148	3.896
j10c5a4	137	130	5.109	164	157	4.268
j10c5a5	136	135	0.735	160	157	1.875
j10c5a6	129	121	6.202	151	144	4.636
j10c5b1	153	141	7.843	171	166	2.924
j10c5b2	123	114	7.317	140	134	4.286
j10c5b3	118	118	0.000	137	137	0.000
j10c5b4	132	132	0.000	162	159	1.852
j10c5b5	165	165	0.000	199	199	0.000
j10c5b6	126	124	1.587	145	145	0.000
j10c5c1	88	82	6.818	110	104	5.455
j10c5c2	97	93	4.124	115	110	4.348
j10c5c3	82	81	1.220	117	114	2.564
j10c5c4	89	85	4.494	103	98	4.854
j10c5c5	92	87	5.435	94	92	2.128
j10c5c6	74	72	2.703	83	79	4.819
j10c5d1	88	86	2.273	107	105	1.869
j10c5d2	77	76	1.299	89	89	0.000
j10c5d3	87	86	1.149	104	101	2.885
j10c5d4	78	75	3.846	101	100	0.990
j10c5d5	81	73	9.877	106	95	10.377
j10c5d6	99	96	3.030	107	106	0.935

j10c10a1	177	176	0.565	189	189	0.000
j10c10a2	175	170	2.857	195	192	1.538
j10c10a3	164	157	4.268	181	181	0.000
j10c10a4	181	180	0.552	197	197	0.000
j10c10a5	162	157	3.086	188	184	2.128
j10c10a6	182	180	1.099	199	197	1.005
j10c10b1	165	162	1.818	193	193	0.000
j10c10b2	184	184	0.000	202	200	0.990
j10c10b3	172	170	1.163	210	204	2.857
j10c10b4	171	171	0.000	198	196	1.010
j10c10b5	182	180	1.099	207	204	1.449
j10c10b6	179	179	0.000	205	204	0.488
j10c10c1	134	133	0.746	159	159	0.000
j10c10c2	146	143	2.055	192	187	2.604
j10c10c3	104	101	2.885	119	117	1.681
j10c10c4	118	117	0.847	129	126	2.326
j10c10c5	131	129	1.527	167	162	2.994
j10c10c6	121	120	0.826	149	148	0.671
j15c5a1	198	194	2.020	234	222	5.128
j15c5a2	185	181	2.162	218	215	1.376
j15c5a3	142	141	0.704	169	168	0.592
j15c5a4	172	172	0.000	207	204	1.449
j15c5a5	178	178	0.000	213	213	0.000
j15c5a6	194	194	0.000	234	234	0.000
j15c5b1	201	195	2.985	220	217	1.364
j15c5b2	189	187	1.058	231	229	0.866
j15c5b3	177	173	2.260	207	207	0.000
j15c5b4	164	160	2.439	193	190	1.554
j15c5b5	197	197	0.000	225	220	2.222
j5c5b6	214	212	0.935	239	235	1.674
j15c5c1	89	88	1.124	101	98	2.970
j15c5c2	112	109	2.679	119	117	1.681
j15c5c3	96	96	0.000	118	115	2.542
j15c5c4	105	101	3.810	131	128	2.290
j15c5c5	95	95	0.000	113	111	1.770
j15c5c6	111	106	4.505	129	126	2.326
j15c5d1	211	210	0.474	244	240	1.639
j15c5d2	101	100	0.990	127	124	2.362
j15c5d3	99	95	4.040	118	114	3.390
j15c5d4	69	65	5.797	86	85	1.163
j15c5d5	80	75	6.25	101	94	6.931
j15c5d6	81	81	0	114	110	3.509
j15c10a1	248	244	1.613	266	261	1.880

j15c10a2	221	220	0.452	242	240	0.826
j15c10a3	213	213	0.000	234	234	0.000
j15c10a4	248	249	-0.403	267	266	0.375
j15c10a5	212	209	1.415	224	221	1.339
j15c10a6	228	228	0.000	238	237	0.420
j15c10b1	234	230	1.709	250	246	1.600
j15c10b2	252	252	0.000	304	301	0.987
j15c10b3	248	245	1.210	291	290	0.344
j15c10b4	234	233	0.427	267	265	0.749
j15c10b5	277	275	0.722	299	295	1.338
j15c10b6	245	241	1.633	278	278	0.000
Average Improvement Rate			2.151			1.97

On the Average Improvement Rate of QPSO with respect to PSO, an improvement of 2.151% in a low disruption level (BD1), 1.97 % in a high disruption level (BD2) has been achieved for the benchmark problems considered in the study.

5.6.2 Result analysis of FJSP in an uncertainty condition (Machine breakdown)

A comparative study is made to evaluate the performance of the schedule obtained using different robust measures with different disturbance scenario. In this section, performance of robustness measures within a multi objective framework is analyzed by implementing the proposed PSO and QPSO algorithms with different robustness measures for flexible job shop scheduling problem. An experimental study is conducted to investigate the solution quality of algorithms with robustness measure. The four factors such as break down (BD), robustness measure (RM), algorithm (ALGORITHM) and problem instances (TEST CASE) factors and their levels are illustrated in Table 5.4. A full factorial experiment design having thirty six experimental runs (2×3×2×3) has been selected to gather sufficient information on model behavior with less number of experiments. The test cases are taken from BR data set according to different problem size. Each test case is subjected to the two different levels of the breakdown (high level and low level) and each time is solved using one of the three robustness measures. Schedule is subjected to four hundred random machine breakdowns in each test in order to draw more accurate responses. Under each treatment, multi objective performance measure (Z) is computed. Analysis of variance (ANOVA) is performed on the response, Z, using the commercial statistical software Minitab 14.

Table 5.4 Factors and their levels for FJSP

Factors	Index of levels	Levels
BD	1	Low level breakdown
	2	High level breakdown
RM	1	RM1
	2	RM2
	3	RM3
ALGORITHM	1	PSO
	2	QPSO
TEST CASE	1	Mk 01
	2	Mk 03
	3	Mk 06

Table 5.5 ANOVA table for FJSP

Source	DF	Seq SS	Adj SS	Adj MS	F	P
BD	1	0.031809	0.031809	0.031809	29.41	0.000
ALGORITHM	1	0.000143	0.000143	0.000143	0.13	0.720
RM	2	0.069648	0.069648	0.034824	32.19	0.000
TEST CASE	2	0.142155	0.142155	0.071077	65.71	0.000
BD* ALGORITHM	1	0.000000	0.000000	0.000000	0.00	0.994
BD*RM	2	0.052590	0.052590	0.026295	24.31	0.000
ALGORITHM *RM	2	0.000018	0.000018	0.000009	0.01	0.992
Error	24	0.025961	0.025961	0.001082		
Total	35	0.322323				

The ANOVA shown in Table 5.5 is used for testing statistical significance of factors. It can be concluded that effect of the robustness measure, test case, break down type and the interaction of break down type and robustness measure are significant at significant level of 0.05. The Figures 5.3 and 5.4 illustrate that the breakdown type (BD) and robustness measure (RM) has a significant impact on the schedule in a multi-objective framework. As indicated in Figure 5.4, better schedule is obtained at BD1 i.e. low breakdown scenario with the RM 3 (the expected realized total flow time) and QPSO algorithm.

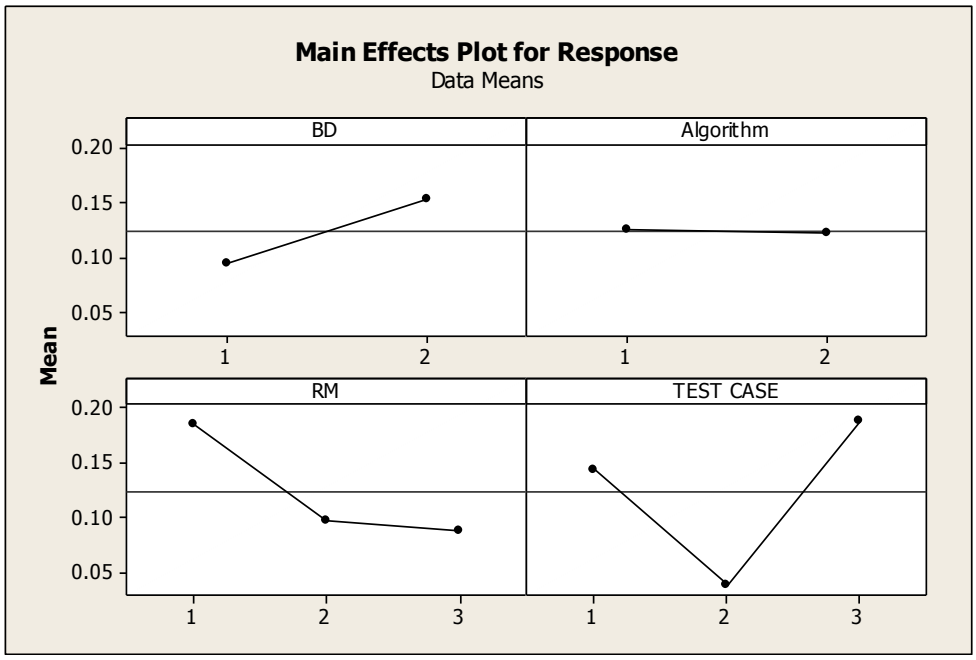


Figure 5.3 Main effects plot (data mean) for Response for FJSP

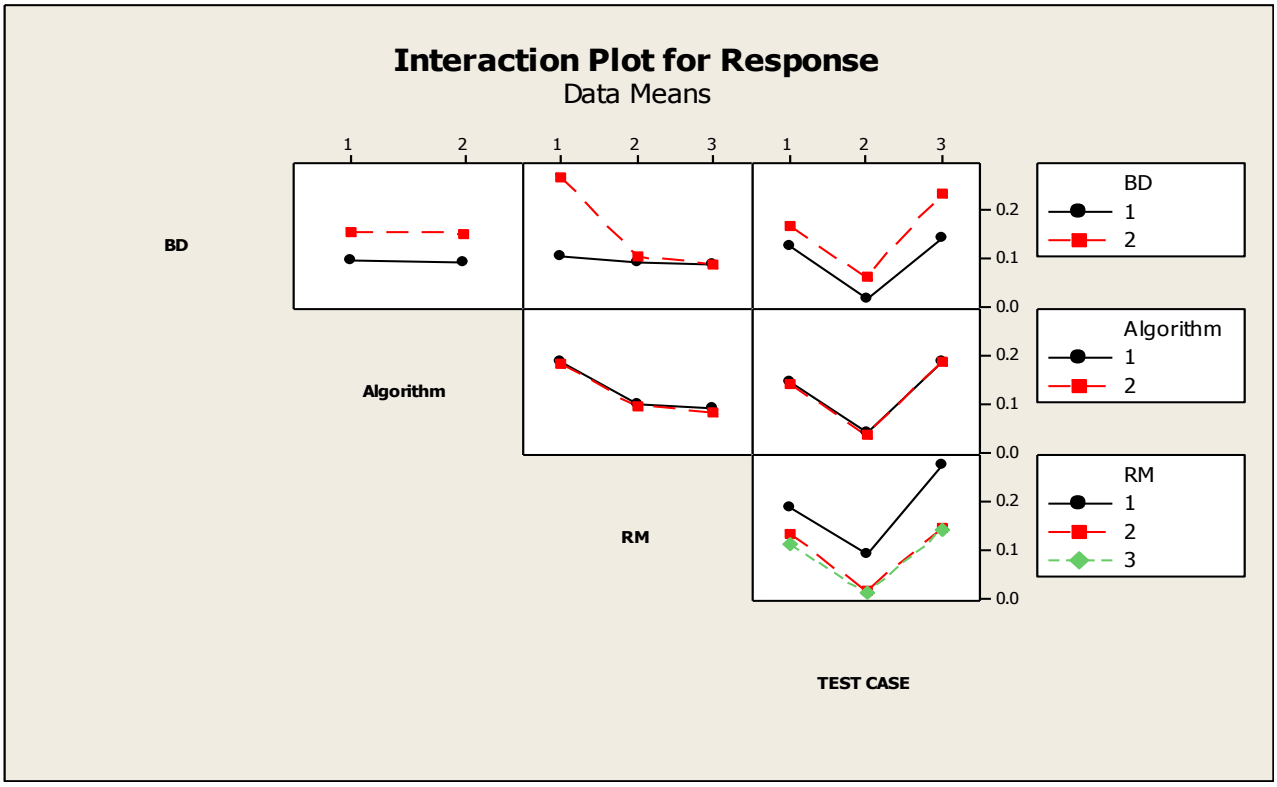


Figure 5.4 Interaction Plot (data means) for Response for FJSP

The proposed PSO and QPSO algorithm is tested in a multi objective framework according to equation 5.6 i.e. the primary objective as makespan and the second objective as the robust measure RM3 (the expected realized total flow time) on two different breakdown scenarios, is analyzed on three sets of problem instances from Kacem et al. [117], Brandimarte [112] and Dauzere-peres and paulli [115] (DP data). Table 5.6-5.8 compares the results of the proposed PSO and QPSO. The first and second columns symbolize the name and size of the problem respectively in which n stands for the number of jobs and m symbolize the number of given machines in the problem. The third and fourth column represents the makespan results from PSO and QPSO during a low disruption level (BD1) and sixth and seventh column represents the makespan results during a high disruption level (BD2). The fifth and eighth column represents the improvement rate. It is conceivable to note that QPSO algorithm improves the makespan over the PSO algorithm during the machine breakdown scenarios. The improvement rate for makespan in breakdown scenarios using QPSO over PSO is defined as follows:

Table 5.6 Results of the Kacem instances at two different machine breakdown scenarios

Problem	n × m	BD 1		Improv-ement rate (%)	BD 2		Improv-ement rate (%)
		Proposed PSO	Proposed QPSO		Proposed PSO	Proposed QPSO	
1	8 × 8	15	15	0	16	16	0
2	10×10	8	8	0	8	8	0
3	15 ×10	13	12	7.692	15	14	6.667
Average Improvement Rate				2.56			2.22

Table 5.7 Results of the BR data instances at two different machine breakdown scenarios

Problem	n×m	BD 1		Improv-ement rate (%)	BD 2		Improv-ement rate (%)
		Proposed PSO	Proposed QPSO		Proposed PSO	Proposed QPSO	
Mk 01	10×6	55	52	5.455	66	66	0.000
Mk 02	10× 6	43	41	4.651	49	45	8.163
Mk 03	15× 8	292	288	1.370	301	301	0.000
Mk 04	15× 8	76	76	0.000	90	88	2.222
Mk 05	15× 4	202	202	0.000	255	253	0.784
Mk 06	10×15	81	79	2.469	97	97	0.000
Mk 07	20× 5	188	185	1.596	211	209	0.948
Mk 08	20×10	615	615	0.000	687	679	1.164
Mk 09	20×10	361	342	5.263	425	421	0.941
Mk 10	20×15	269	265	1.487	302	298	1.325
Average Improvement Rate				2.23			1.55

Table 5.8 Results of the Dauzere-peres instances at two different machine breakdown scenarios

Problem	n×m	BD 1		Improvement rate (%)	BD 2		Improvement rate (%)
		Proposed PSO	Proposed QPSO		Proposed PSO	Proposed QPSO	
1a	10×5	4637	4607	0.647	5069	4967	2.012
2a	10×5	3809	3685	3.255	3858	3814	1.140
3a	10×5	3792	3660	3.481	4239	4056	4.317
4a	10×5	4467	4467	0.000	5359	5324	0.653
5a	10×5	3698	3687	0.297	3780	3813	-0.873
6a	10×5	3781	3790	-0.238	3822	3777	1.177
7a	15×8	4135	3994	3.410	4149	4062	2.097
8a	15×8	3750	3724	0.693	4023	4010	0.323
9a	15×8	3627	3580	1.296	4084	3927	3.844
10a	15×8	4130	4038	2.228	4250	4205	1.059
11a	15×8	3697	3683	0.379	3896	3896	0.000
12a	15×8	3512	3509	0.085	3675	3668	0.190
13a	20×10	4295	4221	1.723	4675	4527	3.166
14a	20×10	3861	3810	1.321	4253	4201	1.223
15a	20×10	3719	3694	0.672	4180	4123	1.364
16a	20×10	3645	3629	0.439	4022	3991	0.771
17a	20×10	3886	3847	1.004	4179	4014	3.948
18a	20×10	3831	3774	1.488	4146	3991	3.739
Average Improvement Rate				1.24			1.67

On the average Improvement Rate of QPSO with respect to PSO, an improvement of 2.56%, 2.23%, 1.24% in a low disruption level (BD1), 2.22%,1.55%,1.67% in a high disruption level (BD2) for Kacem, BR and DP data set, has been achieved for the benchmark problems considered in the study.

The effect of robust measure is demonstrated in the Figure 5.5 and Figure 5.6. Gantt chart is obtained from the schedule of the problem 10×10 of Kacem data is illustrated in Figure 5.5 by using QPSO algorithm. The obtained schedule in Figure 5.5a results a makespan of 7 without machine breakdown. The machine 6 and machine 9 is the busiest machine so the probability of machine breakdown in machine 6 and machine 9 is more. As the break down occurs at machine 6 between 3 and 4 , it can observed from Figure 5.5b that makespan is 8 after the breakdown and the 6 numbers of operations are affected that means the completion time of $O_{4,3}$, $O_{5,3}$, $O_{9,3}$, $O_{10,1}$,$O_{10,2}$ $O_{10,3}$ is deviated from the initial schedule due to machine breakdown. Gantt chart in Figure 5.6 represents

the performance of the multi-objective robustness measure. Multi-objective framework is implemented as per equation 5.6 with the robustness measure RM3. The Figure 5.6a illustrates the performance of a schedule obtained from multi-objective robustness measure without machine breakdown and Figure 5.6b illustrates the performance of a schedule with machine breakdown. When the break down occurs, it is observed that only one operation i.e. $O_{5,3}$ is deviated from the initial schedule. In the both the schedules, the makespan in 7 without any disturbance and makespan after breakdown is 8. The schedule obtained by the robustness measure is most robust and stable as the deviation of operation completion times between the realized schedule and the predictive schedule is minimum.

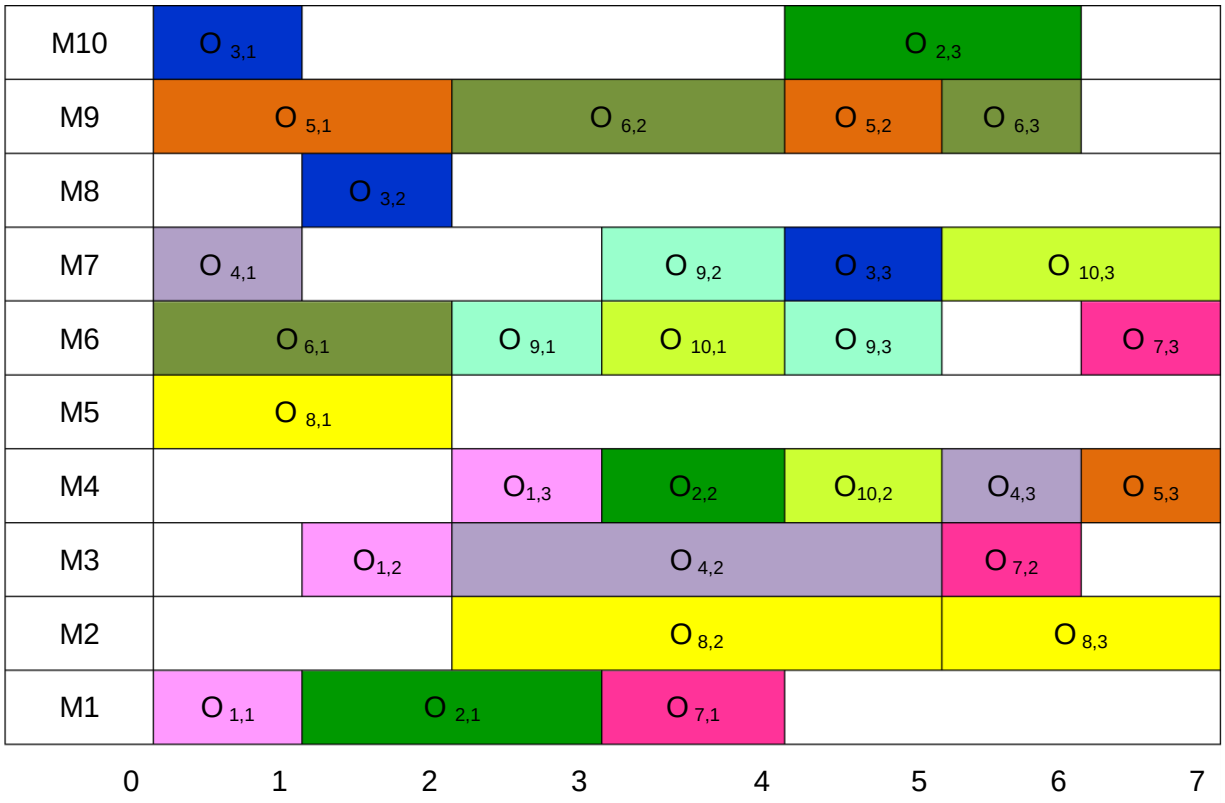


Figure 5.5a Gantt chart obtained by QPSO without machine breakdown
(Problem 10 x 10 from Kacem instance)

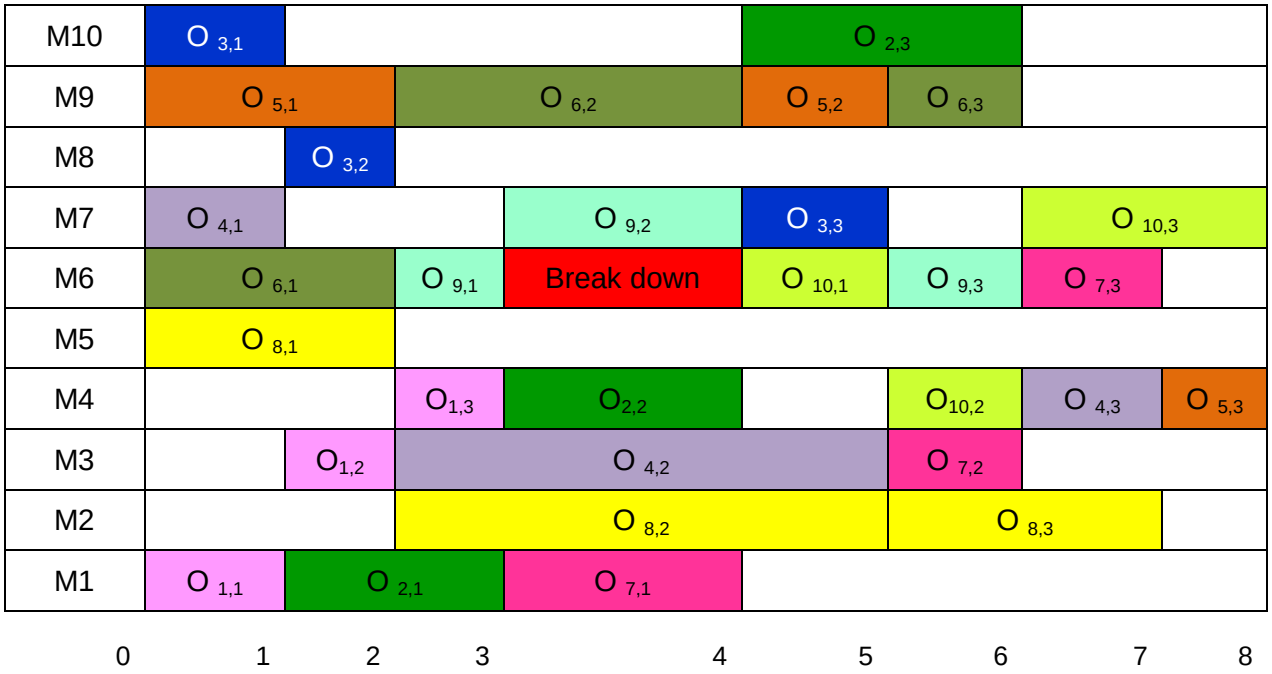


Figure 5.5b Gantt chart obtained by QPSO with machine breakdown.
(Problem 10 x 10 from Kacem instance)

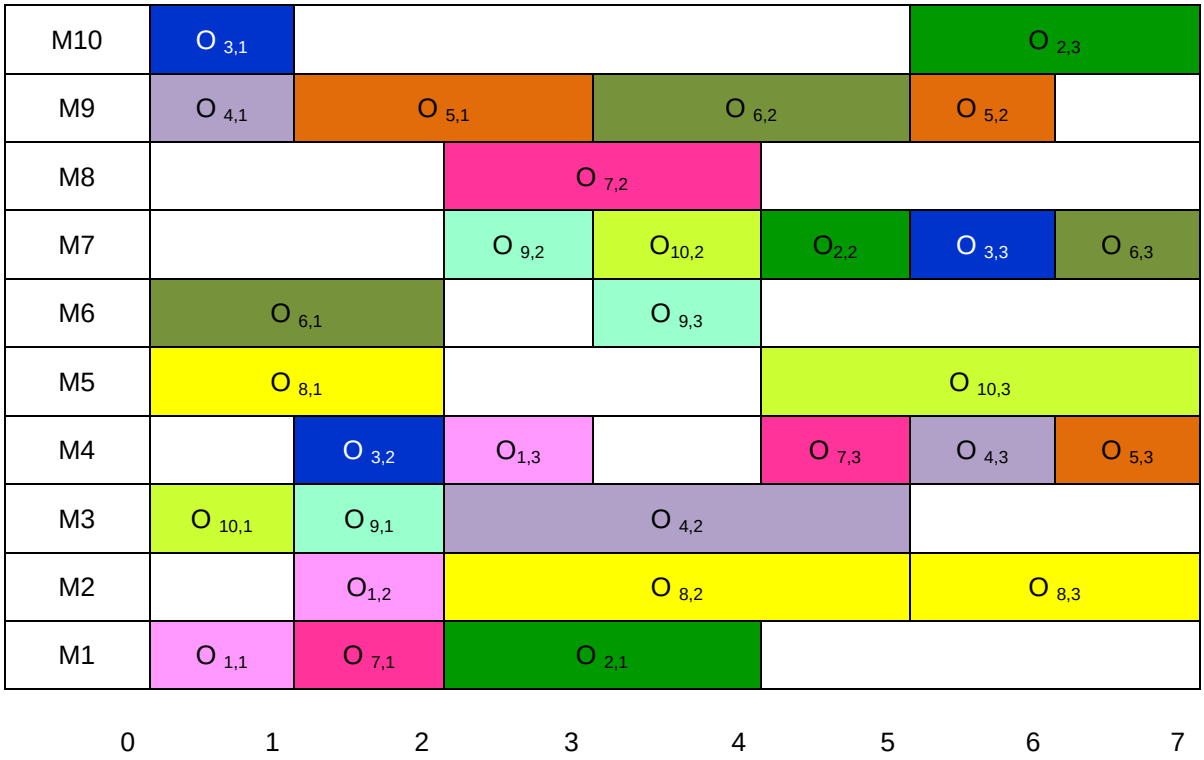


Figure 5.6a Gantt chart obtained by robustness measure without machine breakdown

(Problem 10 x 10 from Kacem instance)

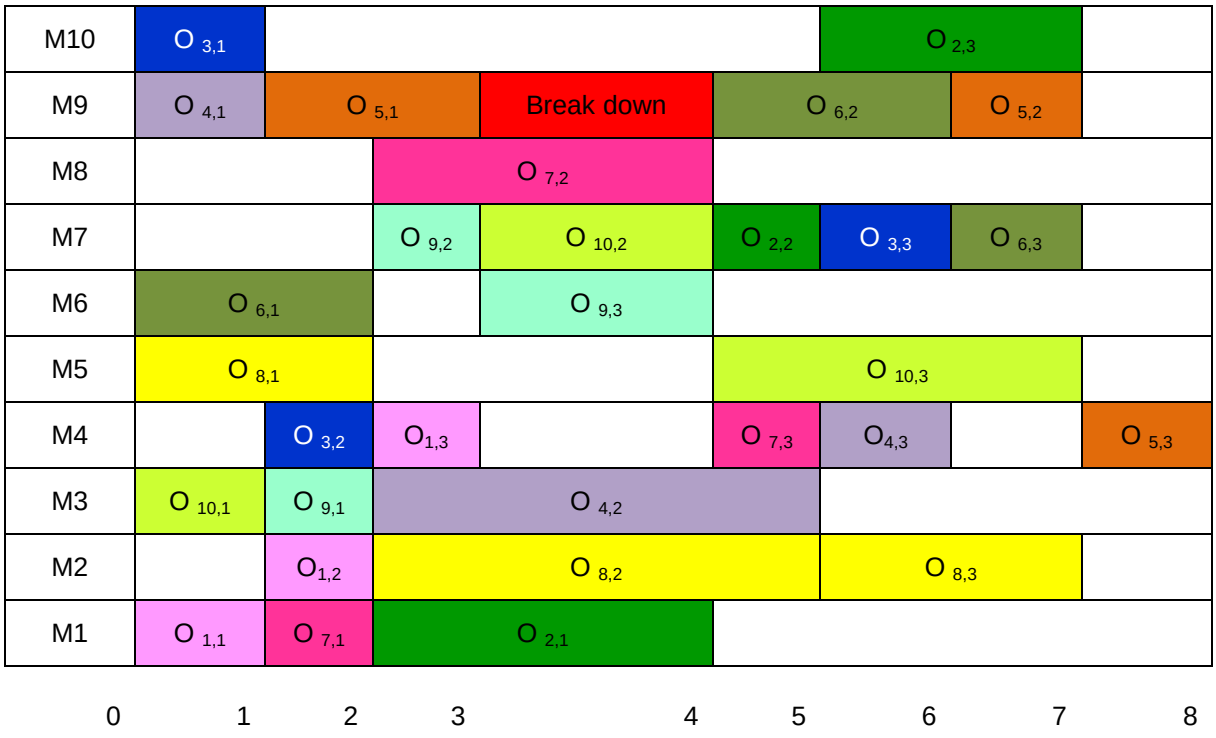


Figure 5.6b Gantt chart obtained by robustness measure machine breakdown (Problem 10 x 10 from Kacem instance)

5.7 Conclusions

In this chapter, a study is made on scheduling of flexible flow shop and job shop scheduling problem under uncertainty situation (i.e. random machine breakdowns) and an efficient quantum particle swarm optimization is proposed to find near-optimal schedules in a disturbance scenario. A multi-objective with robustness measure has been proposed to obtain a robust schedule that minimizes the effect of machine breakdowns in the overall performance such that the makespan is preserved and increases the schedule stability with respect to the deviations of operation completion times between the realized schedule and the predictive schedule. Analysis of variance is conducted to find out significant factors influencing the schedule. ANOVA results revealed that the robustness measure RM3 (the expected realized total flow time) can significantly improve the quality of the schedule in both FFSP and FJSP. The proposed QPSO approach is found to be a good problem solving technique for scheduling problem. The algorithm is applied to seventy seven problem instances of FFSP and thirty three problem instances of FJSP. The obtained results are encouraging in that the proposed QPSO algorithm produces better solutions as compared to proposed PSO in an uncertainty situation. The improvement rate of the proposed QPSO is found to be

2.151% in a low disruption level (BD1) and 1.97% in a high disruption level (BD2) for the FFSP benchmark problems. Similarly, QPSO results an improvement of 2.56%, 2.23% and 1.24% in a low disruption level (BD1), 2.22%, 1.55% and 1.67% in a high disruption level (BD2) for Kacem, BR and DP data set respectively over proposed PSO for FJSP benchmark problems. Therefore, it is concluded that the proposed QPSO algorithm is quite effective in reducing makespan in an uncertainty condition.

In the next chapter, a novel multi-objective particle swarm optimization (MOPSO) technique is proposed and implemented for solving the flexible flow shop scheduling problem (FFSP) and flexible job shop scheduling problem (FJSP) with an objective to minimize makespan, mean flow time and mean tardiness with the goal of finding approximations of the optimal Pareto front and is compared with non-dominated sorting genetic algorithm II (NSGA-II) in terms of four performance metrics.

CHAPTER 6

MULTI-OBJECTIVE FLEXIBLE FLOWSHOP AND JOBSHOP SCHEDULING PROBLEM

6.1 Introduction

Due to intense competition in the market place in terms of shorter product life cycles, customized products and changing demand pattern, effective scheduling has now become an important issue for the growth and survival of manufacturing firms. To sustain in the current competitive environment, it is essential for the manufacturing firms to improve the schedule based on simultaneous optimization of performance measures such as makespan, flow time and tardiness. Minimizing the makespan ensures maximization of the processor utilization, an important criterion from the managerial point of view. Mean flow time criterion bears significance from the operators' point of view as it minimizes maximum in-process time in the shop floor. Tardiness of a job equals to the amount of time required to complete after its due date. Tardiness is important from business perspective as tardy jobs may cause loss of customers and damage reputation. Since all the scheduling criteria are important from business operation point of view, it is vital to optimize all the objectives simultaneously instead of a single objective. According to the shop environments, the shop scheduling can be classified as flow shop, flexible flow shop, job shop, and flexible job shop scheduling. Most existing research addressed these problems with the mono-objective.

The solution strategy for multi-objective scheduling problem (MOSP) is roughly classified into two types such as weighting approach and Pareto-based approach. The weighting approach usually solves by transforming the multi-objective problem into a single-objective problem through assigning a different weight for each objective. The common combination function is known as linear weighted function. However, linear weighted function might not always be able to represent the trade-off relationship between the objectives because determination of weights for objectives is a difficult task. [152, 157, 206] have proposed various algorithms to solve FFSP and FJSP using weighting approach. The Pareto approach on the other hand provides an alternative approach for multi-objective optimization. In Pareto approach, the solutions are compared based on the Pareto dominance relation. Solution 'A' dominates solution 'B', if 'A' is not worse than 'B' for all objectives or is better than 'B' for at least one objective. Solution 'A' is Pareto optimal if it is not dominated by any other solution. The Pareto approach produces a set of Pareto optimal solutions which represent the trade-off between objectives through the distribution of obtained solutions. The user can select the favorite solution directly from the number of Pareto optimal solutions. Multi-objective flexible flow and job shop scheduling problem has been solved incorporating Pareto-

optimal criteria in various algorithms like particle swarm optimization and genetic algorithm [117,161,207,208,209, 210,211].

In this chapter, a novel multi-objective particle swarm optimization (MOPSO) technique is proposed for solving flexible flow shop scheduling problem (FFSP) and flexible job shop scheduling problem (FJSP) with an objective to minimize makespan, mean flow time and mean tardiness with the goal of finding approximations of the optimal Pareto front. In multi-objective optimization problems, convergence and diversity are two important issues. The former specifies the algorithm's capability to find the true Pareto optimal solutions and the latter imitates the algorithm's ability to find as much as possible different Pareto optimal solutions. In order to improve diversity, mutation, a popular operator in genetic algorithm, is embedded in the standard MOPSO algorithm to escape from local optima. However, MOPSO results in a large number of non-dominated solutions. Therefore, maximum deviation theory proposed by Wang [212] has been adopted for ranking the solution to ease the decision making process of choosing the best solution.

6.2 Multi-objective optimization

Multi-objective optimization (MOO) is defined as the problem of finding a vector of decision variables that satisfies all constraints and simultaneously optimizes a vector function whose elements represent the objective functions. Mathematically, the multi-objective optimization problem can be formulized as follows:

$$\text{Minimizing (or Maximizing)}\ \mathrm{F(x)} = \{\mathrm{f_1(x), f_2(x), \ldots\ldots f_q(x)}\}$$

$$\text{subject to}\ \ \mathrm{g(x)} \leq 0,\ \ \mathrm{h(x)} = 0$$

A MOP solutions minimizes (or maximizes) the components of a vector $\mathrm{F(x)}$ where x is a n-dimensional decision variable vector $\mathrm{X} = (\mathrm{x_1, x_2, \ldots\ldots\ldots x_n})$ and $\mathrm{g(x)} \leq 0,\ \ \mathrm{h(x)} = 0$ are set of constraints that determine the feasible solution area in minimizing (or maximizing) $\mathrm{F(x)}$ with 'q' objective functions. In this study, the following objectives of FFSP and FJSP are to be minimized:

Objective 1$(\mathrm{F_1})$: The first objective is to minimize the makespan $(\mathrm{C_{max}})$ i.e, the completion time of all jobs in the last stage.

$$\mathrm{C_{max}} = \max\{\mathrm{C_I}\} \tag{6.1}$$

where $\mathrm{C_i}$ is the completion time of job i at last stage

Objective 2$(\mathrm{F_2})$: The second objective is to minimize the mean tardiness $(\overline{\mathrm{T}})$ i.e, the amount of time by which the completion time of job i differs from the due date.

$$\overline{T} = \frac{1}{n}\sum_{i=1}^{n} \max\{0, (C_i - d_i)\} \quad (6.2)$$

where d_i is the due date of job i

Objective 3 (F_3): The third objective is to minimize the mean Flow time $(\bar{F})$ i.e, the amount of time spent by job in the shop.

$$\overline{F} = \frac{1}{n}\sum_{i=1}^{n}(C_i - r_i) \quad (6.3)$$

where r_i is the release date.

6.3 Multi-objective particle swarm optimization (MOPSO)

Multi-objective optimization (MOO) has been an active area of research in last two decades. Such problems arise in many applications where two or more objective functions have to be optimized simultaneously. PSO has been extended for solving the MOO problems, which is generally known as the multi-objective particle swarm optimization (MOPSO). The main difference between a basic PSO (single-objective) and MOPSO is the distribution of g_{best}. In MOPSO algorithm, g_{best} must be redefined in order to obtain a set of non-dominated solutions (Pareto front). In single-objective problems, there is only one g_{best} exists. In MOO problems, more than one conflicting objectives will be optimized simultaneously. There are multiple numbers of non-dominated solutions which are located on or near the Pareto front. Therefore, each non-dominated solution can be the g_{best}. Extending PSO to handle multi-objectives have been proposed by Mostaghim and Teich [213] and Wang and Singh [214]. Coello et al.[151] have proposed a MOPSO algorithm which adopts an external repository and mutation operator for finding out Pareto-optimal set of solutions.

6.3.1 Proposed MOPSO algorithm

Real world problems involve simultaneous optimization of numerous contradistinctive and conflicting nature objectives. When all objectives are considered, these solutions are optimum in the sense that none of the other solutions in the search area are exceptionally good to another solution. These solutions are called as Pareto-optimal solutions. The image of the efficient set in the objective space is named as non-dominated set as each solution dominates the other solution. To identify the non-dominance, each solution is compared with every single solution and checked for satisfying the rules given below for the solution under consideration.

$$\text{Obj. } 1[l] > \text{Obj. } 1[m] \text{ and Obj. } 2[l] \geq \text{Obj. } 2[m] \quad (6.4)$$

$$\text{Obj. } 1[l] \geq \text{Obj. } 1[m] \text{ and Obj. } 2[l] > \text{Obj. } 2[m] \quad (6.5)$$

where l and m correspond to solution number in the population. $Obj.1$ and $Obj.2$ are two objective function values.

The multi-objective optimization aims at two objectives:

(a) Converging to the Pareto-optimal solution set;

(b) Maintaining diversity and distribution in solutions.

While solving single-objective optimization problems, the g_{best} that each particle uses to update its position is completely determined once a neighborhood topology is established. However, in the case of multi-objective optimizations problems, each particle might have a set of g_{best} from which just one can be selected in order to update its position. Such set of g_{best} is usually stored in a different place from the swarm known as external archive 'A_t'.This is a repository in which the non-dominated solutions found so far are stored. The MOPSO maintains an external archive 'A_t'of non-dominated solutions of the population which is updated after every iteration. The global archive 'A_t'is empty in the beginning and can store a user-specified maximum number of non-dominated solutions. In case the number of non-dominated solutions exceeds the maximum size of the archive, some individuals are cropped. There are several methods of controlling the external archive such as maximin fitness based size control [211], epsilon-dominance based size control [213] and crowding distance based size control [215]. Archive size control is critical because the number of non-dominated solutions can grow very fast although there are studies where archive size is unconstrained [216].

Crowding distance technique has been extensively applied in evolutionary multi-objective algorithms to promote diversity. The use of crowding distance measure in MOPSO for g_{best} selection was first made in Raquel and Naval [215]. The approach is quite capable in converging towards the Pareto front and generating a well-distributed set of non-dominated solutions. In this study, crowding distance approach has only been applied to make g_{best} selection. Crowding distance factor is defined to show how much a non-dominated solution is crowded with other solutions. The crowding distance (CD) factor of a solution provides an estimate of the density of solutions surrounding that solution [217,218]. Figure.6.1 shows the calculation of the crowding distance of point k which is an estimate of the size of the largest cuboid enclosing k without including any other point. CD factor of boundary solutions which have the lowest and highest objective function values (f_{max} and f_{min} respectively) are given an infinite crowding distance values. For other solutions, CD factor for the solution k is calculated by following relation.

$$CD_k = \frac{(f_{k+1} - f_{k-1})}{(f_{max} - f_{min})} \tag{6.6}$$

Finally, the overall crowding factor is computed by adding the entire individual crowding distance values in each objective function.

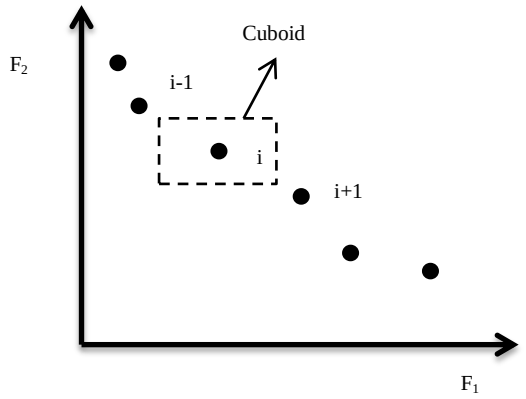


Figure 6.1 The crowding distance

The non-dominated solutions in 'A_t'are sorted in descending crowding distance values and top 10% of them are randomly used as g_{best} guides

Particle swarm optimization typically converges relatively rapidly at the beginning of the search and then slows down or stagnates due to loss of diversity in the population [195]. To overcome this drawback, mutation, a widely used operator in genetic algorithm, is used to introduce diversity in the search procedure. When the change of the whole archive tends to decrease, the mutation process will begin. If the number of iteration is less than the product of maximum number of iteration and probability of mutation then only the mutation is performed on the position of the particle. Given a particle, a randomly chosen variable, say m_p, is mutated to assume a value m'_p as given by following equation.

$$m'_p = \begin{cases} m_p + \Delta(t, UB - m_p) & \text{if flip} = 0 \\ m_p - \Delta(t, m_p - LB) & \text{if flip} = 1 \end{cases} \tag{6.7}$$

when flip denotes the random event of returning 0 or 1. UB and LB denote the upper and lower bound of the variable m_p respectively. The function $\Delta(t, x)$ returns a value in the range $(0, x)$ such that the probability of $\Delta(t, x)$ being close to 0 increases as t increases.

$$\Delta(t, x) = x \times \left(1 - r^{\left(1 - \frac{t}{MAXT}\right)^b}\right) \tag{6.8}$$

where r is the random number generated in the range [0, 1], MAXT is the maximum number of iterations and t is the number of iteration. The parameter b determines the degree of dependence of mutation on the iteration number.

To summarize, the main difference between a basic PSO (single-objective) and MOPSO is the distribution of g_{best}. In single-objective problems, there is only one g_{best} exists. In MOPSO algorithm, g_{best} must be redefined in order to obtain a set of non-dominated solutions (Pareto front). Therefore, multiple numbers of non-dominated solutions are located on or near the Pareto front. Each non-dominated solution can be a g_{best}. The important feature of MOPSO is that the individuals also maintain a personal archive which is known as p_{best} archive with a maximum size. The p_{best} archive contains the most recent non-dominated positions a particle has encountered in the past. In every iteration t, each particle i is allocated with two guides p_{best} and g_{best} from its p_{best} archive and swarms global archive 'A_t'. After the guide selection, positions and velocities of particles are updated according to the equation 6.9 and equation 6.10 where v_{ij}^t represents velocity and x_{ij}^t is the position value of the i^{th} particle with respect to j^{th} dimension. Maximum number of generations is set as termination criterion. The complete algorithm for MOPSO is shown as follows:

MOPSO Algorithm

1. For i= 1 to M (M is the population size)
    a. Initialize position of the particles randomly
    b. Initialize v_{ij}^t= 0 (v is the velocity of each particle)
    c. Evaluate each particle's fitness
    d. Compare each particle's fitness with the particle's p_{best}. Compare the fitness with the population's overall previous best
    e. Find out the personal best (p_{best}) and global best (g_{best}).
2. End For
3. Initialize the iteration counter t= 0
4. Store the non-dominated vectors found into archive 'A_t'
   ('A_t'is the external archive that stores non-dominated solutions found)
5. Repeat
    a. Compute the crowding distance values of each non-dominated solution in the archive 'A_t'
    b. Sort the non-dominated solutions in 'A_t'in descending crowding distance values
    c. For i= 1 to M

i. Randomly select the global best guide from a specified top 10% of the sorted archive 'A_t' and store its position to g_{best}.

ii. Compute the new velocity:

$$v_{ij}^{t} = w^{t-1}v_{ij}^{t-1} + c_1 r_1\left(p_{ij}^{t-1} - x_{ij}^{t-1}\right) + c_2 r_2\left((A_t)_{ij}^{t-1} - x_{ij}^{t-1}\right) \quad (6.9)$$

($(A_t)_{ij}^{t-1}$ is the global best guide for each nondominated solution)

iii. Calculate the new position of $x_{ij}^{t} = x_{ij}^{t-1} + v_{ij}^{t}$ (6.10)

iv. If (t< (t_{max} * PMUT), then perform mutation on x_{ij}^{t}.

(t_{max} is the maximum number of iterations and PMUT is the probability of mutation)

v. Evaluate x_{ij}^{t}

d. End For

e. Insert all new non-dominated solution into archive 'A_t' if they are not dominated by any of the stored solutions. All dominated solutions in the archive are removed by the new solution from the archive. If the archive is reached its maximum, the solution to be substituted is determined by the following steps:

i. Compute the crowding distance values of each non-dominated solution in the archive 'A_t'

ii. Sort the non-dominated solutions in archive 'A_t'in descending crowding distance values

iii. Randomly select a particle from a specified bottom 10% of the sorted archive 'A_t' and replace it with the new solution

f. Update the personal best solution of each particle. If the current p_{best} dominates the position in the memory, the particle position is updated.

g. Increment iteration counter t

6. Until maximum number of iterations is reached.

6.3.2 Solution ranking by maximum deviation theory

Since MOPSO results in a large number of non-dominated solutions, choosing a best solution depends on decision maker's judgment and intuition. Usually, multi-attribute decision making (MADM) approaches are adopted to obtain scores for the solutions and the solution exhibiting maximum score is selected as the best one. However, the weights assigned in multi-attribute decision making process for converting multiple objectives into a single equivalent objective score are reasonably subjective in nature and affect the decision of ranking the alternative solutions considerably. In order to avoid uncertainty of subjective assigning of weights from the experts and extract the accurate information

from the available data, maximum deviation theory (MDT) suggested by Wang [212] is adopted in this work**.** The basic idea of MDT rests on smaller weight should be assigned to the attribute having similar values in comparison to the attribute having larger deviations.

The non-dominated solutions obtained in MOPSO solutions are used as the decision matrix. Every element of the decision matrix denotes the value of j^{th} attribute for i^{th} alternative where i=1, 2 ...n, and j=1, 2...m. Normalization of each attribute is carried out to transform different scales and units among various attributes into a common measurable scale. The normalization of the attribute depends on its type such as “higher the better” and “lower the better”. The following equations are used for normalization of attributes.

$x_{ij}^{*} = \frac{\max_i\{x_{ij}\} - x_{ij}}{\max_i\{x_{ij}\} + \min_i\{x_{ij}\}}$, for lower the better attributes (6.11)

$x_{ij}^{*} = \frac{x_{ij} - \min_i\{x_{ij}\}}{\max_i\{x_{ij}\} - \min_i\{x_{ij}\}}$, for higher the better attributes (6.12)

The difference of performance values for each alternative is computed. For the attribute $\{A_j | j=1, 2\ldots m\}$, the deviation value of the alternative $\{S_i | i = 1, 2 \ldots.n\}$ from all the other alternatives can be computed by the following equation

$$D_{ij}(w_j) = \sum_{i=1}^{N} d(\tilde{r}_{ij}, \tilde{r}_{lj}) w_j \quad (6.13)$$

where w_j is the weight of the attributes to be calculated and $D_{ij}(w_j)$ is the deviation value of the alternatives.

The total deviation values of all alternatives with respect to other alternatives for the attribute $\{A_j | j =1, 2\ldots m\}$ can be computed by the following relation.

$$D_j(w_j) = \sum_{i=1}^{N} D_{ij}(w_j) = \sum_{i=1}^{N}\sum_{l=1}^{N} d(\tilde{r}_{ij}, \tilde{r}_{lj})\, w_j \quad (6.14)$$

where $D_j(w_j)$ is the total deviation value of all the alternatives.

The deviation of all the attributes along all the alternatives can be calculated by the relation

$$D(w_j) = \sum_{j=1}^{M} D_j(w_j) = \sum_{j=1}^{M}\sum_{i=1}^{N}\sum_{l=1}^{N} d(\tilde{r}_{ij}, \tilde{r}_{lj})\, w_j \quad (6.15)$$

where $D(w_j)$ is deviation of all the attributes along all the alternatives.

A linear programming model is constructed for finding out the weight vector w to maximize all deviation values for all the attributes and is given by

$$\begin{cases} D(w_j) = \sum_{j=1}^{M}\sum_{i=1}^{N}\sum_{l=1}^{N} d(\tilde{r}_{ij}, \tilde{r}_{lj})\, w_j \\ s.t \sum_{j=1}^{M} w_j^2 = 1, w_j \geq 0, j = 1,2, \ldots\ldots, M \end{cases} \quad (6.16)$$

A Lagrange function is constructed for solving the above model

$$L(w_j, \alpha) = \sum_{j=1}^{M} \sum_{i=1}^{N} \sum_{l=1}^{N} d(\tilde{r}_{ij}, \tilde{r}_{lj})\, w_j + \alpha\left(\sum_{j=1}^{M} w_j^2 - 1\right) \quad (6.17)$$

where, α is the Lagrange multiplier. The partial derivative of L (w_j,α) with respect to w_j and α are

$$\begin{cases} \dfrac{\partial L}{dw_j} = \displaystyle\sum_{i=1}^{N} \sum_{l=1}^{N} d(\tilde{r}_{ij}, \tilde{r}_{lj}) + 2 \propto w_j = 0 \\ \dfrac{\partial L}{d \propto} = \displaystyle\sum_{j=1}^{M} w_j^2 - 1 = 0 \end{cases} \quad (6.18)$$

Further, w_j and α values are calculated from equation 6.17 and 6.18

$$\begin{cases} 2 \propto = -\sqrt{\displaystyle\sum_{j=1}^{M} \left(\sum_{i=1}^{N} \sum_{l=1}^{N} d(\tilde{r}_{ij}, \tilde{r}_{lj}) \right)^2} \\ w_j = \dfrac{\sum_{i=1}^{N} \sum_{l=1}^{N} d(\tilde{r}_{ij}, \tilde{r}_{lj})}{\sqrt{\sum_{j=1}^{M} \left(\sum_{i=1}^{N} \sum_{l=1}^{N} d(\tilde{r}_{ij}, \tilde{r}_{lj}) \right)^2}} \end{cases} \quad (6.19)$$

The normalized attribute weights can be further determined by the following relation

$$w_j = \frac{\sum_{i=1}^{N} \sum_{l=1}^{N} d(\tilde{r}_{ij}, \tilde{r}_{lj})}{\sum_{j=1}^{M} \sum_{i=1}^{N} \sum_{l=1}^{N} d(\tilde{r}_{ij}, \tilde{r}_{lj})} \quad (6.20)$$

The non-dominated solutions obtained through MOPSO algorithm are ranked by estimating the composite score of each solution by addition of the weighted performance of all attributes. Considering the ranking of the solutions, the tool engineer may choose suitable parametric setting from the top ranking solutions to justify the objectives set by the industry.

6.4 Results and discussions

The most widely used scalarization method for multiobjective optimization is the weighted sum method. This method combines multiple objectives into an aggregated scalar objective function by multiplying each objective function by a weighting factor and summing up all terms as it has been described in session 5.4. In many real-life decisions making process for the multiobjective problems it is not possible to know in advance the relevance of each objective. Even though the decision maker has this information, sometimes it is difficult to decide that what weights should be assigned to the objectives for a better performance. In such situations for a MOP, a Pareto optimality is prescribed

to find solutions (Pareto optimal) that cannot be improved in one objective without deteriorating other's performance. As the Pareto optimality approach provides a set of non-dominated solutions and the knowledge of all these optimal solutions allow the decision maker to be more conscious of the trade-offs among the different objectives while taking their decisions. To illustrate the effectiveness, we have compared with the linearly combined (scalarization method) with pareto optimality approach which is solved by MOPSO technique to find non dominated solutions and MDT as a decision maker to find out the best solution from the non-dominated solutions. Table 6.1 illustrates the comparison of Ten Problem of FFSP with machine breakdown has taken from the previous chapter (Chapter 5).

The first and second columns symbolize the name and lower bound of the problem respectively. The third column represents the best makespan obtained by the scalarization method from proposed QPSO algorithm. The fifth column represents the makespan results from proposed MOPSO and maximum deviation theory during a low disruption level (BD1) and seven and ninth column represents the makespan results during a high disruption level (BD2). The fourth, sixth, eighth and tenth column represents the percentage deviation from the lower bound of the problem according to equation 3.18.

Table 6.1 Comparison between scalarization method and MOPSO for FFSP at two different machine breakdown scenarios

Problem	LB of Makes-pan	BD 1				BD 2			
		Best C_{max} by Scalariz-ation Method	%PD From LB	MOPSO	%PD from LB	Best C_{max} by scalar-zation method	%PD From LB	MOPSO	%PD from LB
j10c5a2	88	97	10.23	94	6.82	110	25.00	102	15.91
j10c5a3	117	126	7.69	119	1.71	148	26.50	132	12.82
j10c5c1	68	82	20.59	75	10.29	104	52.94	97	42.65
j10c5c2	74	93	25.68	88	18.92	110	48.65	102	37.84
j10c10a1	139	176	26.62	151	8.63	189	35.97	180	29.50
j10c10a2	158	170	7.59	169	6.96	192	21.52	187	18.35
j15c5a1	178	194	8.99	182	2.25	222	24.72	209	17.42
j15c5a2	165	181	9.70	170	3.03	215	30.30	192	16.36
j15c10a2	200	220	10.00	208	4.00	240	20.00	221	10.50
j15c10a3	198	213	7.58	204	3.03	234	18.18	219	10.61
Average percentage deviation			13.467		6.564		30.378		21.196

Table 6.1 illustrate that most of the makespan values obtained by the MOPSO in a machine breakdown scenario have smaller PD from lower bound over the scalarization method. The average percentage deviation is 13.67 for scalarization method and 6.564 for MOPSO in low level disturbance and 30.378, 21.196 for scalarization method and MOPSO in high level disturbance respectively.

In the present work, multi-objective particle swarm optimization (MOPSO) has been developed for solving the flexible flow shop scheduling problem (FFSP) and the flexible job shop scheduling problem (FJSP) with bi-objective criteria i.e. minimize makespan as primary objective and mean flow time and mean tardiness as secondary objective with the goal of finding approximations of the optimal Pareto front. In the proposed MOPSO algorithm, problem representation presented in section 3 is used to solve the FJSP. The algorithm is implemented in Matlab 7 on a Pentium IV running at 2 GHz on the Windows XP operating system.

6.4.1 Result analysis for multi-objective FFSP

The proposed algorithm is tested on seventy seven instances of FFSP from Carlier and Neron's [196]. The effectiveness of the proposed MOPSO algorithm is compared with another popular multi-objective algorithm known as non-dominated sorting genetic algorithm II (NSGA-II) which was first introduced by Deb et al. [217] and successfully applied in many multi-objective problems [219, 220, 221].The NSGA-II is coded in Matlab 7 as per the FFSP problem represented in section 3.4.

Based on exhaustive experimentation, Figure 6.2 to 6.9 are drawn to show the Pareto front between makespan and mean flow time and makespan and mean tardiness for eight benchmark instances of FFSP. The pareto fronts reveal that a small decrease of makespan can cause a large increase in the other conflicting objective. The results convey two messages: (1) Focusing on optimizing a single objective may result in bad performance of the other objective (2) The trade-off relationship between the objectives is not always easy to predict.

.

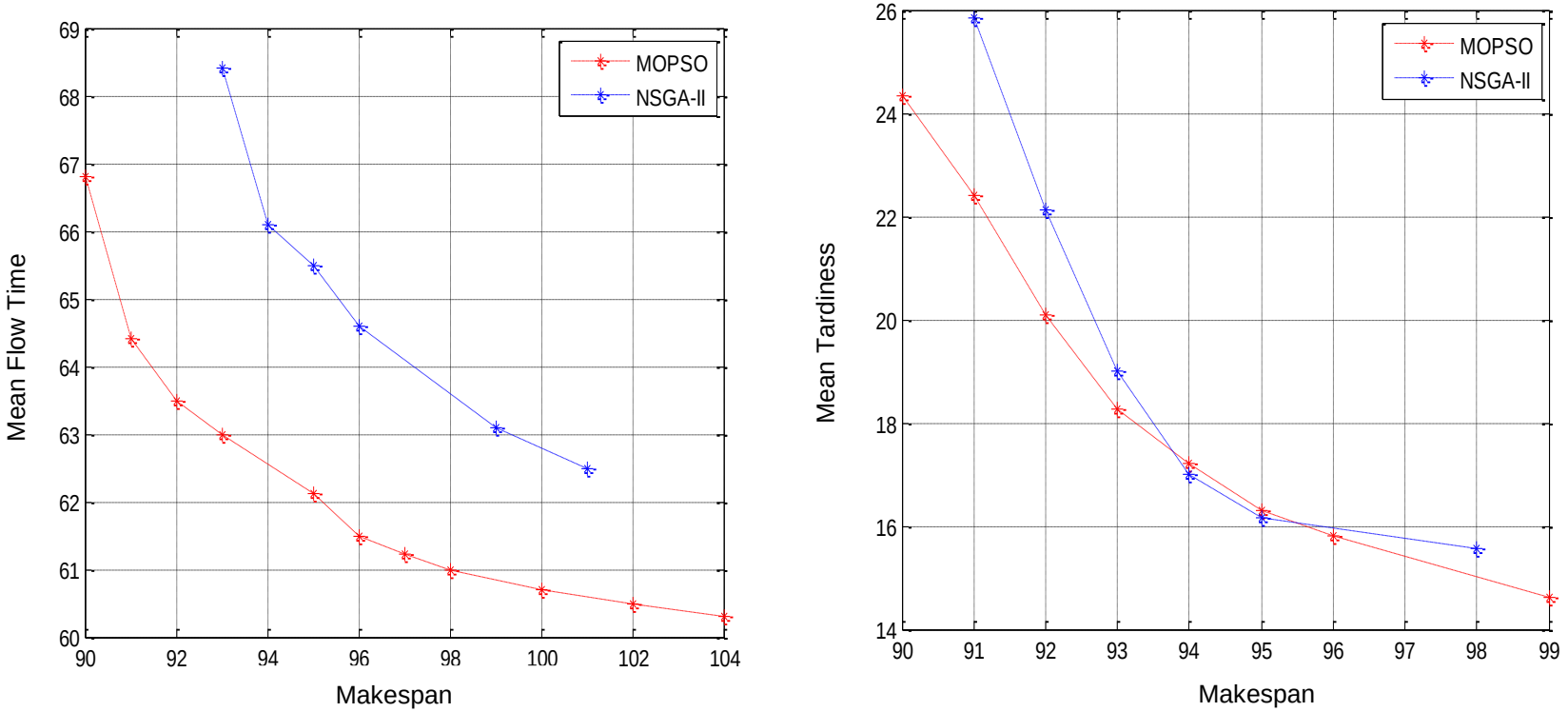


Figure 6.2 Pareto front obtained by the proposed MOPSO and NSGA-II for the instance J10c5a2

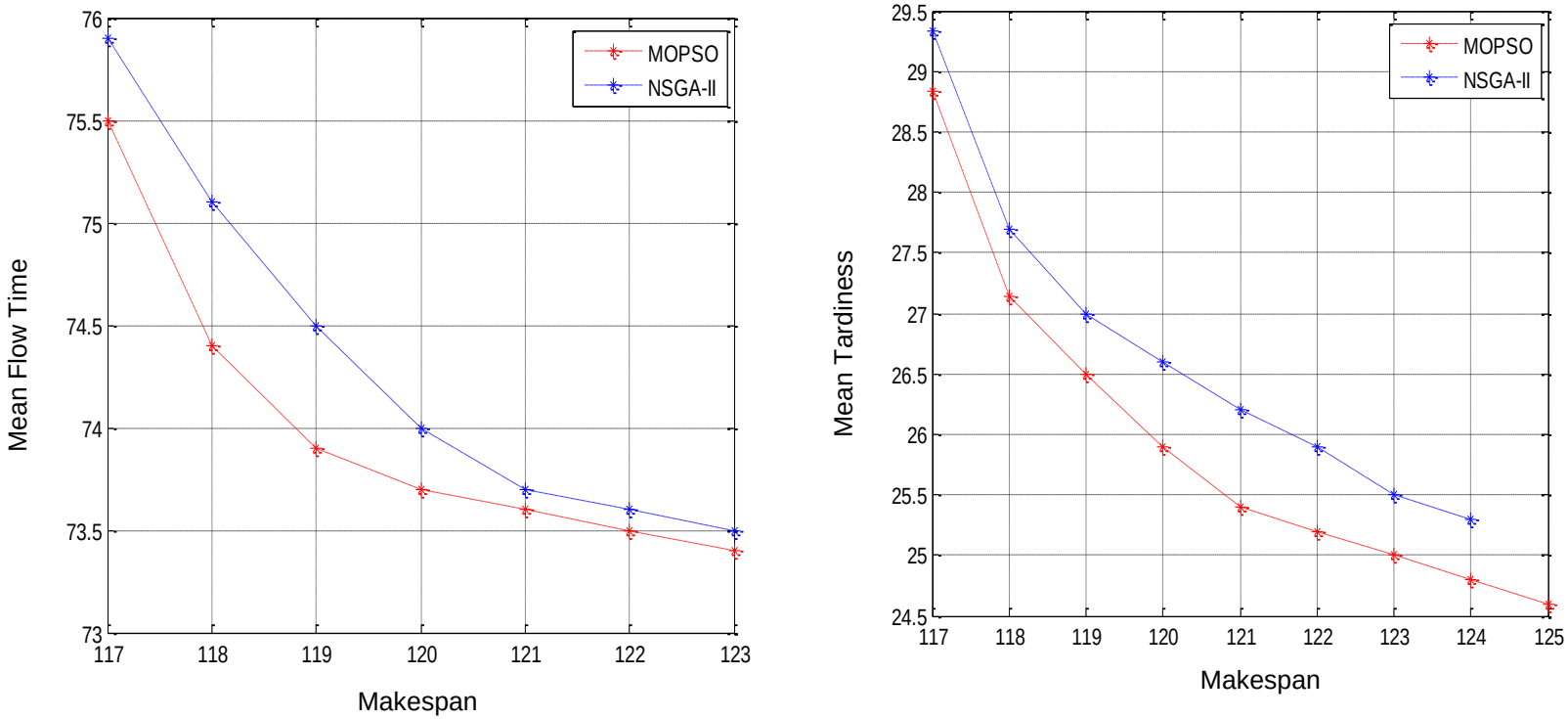


Figure 6.3 Pareto front obtained by the proposed MOPSO and NSGA-II for the instance J10c5a3

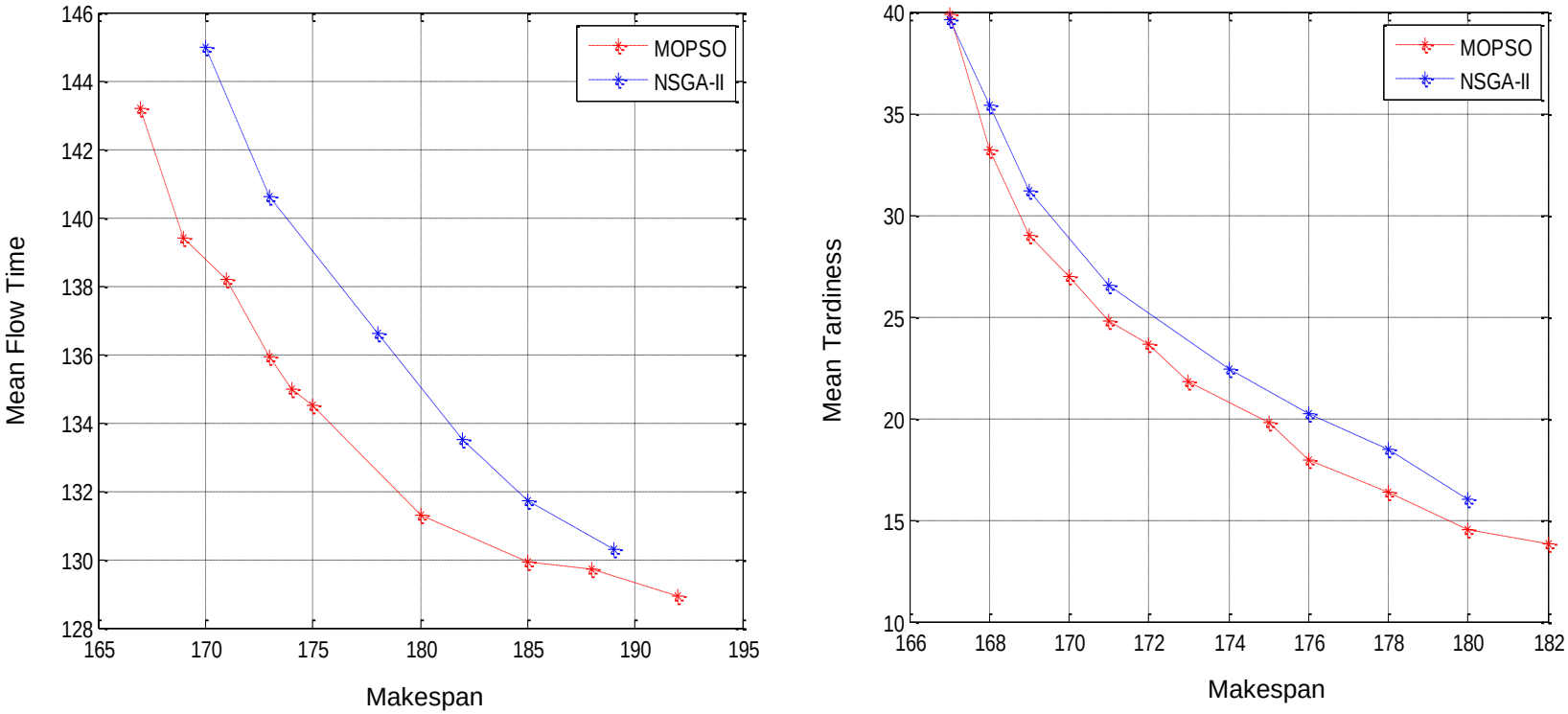


Figure 6.4 Pareto front obtained by the proposed MOPSO and NSGA-II for the instance J10c10a2

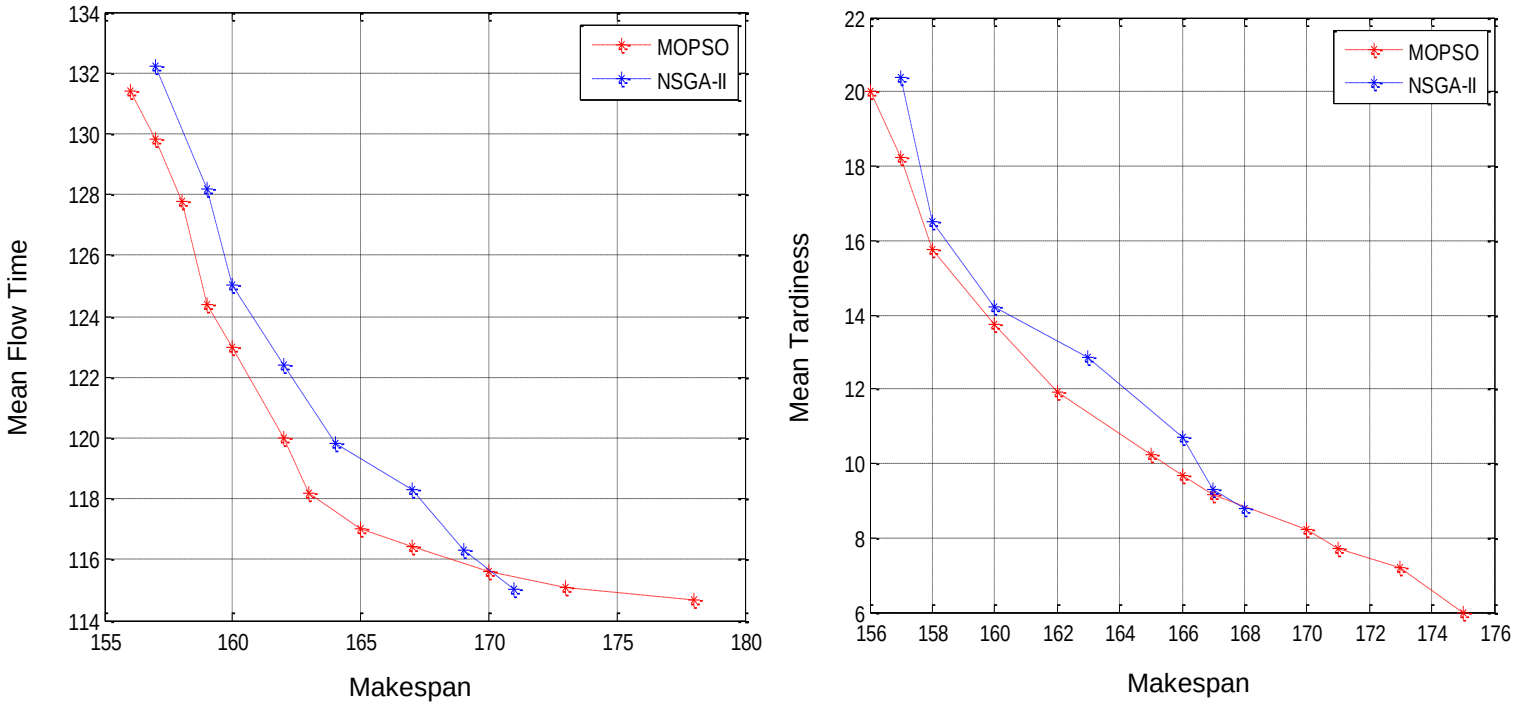


Figure 6.5 Pareto front obtained by the proposed MOPSO and NSGA-II for the instance J10c10a3

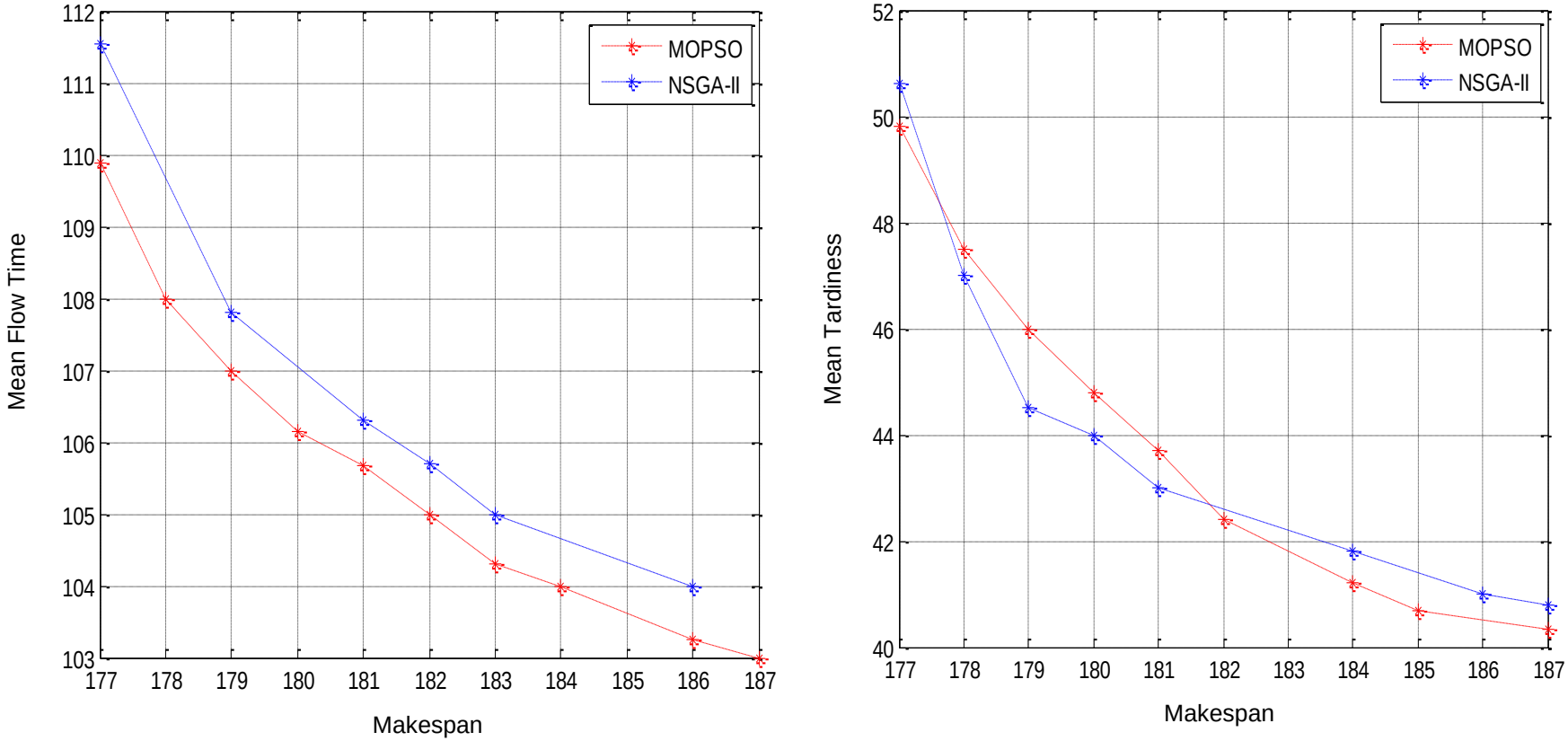
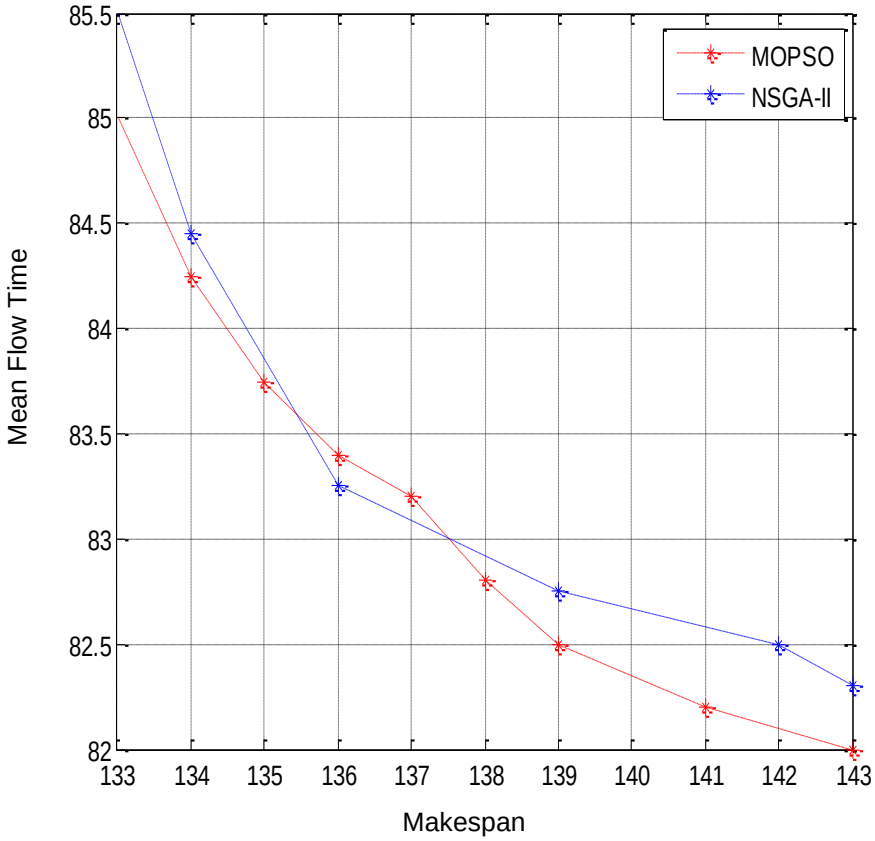


Figure 6.6 Pareto front obtained by the proposed MOPSO and NSGA-II for the instance J15c5a1



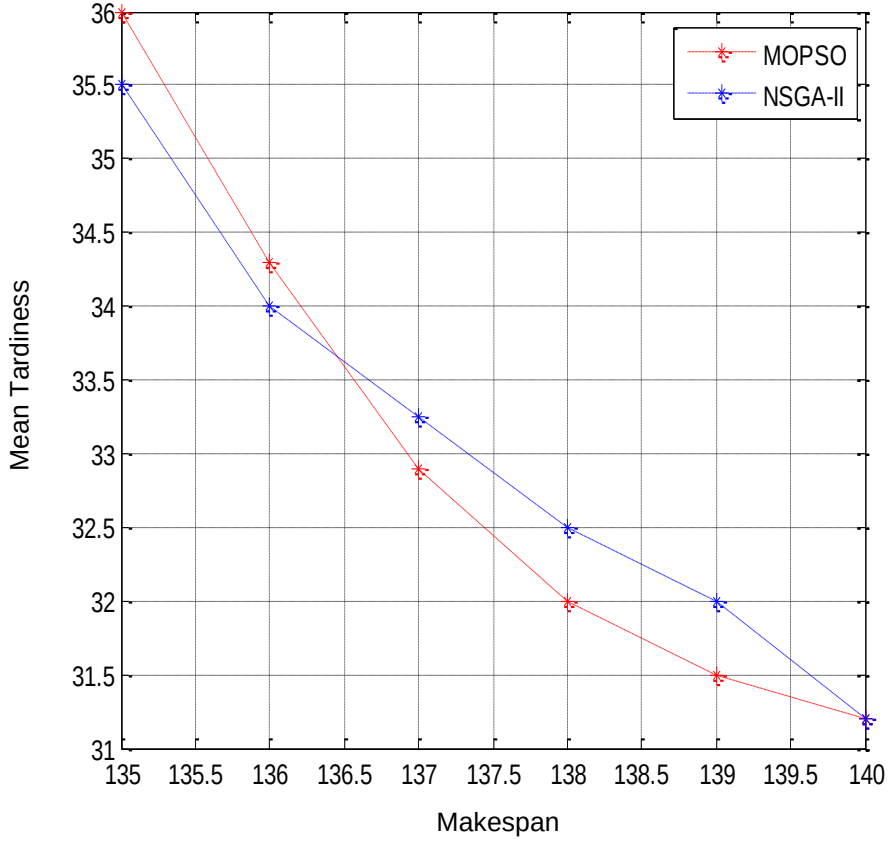


Figure 6.7 Pareto front obtained by the proposed MOPSO and NSGA-II for the instance J15c5a3

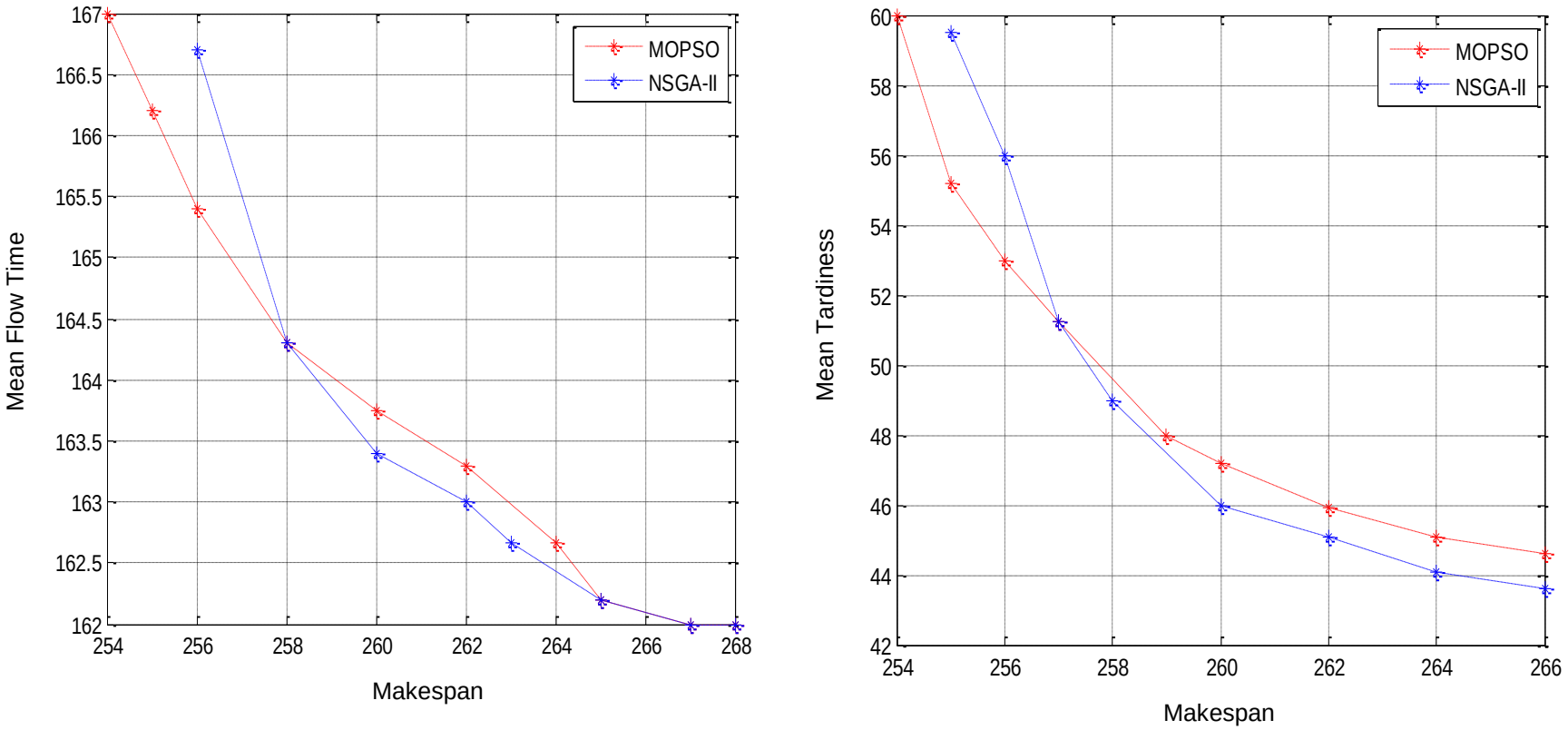


Figure 6.8 Pareto front obtained by the proposed MOPSO and NSGA-II for the instance J15c10a1

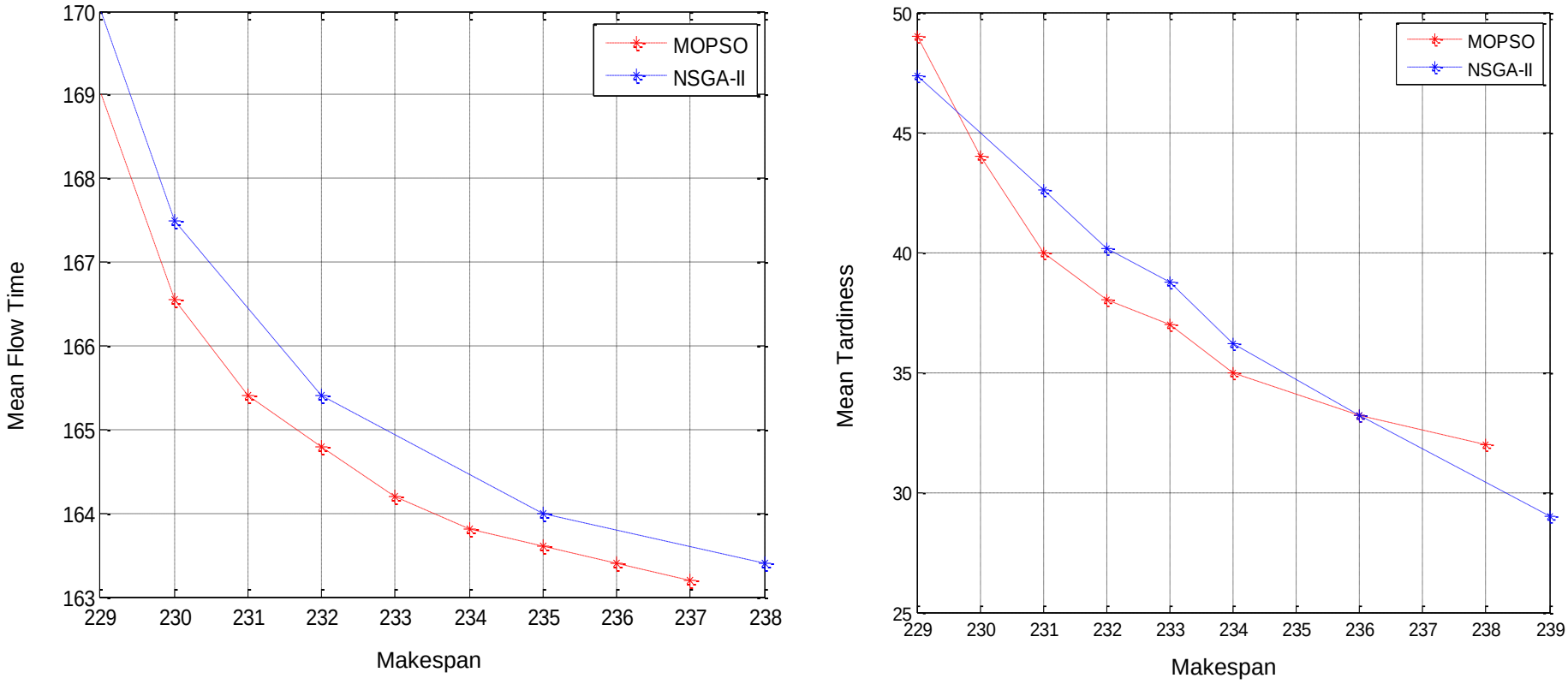


Figure 6.9 Pareto front obtained by the proposed MOPSO and NSGA-II for the instance J15c10a2

There are two goals in a multi-objective optimization: (i) convergence to the Pareto-optimal set (ii) maintenance of diversity in solutions of the Pareto-optimal set. These two tasks cannot be measured adequately with one performance metric. Many performance metrics have been suggested to evaluate the non-dominated solutions [217, 222]. To evaluate comprehensively the non-dominated solutions obtained by the MOPSO and NSGA-II algorithm, four performance metrics are considered in this work. The following performance measures are used to compare the results of non-dominated solutions obtained by multi-objective algorithms.

Mean ideal distance (MID): The MID measurement presents the proximity between non-dominated solutions and ideal point (0, 0). Algorithm A is considered to have more opportunity to reach the Pareto frontier than algorithm B if A has the lower value of MID than B. MID of algorithm can be obtained by the following formulation.

$$\mathrm{MID} = \frac{\sum_{i=1}^{n} C_i}{n} \tag{6.21}$$

where n is the number of non-dominated solutions and $C_i = \sqrt{f_{1i}^2 + f_{2i}^2}$

f_{1i} and f_{2i} are the objective function values for solution i. The performance of the algorithm will be better if the value of MID is lower.

The rate of achievement to two objectives simultaneously (RAS): The value of this measure is calculated from the following relation. Smaller value of this criterion indicates a higher quality solution.

$$\mathrm{RAS} = \frac{\sum_{i=1}^{n} |f_{1i} - f_1^{best}| + |f_{2i} - f_2^{best}|}{n} \tag{6.22}$$

f_1^{best} and f_2^{best} are the best solutions in the non-dominated sets for objectives 1 and 2.

Spread of non-dominance solutions (SNS): The spacing metric aims at assessing the spread (distribution) of vectors throughout the set of non-dominated solutions. This criterion, which is known as an indicator of diversity, is calculated from the following relation:

$$\mathrm{SNS} = \sqrt{\frac{\sum_{i=1}^{n} (\mathrm{MID} - C_i)^2}{n-1}} \tag{6.23}$$

Diversification matrix (DM): This performance measure gives an indication of the diversity of solutions obtained from a given algorithm.

$$\mathrm{DM} = \sqrt{(\max f_1 - \min f_1)^2 + (\max f_2 - \min f_2)^2} \tag{6.24}$$

where $\max f_1$ and $\max f_2$ is the maximum objective functions value of the of non-dominated solutions and $\min f_1$ and $\min f_2$ is the minimum objective functions value of

the of non-dominated solutions. Larger values of SNS and DM are indicative of higher quality solutions.

The effectiveness of the algorithms is tested by solving seventy seven different benchmark problems of Carlier and Neron's data set [196]. The results obtained by the proposed algorithms are compared in terms of the performance metrics with the NSGA-II. Table 6.2 and Table 6.3 Illustrate the comparative results of two algorithms with respect to four performance measures. From Table 6.2 and Table 6.3, it can be concluded that MOPSO outweighs the NSGA-II algorithms in all metrics in terms of the number of optimum results out of twenty eight test problems. It is observed from that Table 6.2 that the proposed MOPSO superior to the NSGA-II in 61, 54, 55 and 55 out of 77 test problems with respect to MID, RAS, SNS and DM performance measures respectively for the objectives of makespan and mean flow time. Table 6.3 indicates that MOPSO performs superior to NSGA-II in 58, 51, 57 and 64 out of 77 test problems in MID, RAS, SNS and DM performance measures respectively for the objective of makespan and mean tardiness.

Table 6.2 Performance metrics of Pareto front obtained by the objective of makespan and mean flow time

Problem	MID		RAS		SNS		Diversity (DM)	
	MOPSO	NSGA-II	MOPSO	NSGA-II	MOPSO	NSGA-II	MOPSO	NSGA-II
j10c5a2	**114.647**	116.272	8.157	**5.866**	**2.908**	1.493	**15.435**	9.941
j10c5a3	**140.991**	141.165	**3.6**	3.828	**1.5084**	1.3961	6.3568	**6.4621**
j10c5a4	**152.812**	153.429	**8.691**	9.6	3.798	**4.584**	17.664	**18.821**
j10c5a5	153.591	**153.264**	4.91	**3.48**	0.832	**0.9909**	**8.345**	5.936
j10c5a6	**140.019**	142.015	6.273	**5.443**	**1.899**	0.8084	**10.925**	8.345
j10c5b1	154.428	**153.29**	4.25	**2.833**	0.511	**1.1081**	**5.656**	4.472
j10c5b2	**122.853**	123.346	**3.075**	4.925	**0.7031**	0.4527	**7.2953**	5.0803
j10c5b3	**131.302**	131.888	**3.365**	3.699	0.3408	**0.8729**	7.1929	**7.584**
j10c5b4	**137.453**	137.831	5.825	**5.614**	**2.1593**	1.897	**7.74661**	5.6568
j10c5b5	115.875	**113.214**	**6.791**	7.67	**4.33**	3.74	**8.109**	7.277
j10c5b6	**146.24**	148.912	**16.602**	16.923	7.537	**8.459**	**11.37**	10.2061
j10c5c1	**139.922**	141.039	**2.944**	4.88	0.8728	**1.079**	**7.56**	4.77
j10c5c2	**151.889**	152.719	6.05	**5.966**	**3.409**	3.320	**13.201**	12.182
j10c5c3	135.733	**134.272**	7.585	**6.0691**	**2.058**	1.6099	10.284	**11.095**
j10c5c4	**110.519**	112.185	**7.477**	7.677	1.901	**2.8232**	12.389	**14.038**
j10c5c5	**110.641**	113.18	**5.623**	9.687	1.7351	**2.6728**	5.063	**14.0459**
j10c5c6	111.747	**110.302**	**2.4**	4.92	**0.5208**	0.4281	3.5510	**6.3253**
j10c5d1	**107.962**	108.134	**5.04**	10.471	**3.657**	2.811	**18.236**	11.403
j10c5d2	**125.16**	127.49	**4.775**	5.21	**0.3813**	0.2746	**7.6321**	7.1727

j10c5d3	**97.951**	99.637	**5.8014**	7.05	**3.2707**	2.5326	**11.1628**	10.25
j10c5d4	**109.844**	113.808	6.375	**6.12**	**3.9137**	3.773	**13.388**	10.0841
j10c5d5	**114.932**	115.26	**9.075**	10.133	**4.7376**	3.996	17.535	**18.629**
j10c5d6	**98.803**	100.285	4.98	**3.475**	1.797	**1.311**	7.102	**5.5461**
j10c10a1	**194.815**	196.419	**9.93**	10.67	**2.7477**	2.012	**17.9175**	13.313
j10c10a2	**222.837**	225.517	16.1	**15.483**	**4.2508**	2.498	**28.801**	24.022
j10c10a3	**204.042**	204.318	14.418	**13.77**	**3.0585**	1.1655	**27.6204**	22.177
j10c10a4	**204.076**	205.065	**3.25**	5.16	**1.0635**	1.027	**6.3488**	5.6356
j10c10a5	204.64	**203.83**	4.903	**4.641**	1.169	**1.382**	6.025	**6.356**
j10c10a6	**197.180**	198.768	**8.556**	9.0714	**6.1863**	4.359	18.56	**19.201**
j10c10b1	**189.903**	191.78	**5.849**	7.025	**1.837**	0.876	**8.534**	6.832
j10c10b2	**232.462**	234.201	**5.29**	6.281	3.414	**6.277**	6.043	**6.862**
j10c10b3	**196.029**	200.572	**6.328**	7.514	**2.8987**	1.8438	**13.2774**	12.7263
j10c10b4	**227.104**	230.546	11.114	**10.27**	**3.712**	2.119	**18.2002**	15.367
j10c10b5	223.239	**221.463**	5.74	**2.982**	**1.0545**	0.761	**5.281**	2.492
j10c10b6	**209.365**	211.607	**4.21**	5.84	**8.137**	6.462	**17.198**	14.852
j10c10c1	**218.419**	220. 810	**5.166**	6.294	**6.0963**	4.752	**3.201**	2.641
j10c10c2	**206.117**	207.456	**8.667**	9.237	**5.507**	4.958	**4.2792**	3.483
j10c10c3	**214.976**	216.192	**4.416**	5.25	**0.497**	0.2817	7.211	**7.971**
j10c10c4	227.167	**226.74**	10.827	**10.229**	**7.463**	6.662	**11.3792**	10.346
j10c10c5	**214.21**	216.96	**8.075**	8.823	**12.37**	10.81	17.236	**18.559**
j10c10c6	210.571	**209.127**	**6.441**	7.156	**10.568**	9.722	**14.279**	13.046
j15c5a1	**210.196**	210.434	7.324	**7.055**	**1.853**	1.5091	**12.1324**	11.7346
j15c5a2	**174.618**	175.532	**9.645**	10.232	**1.427**	1.375	**15.652**	13.213
j15c5a3	**160.604**	161.157	**5.566**	5.991	2.323	**2.9769**	10.440	**10.499**
j15c5a4	154.639	**153.233**	**4.911**	5.107	2.895	**3.719**	**13.892**	12.751
j15c5a5	**196.087**	198.156	**8.506**	6.854	**4.979**	3.194	**14.391**	13.238
j15c5a6	**174.378**	176.028	**6.327**	7.198	3.492	**4.675**	**15.494**	15.187
j15c5b1	**152.846**	153.727	**5.412**	6.183	**4.8214**	4.591	**10.417**	8.7345
j15c5b2	**148.245**	149.773	**4.819**	5.014	**7.419**	6.2471	12.408	**13.619**
j15c5b3	167.924	**166.406**	8.56	**8.173**	**10.371**	9.281	**14.209**	13.443
j15c5b4	**173.707**	174.765	**2.5**	3.412	1.1416	**1.9721**	**13.905**	12.272
j15c5b5	**180.204**	181.075	**6.173**	7.863	**6.702**	5.0192	**11.492**	10.7136
j5c5b6	**152.704**	155.686	**4.346**	5.7865	**3.3795**	2.9887	**8.483**	7.78
j15c5c1	**179.215**	180.626	**2.625**	3.2951	2.519	**3.3977**	4.2426	**5.637**
j15c5c2	**197.906**	199.406	**5.811**	6.2795	**2.494**	1.745	**10.394**	9.287
j15c5c3	**169.941**	171.763	**2.763**	2.8817	**6.9426**	5.523	**14.2258**	12.4927
j15c5c4	**202.873**	203.59	10.317	**9.831**	**1.7016**	1.186	**16.505**	14.372
j15c5c5	**134.127**	134.456	**4.066**	5.25	1.661	**1.822**	8.814	**9.7388**
j15c5c6	157.763	**155.178**	1.916	**1.6512**	**7.9232**	6.746	11.165	**12.3981**
j15c5d1	**157.102**	158.803	**8.2291**	9.318	**3.0253**	2.9707	**14.402**	13.753

j15c5d2	**167.545**	169.323	**3.485**	4.4217	**2.493**	1.0911	**8.4644**	7.529
j15c5d3	**166.342**	166.410	**10.354**	11.071	**6.6824**	5.7765	**17.1251**	16.6873
j15c5d4	**165.223**	167.046	**3.412**	4.5613	**1.4763**	1.1873	**6.1987**	5.9808
j15c5d5	**167.156**	168.252	11.053	**10.752**	**3.9241**	3.169	**17.684**	14.2893
j15c5d6	**163.560**	164.273	**8.452**	6.520	2.667	**3.4315**	**13.513**	11.7124
j15c10a1	308.126	**307.508**	8.782	**8.141**	**3.3631**	3.0013	**14.866**	12.887
j15c10a2	**285.453**	285.981	**5.683**	6.46	1.3375	**1.698**	9.8812	**11.1606**
j15c10a3	**294.108**	295.652	**7.4794**	8.3152	**3.2982**	2.5213	**13.0146**	12.968
j15c10a4	300.731	**298.763**	**6.4564**	6.9345	**4.0748**	3.8743	**15.4038**	13.2571
j15c10a5	**273.554**	275.043	4.5917	**3.3841**	**7.5283**	6.4025	**13.8509**	12.1513
j15c10a6	**308.875**	309.584	**3.9412**	4.5784	8.1547	**9.0521**	**14.7881**	13.6701
j15c10b1	**327.432**	328.570	**6.7731**	7.8019	**3.4365**	2.9918	**11.6097**	10.0839
j15c10b2	321.608	**320.931**	**10.1953**	11.728	**7.4607**	6.9913	16.5853	**17.3217**
j15c10b3	**348.036**	350.458	**3.0341**	4.3672	**6.297**	5.075	**12.510**	10.2119
j15c10b4	**334.412**	334.895	7.5456	**6.2897**	**5.4789**	4.5287	**10.1882**	9.1482
j15c10b5	354.318	**353.425**	**6.2674**	6.9049	9.2581	**10.032**	16.1239	**17.5748**
j15c10b6	**318.678**	320.679	**2.1237**	2.9823	6.4778	**7.626**	**10.0835**	9.6787

* The best obtained values are marked in bold letter

Table 6.3 Performance metrics of Pareto front obtained by the objective of makespan and mean tardiness

Problem	MID		RAS		SNS		Diversity (DM)	
	MOPSO	NSGA-II	MOPSO	NSGA-II	MOPSO	NSGA-II	MOPSO	NSGA-II
j10c5a2	**95.6504**	95.876	7.773	**6.543**	**2.246**	1.77	**13.254**	12.4192
j10c5a3	123.759	**123.432**	5.33	**4.8912**	**2.4092**	2.1185	**9.0494**	8.07718
j10c5a4	**129.22**	132.584	**7.76**	10.197	4.13	**7.7696**	12.247	**18.144**
j10c5a5	**152.306**	152.492	**4.825**	4.94	**1.6694**	1.146	**8.66**	7.5604
j10c5a6	**119.882**	120.186	**4.1008**	4.237	**1.932**	1.819	**6.821**	6.167
j10c5b1	**138.745**	141.42	**12.861**	14.11	**1.298**	0.8744	**6.2154**	4.7845
j10c5b2	**146.23**	148.67	**2.55**	2.645	**0.4262**	0.3955	3.976	**4.242**
j10c5b3	132.473	**131.638**	6.4875	**4.2375**	**2.6122**	1.222	**9.13**	6.655
j10c5b4	**127.055**	129.597	11.212	**6.201**	**4.902**	3.1372	**16.697**	13.934
j10c5b5	109.027	**105.387**	**6.607**	9.44	**3.073**	1.89	**10.174**	6.306
j10c5b6	**122.15**	126.204	**12.875**	13.623	**10.7047**	8.917	**12.461**	9.293
j10c5c1	**118.491**	119.905	**4.167**	4.5067	**1.9103**	1.819	**6.27408**	6.0325
j10c5c2	130.68	**130.14**	**10.797**	11.155	**2.6064**	1.643	13.907	**15.181**
j10c5c3	**110.463**	110.615	5.552	**4.857**	2.9035	**4.013**	**9.7779**	8.94
j10c5c4	105.28	**104.504**	9.0754	**7.23**	**4.025**	3.239	**15.764**	12.5481
j10c5c5	93.6161	**92.051**	11.321	**8.785**	**7.476**	6.029	**19.776**	15.925
j10c5c6	**90.611**	93.55	**7.95**	8.75	**2.4108**	1.668	**22.61**	15.297

j10c5d1	**108.134**	111.178	**5.04**	10.1	**5.6047**	3.8112	**17.075**	12.44
j10c5d2	**91.281**	93.547	10.65	**9.857**	**3.26**	2.514	**20.462**	17.709
j10c5d3	**80.310**	84.8535	**7.435**	10.115	4.301	**6.591**	**16.18**	12.967
j10c5d4	93.294	**91.241**	**9.358**	11.632	**5.619**	5.4102	**17.783**	16.8885
j10c5d5	**85.8672**	86.412	**7.494**	7.632	2.9102	**3.195**	**11.9315**	11.005
j10c5d6	79.823	**79.7817**	**4.809**	5.5366	**4.0455**	2.2713	8.0435	**8.7988**
j10c10a1	162.427	**161.705**	8.885	**7.3625**	**5.584**	2.795	**17.214**	11.205
j10c10a2	175.181	**175.058**	**16.0533**	16.088	**3.805**	3.567	**30.042**	26.9436
j10c10a3	**163.306**	165.468	**10.165**	14.486	**6.1077**	4.1017	**23.6008**	15.986
j10c10a4	**165.763**	166.21	**9.656**	11.75	**8.89**	8.44	17.541	**17.608**
j10c10a5	162.38	**161.685**	**5.733**	6.329	3.931	**5.329**	**12.804**	12.215
j10c10a6	**160.2382**	161.252	**11.5**	12.829	**5.0976**	4.1982	**18.968**	17.0293
j10c10b1	**197.6307**	201.28	**6.122**	8.375	**5.448**	2.5135	**12.804**	11.346
j10c10b2	**189.903**	194.436	**5.849**	7.8493	**4.053**	1.8375	8.5346	**12.048**
j10c10b3	160.611	**160.0244**	**7.215**	7.865	**5.8705**	3.8458	11.3265	**15.126**
j10c10b4	**184.47**	187.235	9.4375	**8.25**	**3.543**	3.2822	**13.268**	11.40175
j10c10b5	172.679	**170.552**	12.378	**12.089**	5.173	**6.627**	11.61	**12.49**
j10c10b6	**165.459**	168.406	**13.243**	13.845	**8.281**	6.554	**12.723**	10.151
j10c10c1	**171. 481**	171. 785	**10.275**	11.452	**6.814**	7.658	16.0262	**17.525**
j10c10c2	**162.349**	163.0547	**7.104**	7.843	4.732	**5.624**	**8.512**	7.231
j10c10c3	**175.617**	177.821	**12.557**	13.142	**6.157**	5.608	**19.893**	18.456
j10c10c4	**189.532**	190.757	14.965	**14.0245**	**7.854**	6.527	**13.145**	12.987
j10c10c5	**163.845**	165.377	**12.884**	13.743	**3.558**	3.006	**9.279**	7.663
j10c10c6	175.617	**174.214**	**12.557**	13.142	**6.157**	5.289	**19.893**	17.651
j15c5a1	**186.751**	186.817	8.160	**7.79125**	2.5345	**2.894**	13.770	**14.015**
j15c5a2	**194.761**	195.561	4.02	**4.64**	**1.729**	1.478	**6.264**	5.621
j15c5a3	**141.415**	141.5705	**4.2833**	4.375	1.4041	**1.465**	**6.931**	6.594
j15c5a4	**139.661**	140.204	9.496	**8.345**	**4.767**	2.188	**9.424**	7.257
j15c5a5	**177.057**	177.6754	10.425	**9.93**	**6.1942**	7.299	20.69	**19.506**
j15c5a6	**135.704**	137.0417	**9.054**	10.545	**2.229**	1.417	**11.7663**	9.9102
j15c5b1	**125.942**	126.5805	**8.6349**	9.8445	**1.1614**	1.053	**12.1907**	11.05982
j15c5b2	**119.6675**	121.4949	**3.5847**	4.863	**5.863**	4.1632	**14.6752**	12.7748
j15c5b3	**132.9352**	133.6216	**2.0476**	3.339	**8.613**	7.6752	**13.8964**	12.4785
j15c5b4	139.3355	**138.141**	**4.718**	5.051	**6.7623**	5.1513	**9.538**	9.2775
j15c5b5	**128.876**	130.464	**3.1062**	4.433	**7.246**	6.0636	**10.927**	9.5606
j5c5b6	**115.6059**	117.3896	**5.0904**	5.4216	11.7082	**12.2141**	**15.5896**	14.5478
j15c5c1	**157.656**	161.352	**5.02**	5.82	1.574	**1.2895**	8.360	**8.993**
j15c5c2	**175.370**	176.0719	12.501	**10.76**	3.594	**3.5519**	**14.732**	13.0941
j15c5c3	**164.5367**	166.547	**5.751**	6.2126	8.6397	**8.0205**	**11.3176**	10.4492
j15c5c4	**178.912**	179.661	**7.582**	9.1	4.085	**4.228**	**15.439**	13.892
j15c5c5	117.782	**117.618**	**6.386**	8.455	**2.9870**	6.057	**13.025**	10.77

j15c5c6	**135.1804**	137.283	**3.4921**	4.285	6.6273	**6.7559**	**11.9508**	10.768
j15c5d1	**134.520**	134.771	10.12	**9.897**	4.2032	**5.048**	**15.9452**	14.8112
j15c5d2	**148.212**	151.163	14.086	**13.35**	7.876	**8.329**	**26.963**	24.144
j15c5d3	**140.6320**	141.743	**9.4553**	11.303	**5.6958**	4.362	**15.5769**	15.02156
j15c5d4	**140.0923**	140.7862	**6.1756**	7.4103	**1.7791**	1.0245	**8.8079**	7.4596
j15c5d5	**142.2092**	143.847	**12.4338**	13.6152	**2.7419**	2.2357	**18.1046**	16.3782
j15c5d6	**138.355**	140.0734	**8.2917**	8.8364	**9.1075**	10.546	**12.331**	10.894
j15c10a1	**264.0627**	264.4576	10.66111	**10.4762**	**3.173**	2.9276	**19.5364**	19.347
j15c10a2	**236.1089**	236.6064	**10.42**	13.6032	2.1626	**2.317**	**16.745**	15.291
j15c10a3	**247.2432**	248.827	**7.1507**	7.3962	**4.5736**	3.1914	**10.175**	8.4179
j15c10a4	257.724	**256.0263**	2.7039	**1.3126**	**7.389**	6.188	**15.9857**	14.7902
j15c10a5	**239.1892**	241.2065	5.8772	**4.6531**	2.2587	**3.6599**	**11.6807**	10.4051
j15c10a6	261.4218	**260.2784**	7.3204	**6.2154**	**7.5947**	6.1203	**17.6245**	16.4887
j15c10b1	288.5172	**286.7642**	**3.741**	4.574	6.2254	**7.827**	**10.2085**	8.7214
j15c10b2	**302.1359**	305.8405	**4.6092**	5.8942	**8.7087**	7.1773	**13.3719**	12.1065
j15c10b3	**319.2841**	320.036	6.3714	**4.4620**	**3.184**	2.2907	**9.412**	8.704
j15c10b4	**305.6367**	307.2287	7.917	**6.2867**	**4.5597**	3.4283	**8.1987**	6.9154
j15c10b5	**315.78**	316.2417	**2.0965**	3.1324	**7.147**	6.6254	**10.1974**	9.3012
j15c10b6	**297.972**	299.4037	3.5752	**2.163**	**4.5087**	3.8179	12.435	**13.5387**

* The best obtained result is marked in bold letter

In the present investigation, application of MOPSO results in large number of non-dominated solutions for optimization of objectives. The Pareto-optimal solutions obtained through MOPSO have been ranked by the composite scores obtained through maximum deviation theory (MDT) to choose the best solution. The decision matrix is normalized using the equations 6.11 and equation 6.12. The objective weights are determined for the normalized values of objectives by applying maximum deviation method using equation 6.13-6.20. The weighted objective values are estimated by multiplying the normalized objective values and the objective weights. The best solution is selected depending upon the composite scores obtained by addition of the all the weighted objective function values for each alternative. The objectives with highest composite score are chosen as the best solution. The solution ranking of the optimal solution set of problem j10c10a3 for makespan and mean tardiness has been given in Table 6.4.

Table 6.4 Solution ranking obtained through maximum deviation theory for the problem j10c10a3

Run order	Objective values	function	Normalized objective values	function	Weighted objective values	function	Composite Score	Solution ranking
	Makespan C_{max}	mean tardiness $(\overline{T})$	N_{cmax}	$N_{\overline{T}}$	WN_{cmax}	$WN_{\overline{T}}$		
1	156	20	1.0000	0	0.5206	0	0.5206	10
2	157	18.25	0.9474	0.1250	0.4932	0.0599	0.5531	7
3	158	15.74	0.8947	0.3043	0.4658	0.1459	0.6117	3
4	160	13.75	0.7895	0.4464	0.4110	0.2140	0.6250	2
5	**162**	**11.9**	**0.6842**	**0.5786**	**0.3562**	**0.2774**	**0.6336**	**1**
6	165	10.24	0.5263	0.6971	0.2740	0.3342	0.6082	4
7	166	9.67	0.4737	0.7379	0.2466	0.3537	0.6003	5
8	167	9.15	0.4211	0.7750	0.2192	0.3715	0.5907	6
9	170	8.21	0.2632	0.8421	0.1370	0.4037	0.5407	8
10	171	7.73	0.2105	0.8764	0.1096	0.4202	0.5298	9
11	173	7.2	0.1053	0.9143	0.0548	0.4383	0.4931	11
12	175	6	0	1.0000	0	0.4794	0.4794	12

* The best obtained result is marked in bold letter

Table 6.5 represents the best makespan value obtained by maximum deviation theory based on the of highest composite score and the percentage deviation (%PD) of the makespan from the lower bound (LB) [108]. Percentage Deviation (%PD) is defined in the equation 3.18.

Table 6.5 The computational results of maximum deviation theory

Problem	LB	Objectives: Makespan and mean flow time				Objectives: Makespan and mean tardiness			
		MOPSO		NSGA-II		MOPSO		NSGA-II	
		best C_{max} by MDT	%PD	best C_{max} by MDT	%PD	best C_{max} by MDT	%PD	best C_{max} by MDT	%PD
j10c5a2	88	92	4.545	94	6.818	93	5.682	94	6.818
j10c5a3	117	119	1.709	120	2.564	119	1.709	119	1.709
j10c5a4	121	125	3.306	129	6.612	128	5.785	129	6.612
j10c5a5	122	127	4.098	127	4.098	128	4.918	127	4.098
j10c5a6	110	116	5.455	117	6.364	115	4.545	117	6.364
j10c5b1	130	142	9.231	143	10.000	142	9.231	142	9.231
j10c5b2	107	117	9.346	119	11.215	117	9.346	118	10.280
j10c5b3	109	125	14.679	127	16.514	123	12.844	126	15.596
j10c5b4	122	130	6.557	134	9.836	130	6.557	133	9.016
j10c5b5	153	173	13.072	176	15.033	170	11.111	176	15.033
j10c5b6	115	134	16.522	137	19.130	132	14.783	135	17.391

j10c5c1	68	119	75.000	120	76.471	119	75.000	121	77.941
j10c5c2	74	126	70.270	126	70.270	124	67.568	129	74.324
j10c5c3	71	107	50.704	108	52.113	103	45.070	107	50.704
j10c5c4	66	90	36.364	92	39.394	98	48.485	103	56.061
j10c5c5	78	89	14.103	93	19.231	84	7.692	90	15.385
j10c5c6	69	88	27.536	90	30.435	89	28.986	92	33.333
j10c5d1	66	86	30.303	89	34.848	84	27.273	87	31.818
j10c5d2	73	100	36.986	98	34.247	95	30.137	99	35.616
j10c5d3	64	77	20.313	80	25.000	75	17.188	80	25.000
j10c5d4	70	87	24.286	94	34.286	87	24.286	89	27.143
j10c5d5	66	89	34.848	95	43.939	86	30.303	90	36.364
j10c5d6	62	78	25.806	82	32.258	75	20.968	81	30.645
j10c10a1	139	152	9.353	155	11.511	157	12.950	159	14.388
j10c10a2	158	180	13.924	178	12.658	171	8.228	171	8.228
j10c10a3	148	163	10.135	164	10.811	158	6.757	162	9.459
j10c10a4	149	160	7.383	166	11.409	160	7.383	164	10.067
j10c10a5	148	162	9.459	162	9.459	158	6.757	161	8.784
j10c10a6	146	158	8.219	160	9.589	156	6.849	155	6.164
j10c10b1	163	199	22.086	207	26.994	196	20.245	201	23.313
j10c10b2	157	186	18.471	190	21.019	179	14.013	186	18.471
j10c10b3	169	172	1.775	179	5.917	175	3.550	181	7.101
j10c10b4	159	176	10.692	181	13.836	176	10.692	179	12.579
j10c10b5	165	180	9.091	184	11.515	178	7.879	183	10.909
j10c10b6	165	177	7.273	181	9.697	177	7.273	178	7.879
j10c10c1	113	154	36.283	173	53.097	154	36.283	170	50.442
j10c10c2	116	161	38.793	167	43.966	158	36.207	164	41.379
j10c10c3	98	129	31.633	133	35.714	126	28.571	130	32.653
j10c10c4	103	146	41.748	146	41.748	141	36.893	145	40.777
j10c10c5	121	168	38.843	165	36.364	163	34.711	165	36.364
j10c10c6	97	142	46.392	148	52.577	140	44.330	146	50.515
j15c5a1	178	183	2.809	187	5.056	183	2.809	184	3.371
j15c5a2	165	170	3.030	174	5.455	169	2.424	173	4.848
j15c5a3	130	136	4.615	139	6.923	136	4.615	137	5.385
j15c5a4	156	171	9.615	175	12.179	167	7.051	170	8.974
j15c5a5	164	182	10.976	181	10.366	180	9.756	180	9.756
j15c5a6	178	195	9.551	198	11.236	192	7.865	194	8.989
j15c5b1	170	191	12.353	193	13.529	188	10.588	192	12.941
j15c5b2	152	177	16.447	179	17.763	174	14.474	179	17.763
j15c5b3	157	172	9.554	173	10.191	168	7.006	172	9.554
j15c5b4	147	168	14.286	174	18.367	169	14.966	173	17.687
j15c5b5	166	192	15.663	195	17.470	190	14.458	194	16.867

J15c5b6	175	203	16.000	204	16.571	200	14.286	203	16.000
j15c5c1	85	145	70.588	151	77.647	142	67.059	148	74.118
j15c5c2	90	183	103.333	187	107.778	180	100.000	183	103.333
j15c5c3	87	170	95.402	173	98.851	169	94.253	172	97.701
j15c5c4	89	168	88.764	173	94.382	168	88.764	173	94.382
j15c5c5	73	111	52.055	114	56.164	113	54.795	113	54.795
j15c5c6	91	152	67.033	155	70.330	157	72.527	158	73.626
j15c5d1	167	205	22.754	209	25.150	204	22.156	209	25.150
j15c5d2	82	139	69.512	142	73.171	137	67.073	140	70.732
j15c5d3	77	137	77.922	138	79.221	134	74.026	137	77.922
j15c5d4	61	122	100.000	126	106.557	125	104.918	126	106.557
j15c5d5	67	131	95.522	134	100.000	129	92.537	132	97.015
j15c5d6	79	147	86.076	147	86.076	142	79.747	147	86.076
j15c10a1	236	258	9.322	262	11.017	259	9.746	260	10.169
j15c10a2	200	230	15.000	232	16.000	232	16.000	234	17.000
j15c10a3	198	245	23.737	247	24.747	244	23.232	248	25.253
j15c10a4	225	264	17.333	269	19.556	265	17.778	267	18.667
j15c10a5	182	227	24.725	232	27.473	228	25.275	230	26.374
j15c10a6	200	250	25.000	254	27.000	250	25.000	256	28.000
j15c10b1	222	271	22.072	273	22.973	268	20.721	272	22.523
j15c10b2	187	238	27.273	241	28.877	239	27.807	241	28.877
j15c10b3	222	254	14.414	259	16.667	254	14.414	257	15.766
j15c10b4	221	282	27.602	286	29.412	284	28.507	287	29.864
j15c10b5	200	242	21.000	246	23.000	245	22.500	245	22.500
j15c10b6	219	275	25.571	278	26.941	277	26.484	278	26.941
Average Percentage Deviation			28.617		31.33		27.593		30.538

It is observed from the Table 6.5 that most of the best makespan values obtained by MDT from the non-dominated solutions of MOPSO have smaller %PD than the NSGA-II. Average percentage deviation (APD) of makespan by the proposed MOPSO and NSGA-II are 28.617 and 31.33 respectively for objectives of makespan and mean flow time. Similarly, the average percentage deviation (APD) of makespan by the proposed MOPSO and NSGA-II are 27.593 and 30.538 respectively for objectives of makespan and mean tardiness.

The improvement rate of APD using MOPSO is defined as follows:

$$\text{Improvement rate } (\%) = \frac{(\text{APD}_{\text{NSGA-II}} - \text{APD}_{\text{MOPSO}})}{\text{APD}_{\text{NSGA-II}}} \quad (6.25)$$

The improvement rate of APD is found to be 8.66 % for the objectives makespan and mean flow time and 9.62 % for the objectives makespan and mean tardiness.

6.4.2 Result analysis for multi-objective FJSP

In this section, proposed multi-objective particle swarm optimization (MOPSO) is used for solving the flexible job shop scheduling problem (FJSP). Based on thorough analysis, Figure 6.10-6.17 are drawn to show the Pareto front between makespan and mean flow time and makespan and mean tardiness for four benchmark instances of FJSP. The efficiency of the proposed MOPSO algorithm is compared with another popular multi-objective algorithm known as non-dominated sorting genetic algorithm II (NSGA-II).Based on thorough testing, Figure 6.10 to 6.17 are drawn to show the Pareto front between makespan and mean flow time and makespan and mean tardiness for eight benchmark instances of FJSP.

The MOPSO is tested by solving twenty eight different benchmark problems of Brandimarte [112] and Dauzere-peres data set [115]. The results obtained by the proposed algorithms are compared in terms of the performance metrics with the NSGA-II. Table 6.6 and Table 6.7 illustrate the comparative results of two algorithms with respect to four performance measures. From Table 6.6 and Table 6.7, it can be concluded that MOPSO outweighs the NSGA-II algorithms in all metrics in terms of the number of optimum results out of twenty eight test problems. It is observed from that Table 6.6 that the proposed MOPSO superior to the NSGA-II in 21, 18, 20 and 20 out of 28 test problems with respect to MID, RAS, SNS and DM performance measures respectively for the objectives of makespan and mean flow time. Table 6.7 indicates that MOPSO performs superior to NSGA-II in 23, 20, 19 and 22 out of 28 test problems in MID, RAS, SNS and DM performance measures respectively for the objective of makespan and mean tardiness.

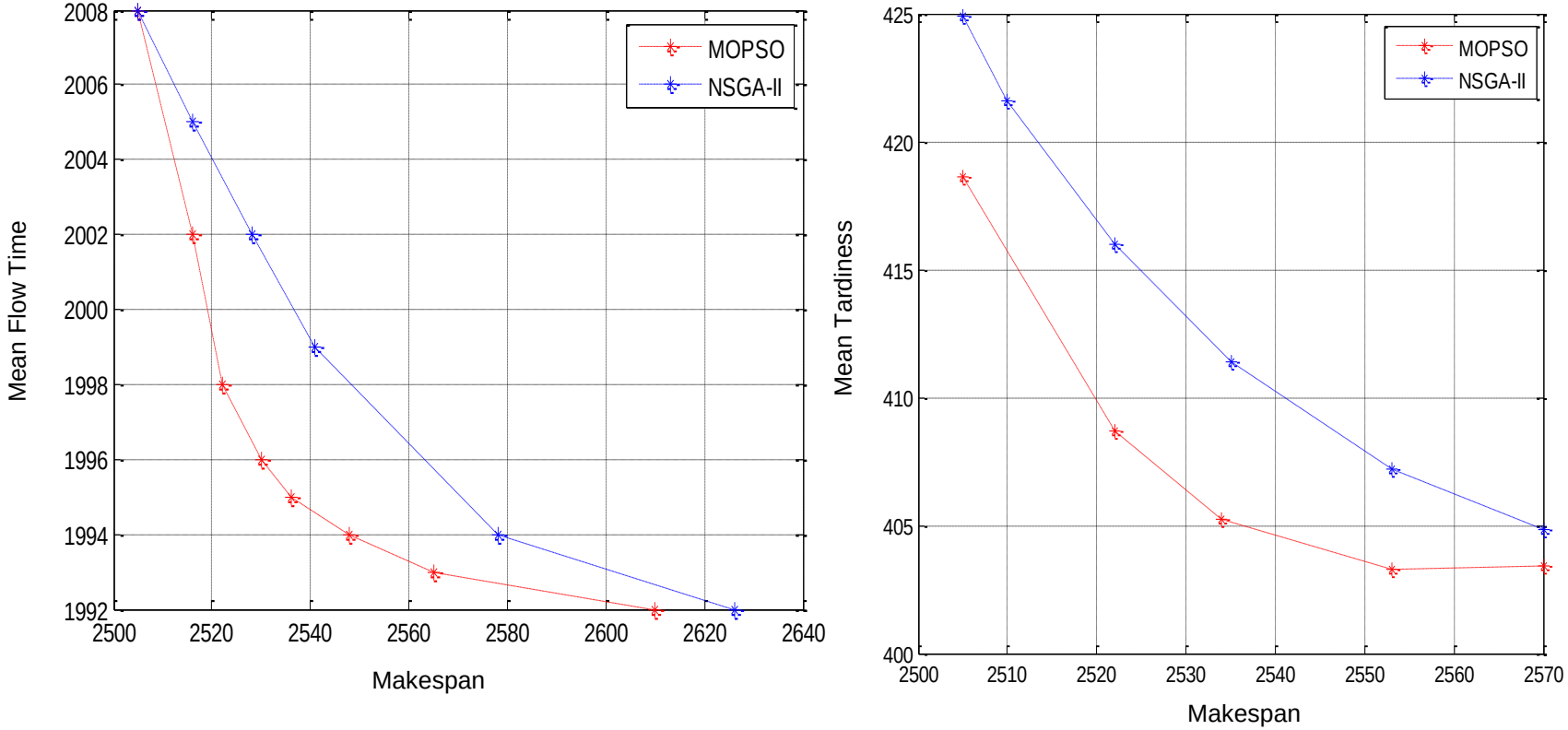


Figure 6.10 Pareto front obtained by the proposed MOPSO and NSGA-II for the instance 1a

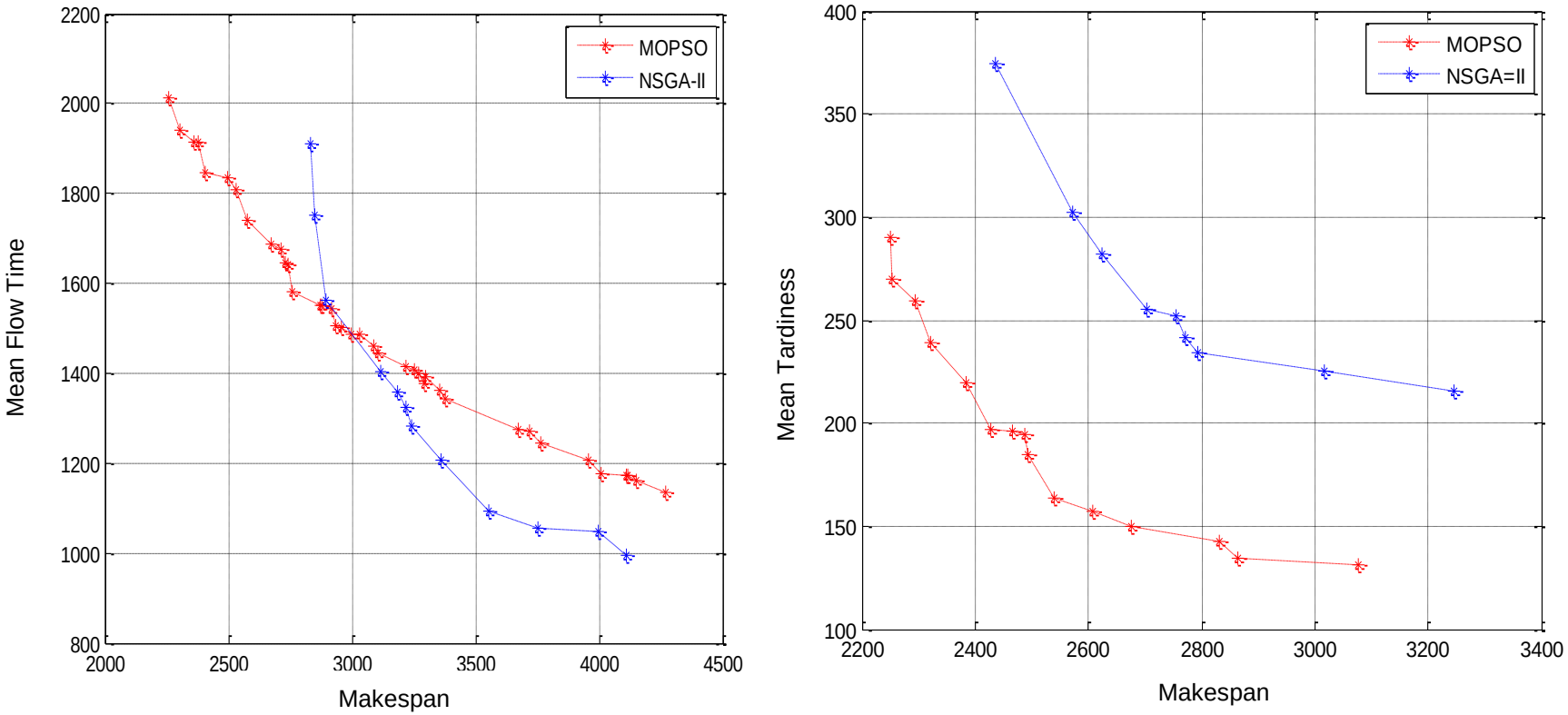


Figure 6.11 Pareto front obtained by the proposed MOPSO and NSGA-II for the instance 3a

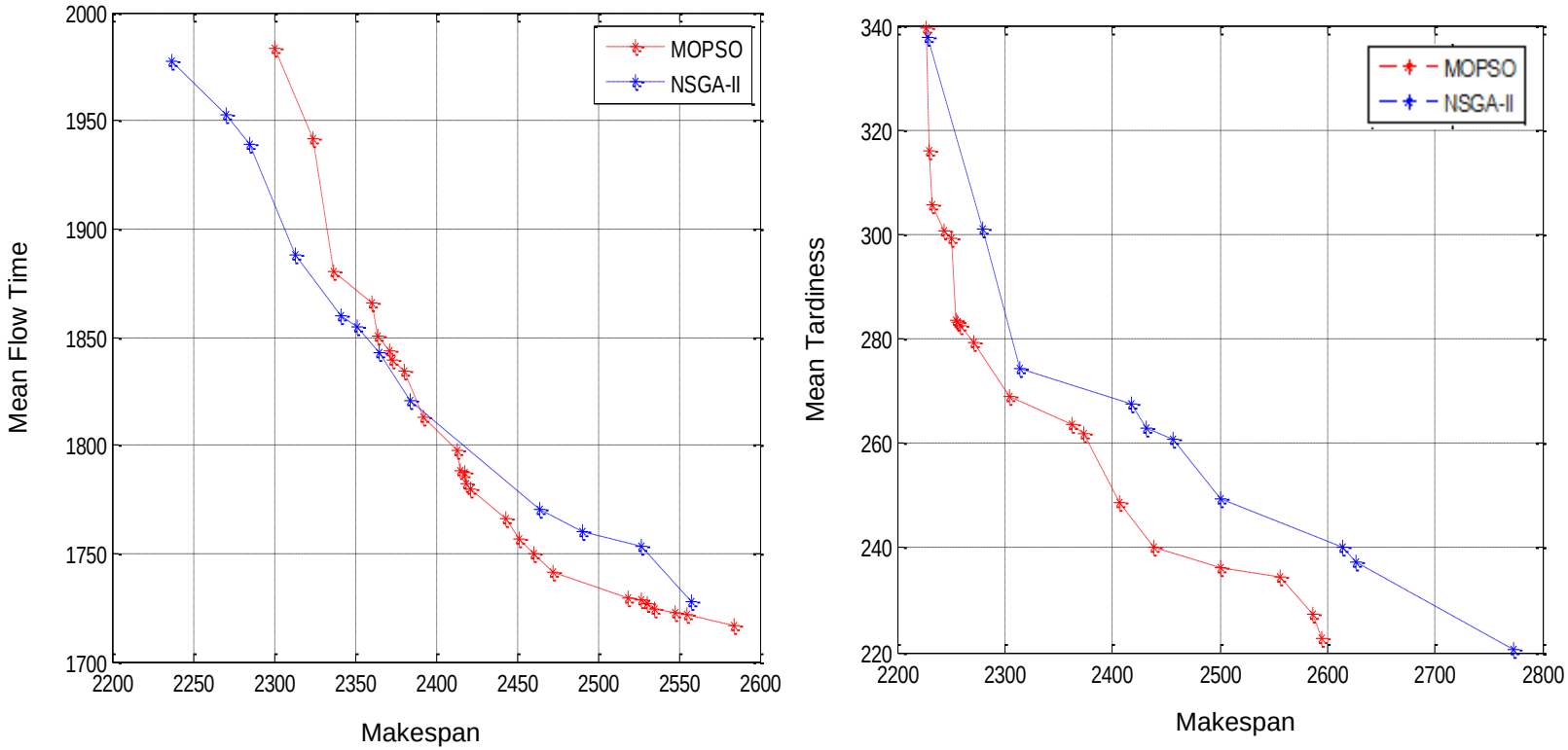


Figure 6.12 Pareto front obtained by the proposed MOPSO and NSGA-II for the instance 5a

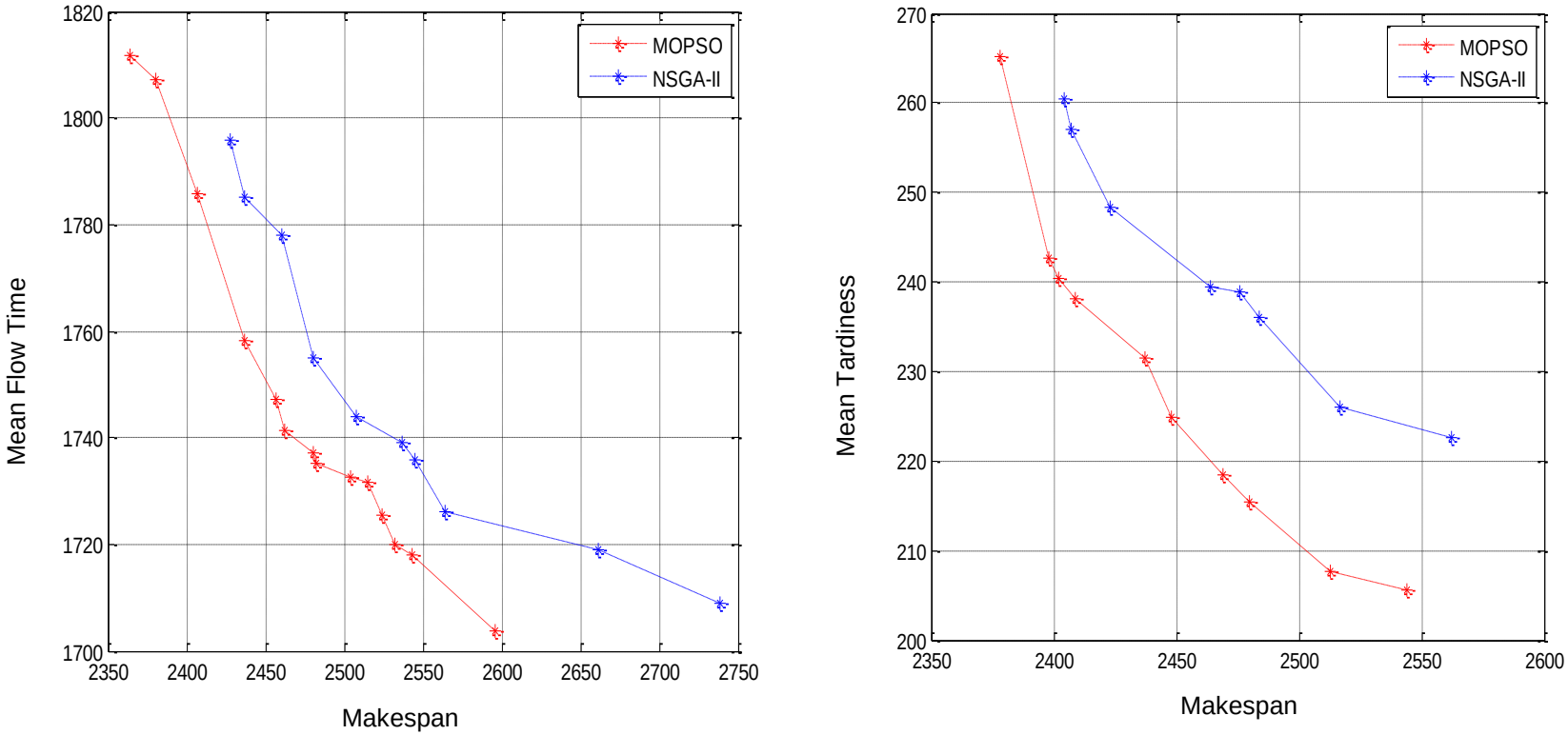


Figure 6.13 Pareto front obtained by the proposed MOPSO and NSGA-II for the instance 10a

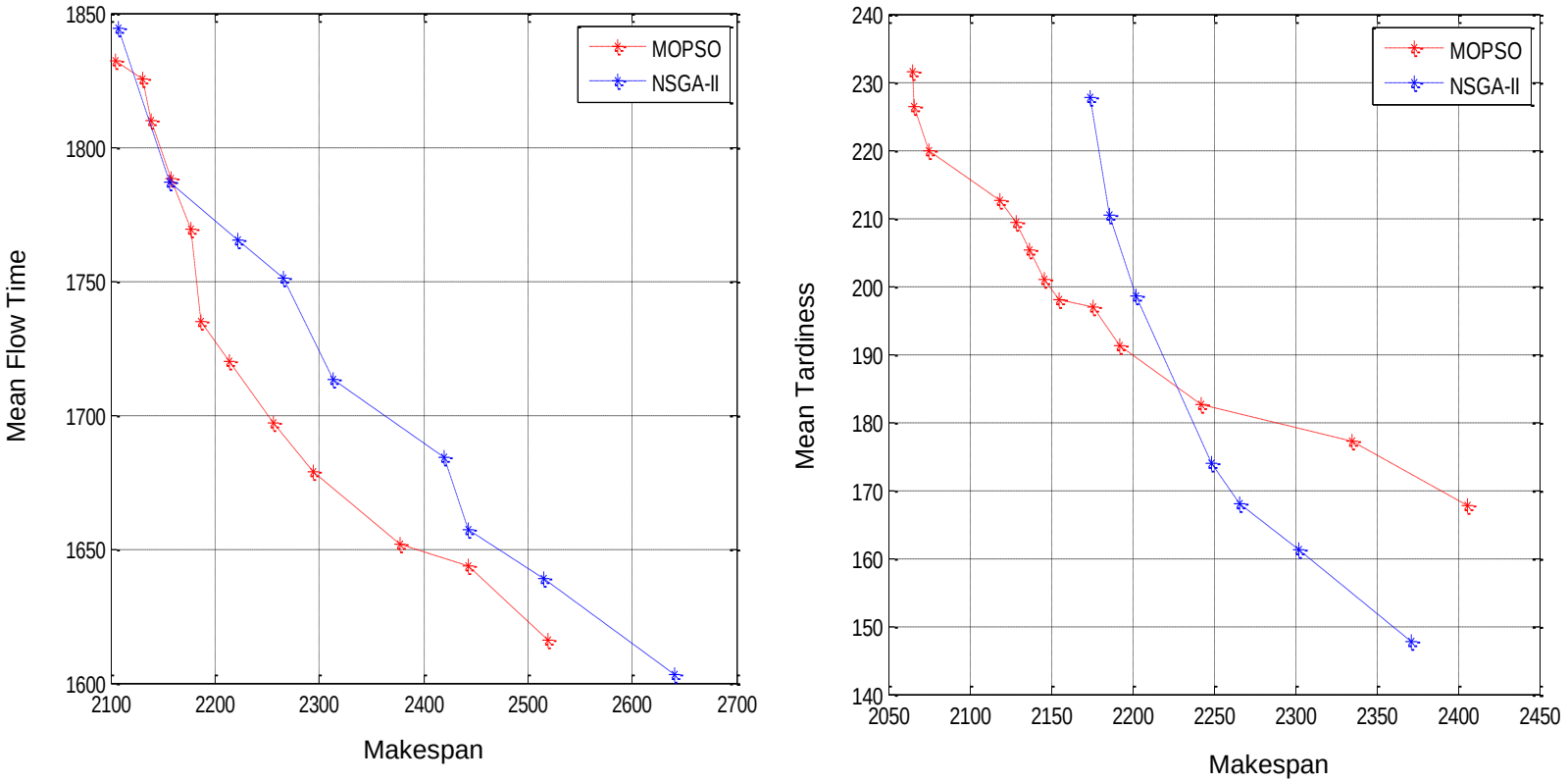


Figure 6.14 Pareto front obtained by the proposed MOPSO and NSGA-II for the instance 11a

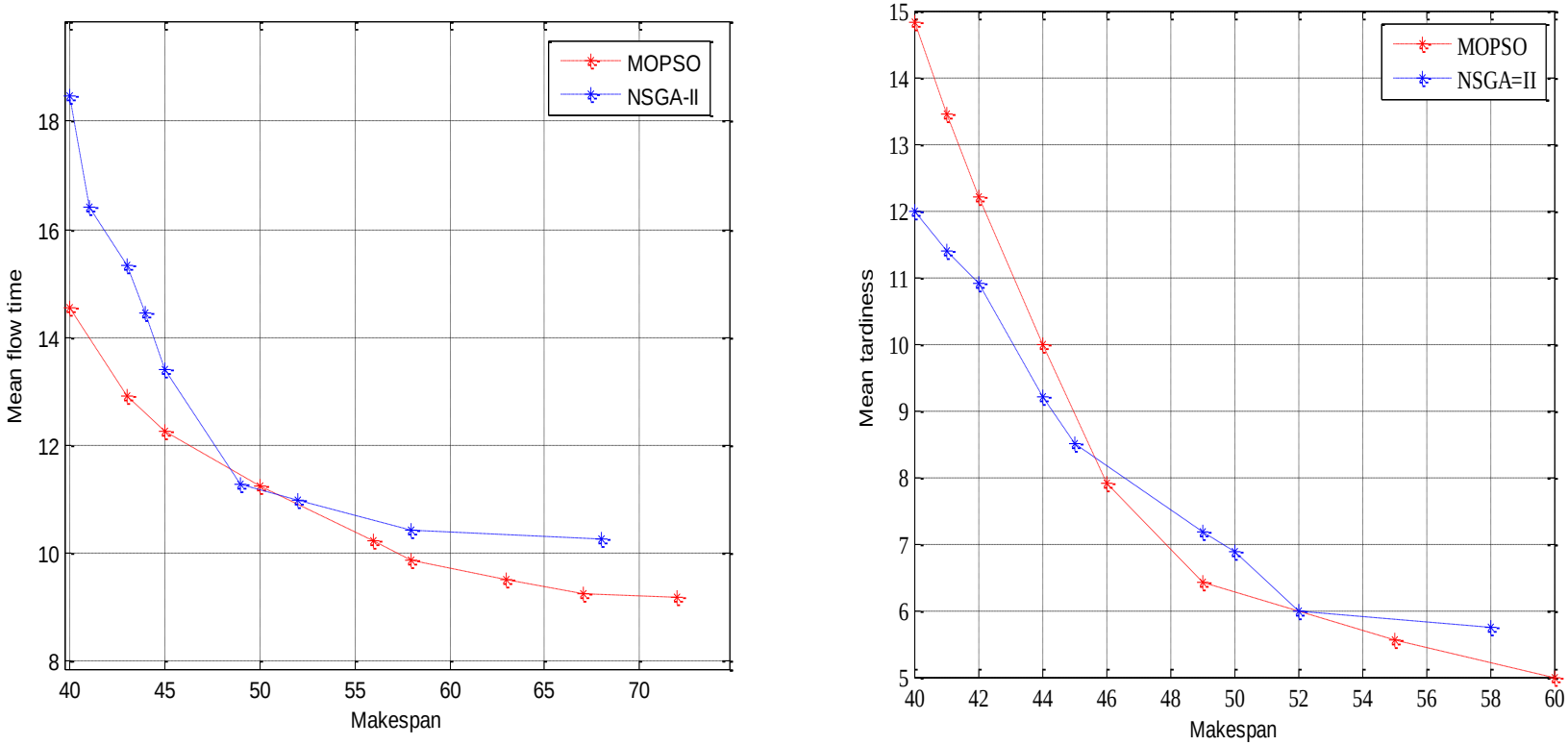


Figure 6.15 Pareto front obtained by the proposed MOPSO and NSGA-II for the instance Mk 01

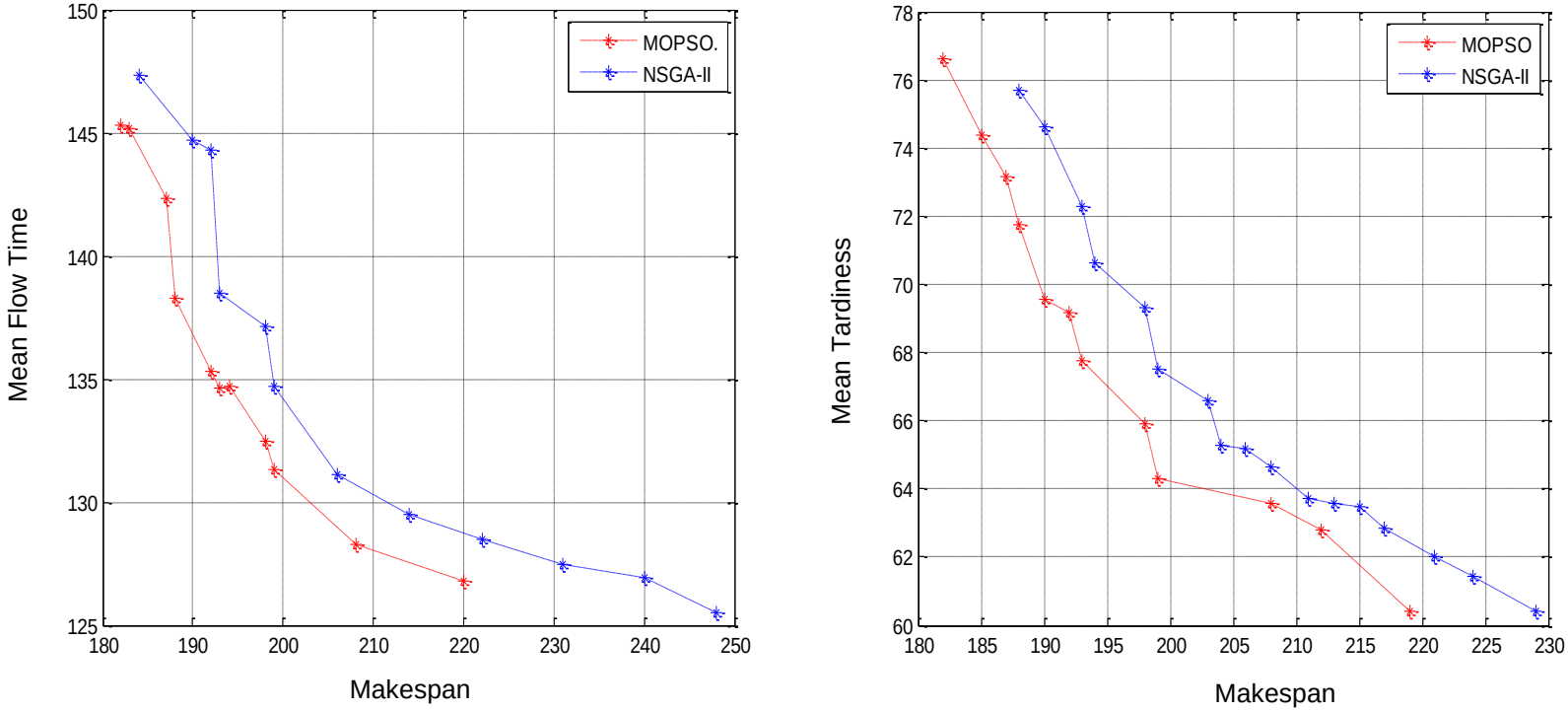
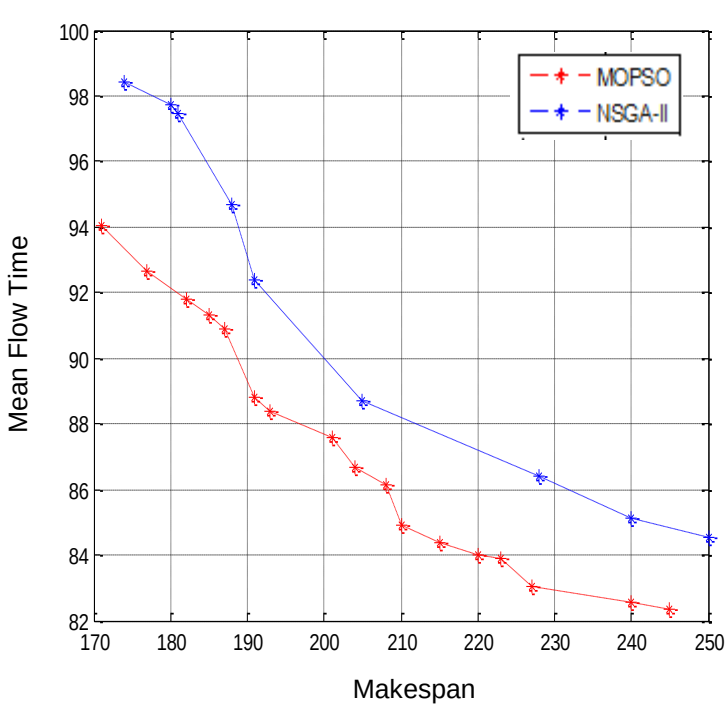


Figure 6.16 Pareto front obtained by the proposed MOPSO and NSGA-II for the instance Mk 05



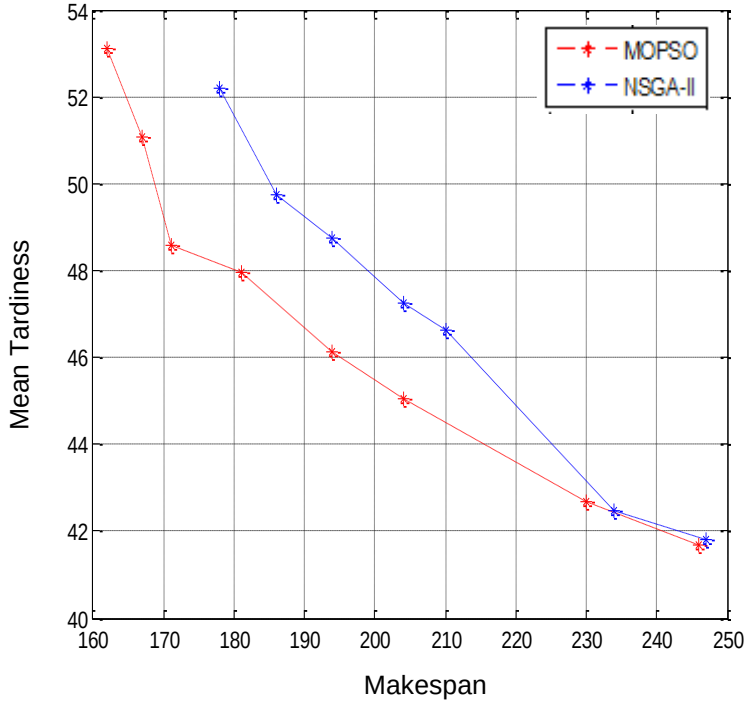


Figure 6.17 Pareto front obtained by the proposed MOPSO and NSGA-II for the instance Mk 07

Table 6.6 Performance metrics of Pareto front obtained by the objective of makespan and mean flow time

Problem	n x m	MID		RAS		SNS		Diversity (DM)	
		MOPSO	NSGA-II	MOPSO	NSGA-II	MOPSO	NSGA-II	MOPSO	NSGA-II
Mk 01	10 x 6	56.119	**50.93**	**10.71**	12.137	**10.57**	8.269	**32.44**	29.17
Mk 02	10 x 6	**43.805**	46.625	8.977	**8.471**	**3.926**	3.688	**17.88**	17.1918
Mk 03	15 x 8	**257.273**	288.285	**40.51**	62.709	19.381	**25.324**	87.531	**108.847**
Mk 04	15 x 8	104.0892	**103.9267**	22.0708	**21.352**	10.0842	**13.052**	37.824	**44.8517**
Mk 05	15 x 4	**236.83**	249.59	**18.081**	35.55	**16.78**	14.53	30.27	**67.61**
Mk 06	10x15	**151.58**	155.12	**15.21**	19.37	**2.266**	1.46	**23.34**	21.34
Mk 07	20 x 5	**222.77**	229.75	**38.49**	43.45	**18.605**	13.37	**74.92**	67.73
Mk 08	20x10	**644.825**	645.094	36.101	**35.988**	**6.9437**	5.5701	**49.8**	48.2801
Mk 09	20x10	**752.87**	755.23	**66.75**	66.87	**23.64**	22.78	36.67	**37.12**
Mk 10	20x15	1677.72	**1672.1**	**347.56**	354.3	**122.45**	122.1	**267.69**	264.22
1a	10x5	**3214.13**	3244.062	85.71	**57**	**31.45**	31.001	**129.76**	122.054
2a	10x5	**3097.135**	3157.46	371.75	**266.908**	**90.293**	58.711	**658.67**	428.78
3a	10x5	**3494.793**	3620.75	**836.5**	897.275	**419.971**	308.3	**2194.8**	1571.38
4a	10x5	3141.27	**3095.47**	207.99	**136.56**	**79.81**	48.47	**412.36**	225.51
5a	10x5	3027.77	**3015.9**	**214.904**	262.67	30.563	**36.16**	389.8	**406.253**
6a	10x5	**2951.502**	3041.99	**372.48**	429.75	**185.605**	**141.98**	626.06	**825.412**
7a	15x8	**3023.466**	3038.891	139.01	**87.5**	**29.588**	22.894	**220.99**	175.513
8a	15x8	**3057.918**	3134.15	**471.034**	573.98	**265.033**	215.541	**1362**	1047.23
9a	15x8	**3061.157**	3176.37	**469.39**	492.52	**202.92**	201.47	**1068.2**	967.12
10a	15x8	**3031.89**	3081.275	**156.47**	167.6	35.25	**67.8**	**355.9**	322.939
11a	15x8	**2841.59**	2908.101	359.58	**347.66**	66.8	**97.25**	**637.96**	584.041
12a	15x8	**3098.43**	3154.67	**345.21**	346.79	172.34	**175.22**	**867.79**	867.45
13a	20x10	**3329.32**	3364.32	**262.27**	311.29	**74.311**	57.81	**385.31**	308.62
14a	20x10	**3619.43**	3846.77	**525.37**	556.65	**176.97**	154.88	**924.13**	894.68
15a	20x10	3087.86	**3197.5**	**367.56**	413.46	286.48	267.72	346.76	**335.41**
16a	20x10	**3438.78**	3475.25	**481.38**	490.11	**312.98**	284.77	**189.4**	175.5
17a	20x10	**3310.2**	3346.841	679.53	**678.87**	148.83	**164.7**	415.62	**417.23**
18a	20x10	3864.58	**3749.15**	784.29	**776.71**	**229.46**	227.69	**523.84**	521.92

* The best obtained values are marked in bold letter

Table 6.7 Performance metrics results of Pareto front obtained by the objective of makespan and mean tardiness

Problem	n x m	MID		RAS		SNS		Diversity (DM)	
		MOPSO	NSGA-II	MOPSO	NSGA-II	MOPSO	NSGA-II	MOPSO	NSGA-II
Mk 01	10 x 6	**48.26**	49.6	11.64	**10.37**	**7.42**	7.4063	**27.99**	26.91
Mk 02	10 x 6	**34.1**	43.59	**5.15**	8.741	82.55	**84.37**	**28.65**	21.013
Mk 03	15 x 8	**234.9**	259.68	**32.0726**	37.66	**16.486**	15.52	52.0522	**59.55**
Mk 04	15 x 8	**85.46**	108.75	**11.79**	16.6	**11.62**	6.17	**27.982**	12.6
Mk 05	15 x 4	**207.8**	217.201	**21.96**	24.64	9.406	**10.23**	**44.391**	43.75
Mk 06	10x15	**128.96**	131.97	6.0175	**5.979**	**2.153**	1.274	6.562	**10.598**
Mk 07	20 x 5	**193.196**	212.9	**30.09**	34.74	22.65	**23.74**	68.78	**69.77**
Mk 08	20x10	585.142	**572.823**	**6.64**	17.94	4.624	**5.158**	**9.18**	7.7846
Mk 09	20x10	426.732	**378.553**	54.01	**31.918**	**34.360**	31.107	**94.66**	89.18
Mk 10	20x15	**941.95**	1016.71	512.16	**304.73**	**302.92**	300.25	**432.55**	409.43
1a	10x5	2569.941	**2566.48**	**36.1**	39.45	**28.513**	23.7870	65.397	**68.216**
2a	10x5	2427.752	**2358.34**	**263.23**	349.18	**128.801**	125.165	**495.1**	430.6
3a	10x5	**2540.038**	2781.75	344.798	**381.75**	**237.84**	235.678	**840.107**	824.4
4a	10x5	2550.6519	**2496.806**	107.73	**108.03**	62.82	**73.2**	**257.068**	234.362
5a	10x5	**2368.582**	2478.792	**174.801**	280.5	126.849	**165.92**	386.09	**557.396**
6a	10x5	**2312.461**	2338.38	**140.594**	154.11	**69.876**	61.56	**266.430**	242.667
7a	15x8	**2446.998**	2483.246	**100.6**	124.73	59.655	**73.91**	**173.53**	149.5
8a	15x8	**2319.009**	2416.21	**100.454**	113.51	**48.775**	38.146	**179.895**	165.25
9a	15x8	**2206.967**	2215.49	**117.54**	127.974	**113.0686**	104.72	**378.748**	331.66
10a	15x8	**2458.58**	2478.9424	93.285	**81.55**	51.95	**53.768**	**176.418**	162.435
11a	15x8	**2182.0226**	2257.72	141.42	**112.32**	**100.30**	68.0005	**346.93**	212.62
12a	15x8	**2164.65**	2458.46	**264.19**	292.63	**107.11**	94.74	**256.41**	217.72
13a	20x10	**3323.468**	3375.571	**219.22**	306	**71.80629**	67.090	362.095	**374.108**
14a	20x10	**3172.51**	3211.188	**376.417**	391.62	**526.51**	504.348	**817.54**	809.316
15a	20x10	**3249.22**	3512.75	**457.15**	485.72	**317.02**	305.44	**241.37**	218.71
16a	20x10	**3002.78**	3643.46	**413.71**	457.18	277.49	**304.27**	**224.33**	204.24
17a	20x10	**3011.3**	3126.245	**247.29**	257.94	**269.2**	244.78	**514.75**	498.38
18a	20x10	**3261.75**	3456.68	**527.17**	612.35	**129.81**	108.3	**421.74**	413.56

* The best obtained values are marked in bold letter

Table 6.8 represents the best makespan value obtained by maximum deviation theory based on the of highest composite score and the percentage deviation (%PD) [Equation 3.18] of the makespan from the lower bound (LB) [121,129].

Table 6.8 The computational results of maximum deviation theory

Problem	n x m	LB	Objectives: Makespan and mean flow time				Objectives: Makespan and mean tardiness			
			MOPSO		NSGA-II		MOPSO		NSGA-II	
			best C_{max} by MDT	%PD	best C_{max} by MDT	%PD	best C_{max} by MDT	%PD	best C_{max} by MDT	%PD
Mk 01	10 x 6	36	50	38.89	52	44.44	49	36.11	52	44.44
Mk 02	10 x 6	24	37	54.17	39	62.50	30	25.00	45	87.50
Mk 03	15 x 8	204	236	15.69	258	26.47	220	7.84	235	15.2
Mk 04	15 x 8	48	84	75.00	87	81.25	75	56.25	79	64.58
Mk 05	15 x 4	168	199	18.45	206	22.62	199	18.45	204	21.43
Mk 06	10x15	33	125	278.79	128	287.88	130	293.94	132	300.00
Mk 07	20 x 5	133	210	57.89	188	41.35	171	28.57	182	36.84
Mk 08	20x10	523	564	7.84	542	3.63	533	1.91	551	5.35
Mk 09	20x10	299	371	24.08	384	28.43	374	25.08	382	27.76
Mk 10	20x15	165	220	33.33	235	42.42	195	18.18	211	27.88
1a	10x5	2,505	2522	0.68	2578	2.91	2522	0.68	2535	1.20
2a	10x5	2,228	2421	8.66	2541	14.05	2527	13.42	2599	16.65
3a	10x5	2,228	2934	31.69	3242	45.51	2540	14.00	2793	25.36
4a	10x5	2,503	2533	1.20	2616	4.51	2561	2.32	2524	0.84
5a	10x5	2,189	2421	10.60	2464	12.56	2373	8.41	2418	10.46
6a	10x5	2,162	2409	11.42	2606	20.54	2279	5.41	2291	5.97
7a	15x8	2,187	2535	15.91	2437	11.43	2443	11.71	2513	14.91
8a	15x8	2,061	2811	36.39	2867	39.11	2793	35.52	2845	38.04
9a	15x8	2,061	2827	37.17	2814	36.54	2641	28.14	2771	34.45
10a	15x8	2,178	2462	13.04	2564	17.72	2402	10.28	2517	15.56
11a	15x8	2,017	2294	13.73	2374	17.70	2192	8.68	2249	11.50
12a	15x8	1,969	2208	12.14	2410	22.40	2145	8.94	2182	10.82
13a	20x10	2,161	2297	6.29	2384	10.32	2637	22.03	2512	16.24
14a	20x10	2,161	2178	0.79	2477	14.62	2291	6.02	2298	6.34
15a	20x10	2,161	2209	2.22	2381	10.18	2265	4.81	2310	6.89
16a	20x10	2,148	2249	4.70	2392	11.36	2183	1.63	2271	5.73
17a	20x10	2,088	2140	2.49	2267	8.57	2252	7.85	2319	11.06
18a	20x10	2,057	2355	14.49	2496	21.34	2354	14.44	2401	16.72
Average Percentage Deviation				29.56		34.37		25.56		31.42

It is observed from the Table 6.8 that most of the best makespan values obtained by MDT from the non-dominated solutions of MOPSO have smaller %PD than the NSGA-II. Average percentage deviation (APD) of makespan by the proposed MOPSO and NSGA-II are 29.56 and 34.37 respectively for objectives of makespan and mean flow time.

Similarly, the average percentage deviation (APD) of makespan by the proposed MOPSO and NSGA-II are 25.56 and 31.42 respectively for objectives of makespan and mean tardiness. The improvement rate of APD is found to be 13.99 % for the objectives makespan and mean flow time and 18.65 % for the objectives makespan and mean tardiness.

6.5 Conclusions

In this chapter, benchmark instances from literature for flexible flow shop and flexible job shop scheduling problem are solved by an efficient multi-objective particle swarm optimization to find near-optimal schedules. The mutation operator generally used in genetic algorithm is embedded in MOPSO to avoid premature convergence and improve solution diversity. Further, maximum deviation theory (MDT) is used to determine the weights of the attributes to develop a composite score to ease the decision maker for selecting the best solution from a large set of Pareto solutions. The composite score for all the non-dominated solutions is obtained through summing the weighted objective values. The best solution is selected from all the non-dominated solution considering the highest composite score to avoid subjectiveness and impreciseness in the decision making for the managers. This work offers an effective guideline to select optimum schedule for achieving the desired different objective simultaneously. From the comparative analysis, it can be concluded that the MOPSO algorithm is superior to NSGA-II for different performance measures. The improvement rate of APD of MOPSO is found to be 8.66 % for the objectives makespan and mean flow time and 9.62 % for the objectives makespan and mean tardiness over NSGA-II for FFSP benchmark problems and an improvement of 13.99 % for APD of MOPSO with respect to NSGA-II for the objectives makespan and mean flow time is reported. Similarly, an improvement of 18.65 % for APD of MOPSO with respect to for the objectives makespan and mean tardiness is obtained for the instances of FJSP. The next chapter presents the summary of the results, recommendations and scope for future work in the direction of job scheduling.

CHAPTER 7

DISCUSSION AND CONCLUSION

7.1 Introduction

In this thesis, flexible flow shop and job shop scheduling problem considered as NP-hard problems are dealt in detail using efficient algorithms such as PSO and QPSO. A mutation operator commonly used in genetic algorithm is embedded in PSO and QPSO to avoid premature convergence and improve solution diversity. Solution diversity is also improved through the use of chaotic numbers (Logistic map) instead of random numbers to improve the diversity so that exploration capability of the algorithms can be enhanced. Thus, computational efforts can be reduced to a large extent. Generally, swarm optimization is used for a continuous optimization problem but the scheduling is a combinatorial optimization problem. Section 3.4 and section 4.2 illustrate mapping mechanism for combinatorial problem in a continuous domain applied to flexible flow and job shop scheduling problems respectively. Single and multiple objective frameworks have been proposed for solving such problems. In addition, scheduling in an uncertain environment where machine breakdown occurs has been demonstrated.

7.2 Summary of findings

An extensive computational study has been carried out on a set of seventy seven benchmark instances taken from Carlier and Nerons [196] by the help of efficient PSO and QPSO algorithm. Simulation tests of benchmark instances demonstrates that proposed PSO and QPSO algorithm are capable of solving FFSP effectively and efficiently due to the balance of global exploration and local exploitation capability. Comparison of simulation results with the existing results of several algorithms like GA, AIS, ACO, TS, AIS and B&B in terms of percentage deviation leads to the conclusion that proposed algorithms results in less percentage deviation from the lower bound. The improvement rate of average percentage deviation (APD) of proposed QPSO is found to be 26.08112 % with respect to proposed PSO, 37.73% with respect to GA, 38.95% with respect to AIS, 29.946% with respect to ACO and 72.18 % with respect to branch and bound (B&B) for seventy seven benchmark problems considered in the study. Simulation for FJSP was conducted on three sets of problem instances from Kacem et al., [117], Brandimarte [112] and Dauzere-peres [115] (DP data). It has been observed that the results obtained by proposed QPSO has an improvement rate of 7.369 % with respect to proposed PSO, 27.936% with respect to tabu search (TS), 31.836 % with respect to hybrid genetic algorithm (hGA) for the DP data set in terms of APD. The improvement rate in terms of APD is 1.889 % with respect to proposed PSO, 7.534% with respect to GA, 0.825% with respect to parallel variable neighborhood search (PVNS) algorithm,

5.5366 % with respect to Integrated genetic algorithm (IGA) and 26.63% with respect to knowledge based ant colony optimization (KBACO) for the Brandimarte data set. For the Dauzere-peres data set, the proposed QPSO has an improvement rate (in APD) of 7.369 % with respect to proposed PSO, 27.936% with respect to TS, and 31.836 % with respect to hGA. It is conceivable to note that the algorithm with chaotic numbers improves the makespan and converges towards the best value faster. This is because of higher degree of disorderness of the chaotic numbers which facilitates high diversity in the particles and helps the algorithm to converge rapidly towards the solution. The convergence curves represent that the proposed algorithms (embedding the mutation operator in the proposed PSO and proposed QPSO) helps to improve the solution diversity and avoid premature convergence.

A multi-objective framework is analyzed by implementing the proposed PSO and QPSO algorithms with different robustness measures for FFSP and FJSP considering machine breakdown. An experimental study is conducted to investigate the solution quality of algorithms with robustness measure. Four factors such as breakdown (BD), robustness measure (RM), algorithm (ALGORITHM) and problem instances (TEST CASE) at different levels are considered. A full factorial experiment design having thirty six experimental runs (2×3×2×3) has been conducted to gather sufficient information on model behavior with less number of experiments. Under each experiment, multi-objective performance measure (Z) is computed. Analysis of variance (ANOVA) is performed on the response to find out significant factors influencing the FFSP and FJSP under uncertainty situation (i.e. random machine breakdowns). ANOVA results reveal that the robustness measure RM3 (the expected realized total flow time) can significantly improve the quality of the schedule in both FFSP and FJSP. Simulation for FFSP and FJSP in a machine breakdown condition is conducted for two disruption levels i.e low disruption level (BD1) and high disruption level (BD2). The improvement rate of the proposed QPSO is found to be 2.151% in a low disruption level (BD1) and 1.97% in a high disruption level (BD2) for the FFSP benchmark problems. Similarly, QPSO results an improvement of 2.56%, 2.23% and 1.24% in a low disruption level (BD1), 2.22%, 1.55% and 1.67% in a high disruption level (BD2) for Kacem, BR and DP data set respectively over proposed PSO for FJSP benchmark problems. Therefore, it is concluded that the proposed QPSO algorithm is quite effective to produce robust schedule in an uncertainty condition.

A multi-objective framework is investigated to compare scalarization method and pareto optimality approach using multi-objective particle swarm optimization (MOPSO) technique. MOPSO happens to be superior to scalarization method because the MOPSO provides smaller APD from lower bound. An efficient multi-objective particle swarm optimization has been proposed and compared with another popular multi-objective algorithm known as NSGA-II. The improvement rate of APD in MOPSO is found to be 8.66 % for the objectives makespan and mean flow time and 9.62 % for the objectives makespan and mean tardiness over NSGA-II for FFSP benchmark problems. Similarly, an improvement of 13.99 % in terms of APD using MOPSO for the objectives as makespan and mean flow time is obtained for FJSP over NSGA-II. An improvement of 18.65 % in APD using MOPSO with respect to NSGA-II for the objectives as makespan and mean tardiness is obtained.

7.3 Contribution of the research work

PSO has an inherent drawback of getting trapped at local optimum due to large reduction in velocity values as iteration proceeds and poses difficulty in reaching at best solution. A new variant of PSO, called quantum-behaved particle swarm optimization (QPSO), has been proposed in order to improve the global search ability of the original PSO. The application of chaotic sequences based on chaotic logistic mapping instead of random sequences in PSO and QPSO happens to be a powerful strategy to diversify the initial population and improve the algorithm's performance by preventing the premature convergence to local minima of the algorithm. The mutation operator is introduced to improve the solution diversity and accelerate the convergence rate. The search mechanism of the proposed PSO and proposed QPSO is explored to solve FFSP and FJSP.

A multi-objective (scalarization method) with robustness measure has been proposed to obtain a robust schedule that minimizes the effect of machine breakdowns in the overall performance such that the makespan is preserved and decreases the deviations of operation completion times between the realized schedule and the predictive schedule. A novel multi-objective particle swarm optimization (MOPSO) technique is proposed for FFSP and FJSP with an objective to minimize makespan, mean flow time and mean tardiness with the goal of finding approximations of the optimal Pareto front. Maximum deviation theory has been proposed for ranking the non-

dominated solutions obtained from MOPSO to ease the decision making process of choosing the best solution from the non-dominated solution set.

7.4 Limitations of the study

In spite of advantages obtained through proposed study, few limitations exist in the study because they have not been addressed. The benchmark problems considered in this study are static scheduling problems but the real industrial environment is dynamic in nature i.e jobs to be processed continuously change over time. The dynamic nature of scheduling approach has not been considered in the study. Re-entrant jobs, buffer capacity and setup times, normally observed in real manufacturing environment, have not been not considered in this study because consideration of these real manufacturing factors increases the complexity of problem formulation.

Moreover, the work has been restricted to only three performance measures such as makespan, flow time and tardiness. Other performance measures of scheduling like lateness and number of tardy jobs can be considered to study the behavior of the proposed approach. Similarly, machine breakdown is considered as an uncertain event. However, many uncertainties in terms of processing time, operator absenteeism, job arrival, tool slots, availability of jigs and fixtures may exist in real manufacturing environment.

7.5 Scope of future research

As mentioned in section 7.4, the limitations of the study may be accounted for formulating the flexible flow and job shop problems to obtain schedules for real manufacturing environment. Particularly, the study may be extended to dynamic stochastic shop scheduling where the jobs arrive continuously in time and test the performance of proposed algorithms. Scope exists to extend the study for finding optimum schedules in flexible flow and job shop environment considering work load on critical machine and machine loading capacity. In many real manufacturing environments, the nature of the job and machine is probabilistic in nature so stochastic processing time may be considered for the enhancement of the study. In certain applications, setup times cannot be ignored or integrated into the processing times particularly when setup times is sequence-dependent (setup time depends on the previous and next operation performed on the machine). The study can be extended integrating set up time as a critical parameter in the scheduling problem.

More robust measures should be analyzed to obtain a better robust schedule in an uncertain scenario. Other uncertain events like rush jobs, due date changes, job cancellation, changes in job priority, early or late arrival of jobs, changes in job processing time and preventive maintenance (PM) many be considered in the future work. Similarly, buffer capacities between machines may be considered in the scheduling problem in future.

Bibliography

[1] Pinedo, M. (2005). Planning and scheduling in manufacturing and services, New York, Springer.

[2] McKay, K. N., and Wiers, V. (1999). Unifying the theory and practice of production scheduling. Journal of Manufacturing Systems, 18(4), 241-255.

[3] Zhou, H., Cheung, W., & Leung, L. C. (2009). Minimizing weighted tardiness of job-shop scheduling using a hybrid genetic algorithm. European Journal of Operational Research, 194(3), 637-649.

[4] Lenstra, J. K., Rinnooy Kan, A. H. G. and Brucker, P. (1977). Complexity of machine scheduling problems. Annals of Discrete Mathematics, 1, 343-362.

[5] Suliman, S. M. A. (2000). A two-phase heuristic approach to the permutation flow-shop scheduling problem. International Journal of Production Economics, 64(1), 143-152.

[6] Brah, S. A. and Hunsucker, J. L. (1991). Branch and bound algorithm for the flow shop with multiple processors. European Journal of Operational Research, 51(1), 88-99.

[7] Graves, S. C. (1981). A review of production scheduling. Operations research, 29(4), 646-675.

[8] Allaoui, H. and Artiba, A. (2004). Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints. Computers and Industrial Engineering, 47(4), 431-450.

[9] Kim, S. C., and Bobrowski, P. M. (1994). Impact of sequence-dependent setup time on job shop scheduling performance. The International Journal of Production Research, 32(7), 1503-1520.

[10] Baker, K. R. and Kanet, J. J. (1983). Job shop scheduling with modified due dates. Journal of Operations Management, 4(1), 11-22.

[11] Holthaus, O. and Rajendran, C. (1997). New dispatching rules for scheduling in a job shop-an experimental study. International Journal of Advanced Manufacturing Technology, 13(2), 148-153.

[12] Dahal, K. P., Tan, K. C. and Cowling, P. I. (2007). Evolutionary scheduling, Berlin, Springer.

[13] Baker, K. R. and Trietsch, D. (2009). Principles of sequencing and scheduling. New York, John Wiley and Sons.

[14] Cheng, R., Gen, M and Tsujimura, Y. (1996). A tutorial survey of job-shop scheduling problems using genetic algorithms. Computers and Industrial Engineering, 30(4), 983-997.

[15] Morton, T. (1993). Heuristic scheduling systems: with applications to production systems and project management (Vol. 3). John Wiley & Sons.

[16] McKay, K. N., and Wiers, V. (1999). Unifying the theory and practice of production scheduling. Journal of Manufacturing Systems, 18(4), 241-255.

[17] Rajendran, C. and Chaudhuri, D. (1992). An efficient heuristic approach to the scheduling of jobs in a flow-shop. European Journal of Operational Research, 61(3), 318-325.

[18] Manikas, A. and Chang, Y. L. (2009). Multi-criteria sequence-dependent job shop scheduling using genetic algorithms. Computers and Industrial Engineering, 56(1), 179-185.

[19] Alaykyran, K., Engin, O. and Doyen, A. (2007). Using ant colony optimization to solve hybrid flow shop scheduling problems. International Journal of Advanced Manufacturing Technology, 35(5-6), 541-550.

[20] Engin, O. and Doyen, A. (2004). A new approach to solve hybrid flow shop scheduling problems by artificial immune system. Future Generation Computer Systems, 20(6), 1083-1095.

[21] Qian, B., Wang, L., Huang, D. X., Wang, W. L. and Wang, X. (2009). An effective hybrid DE-based algorithm for multi-objective flow shop scheduling with limited buffers. Computers and Operations Research, 36(1), 209-233.

[22] Sha, D. Y. and Lin, H. H. (2010). A multi-objective PSO for job-shop scheduling problems. Expert Systems with Applications, 37(2), 1065-1070.

[23] Belmecheri, F., Prins, C., Yalaoui, F. and Amodeo, L. (2013). Particle swarm optimization algorithm for a vehicle routing problem with heterogeneous fleet, mixed backhauls, and time windows. Journal of Intelligent Manufacturing, 24(4), 775-789.

[24] Bachlaus, M., Pandey, M. K., Mahajan, C., Shankar, R. and Tiwari, M. K. (2008). Designing an integrated multi-echelon agile supply chain network: a hybrid taguchi-particle swarm optimization approach. Journal of Intelligent Manufacturing, 19(6), 747-761.

[25] Kim, B. I. and Son, S. J. (2012). A probability matrix based particle swarm optimization for the capacitated vehicle routing problem. Journal of Intelligent Manufacturing, 23(4), 1119-1126.

[26] Johnson, S. M. (1954). Optimal two-and three-stage production schedules with setup times included. Naval Research Logistics Quarterly, 1(1), 61-68.

[27] Palmer, D. S. (1965). Sequencing jobs through a multi-stage process in the minimum total time: a quick method of obtaining a near optimum. Operations Research, 16(1), 101-107.

[28] Campbell, H. R. and Smith, D. M. (1970). A heuristic algorithm for n-jobs m-machines sequencing problem. Management Science, 16, 630–637.

[29] Gupta, J. N. (1971). A functional heuristic algorithm for the flow-shop scheduling problem. Operational Research, 22(1), 39-47.

[30] Nawaz, M., Enscore, J.,andHam, I. (1983). A Heuristic algorithm for the M machine, n-job flow shop sequencing problem. OMEGA, 11(1), 91–95.

[31] Ho, J. C. and Chang, Y.-L. (1991). A new heuristic for then-jobs, m-machine flow-shop scheduling problem. European Journal of Operational Research, 52(1), 194-202.

[32] Rajendran, C. (1995). Heuristics for scheduling in flow-shop with multiple objectives. European Journal of Operational Research, 82(3), 540-555.

[33] Choi, I. C. and Choi, D. S. (2002). A local search algorithm for job-shop scheduling problems with alternative operations and sequence-dependent setups. Computers and Industrial Engineering, 42(1), 43-58.

[34] Bulfin, R. L. and M'hallah, R. (2003). Minimizing the weighted number of tardy jobs on a two-machine flow shop. Computers and Operations Research, 30(12), 1887-1900.

[35] Pranzo, M. (2004). Batch scheduling in a two-machine flow shop with limited buffer and sequence independent setup times and removal times. European Journal of Operational Research, 153(3), 581-592.

[36] Brown, S. I., McGarvey, R. G. and Ventura, J. A. (2004). Total flow time and makespan for a no-wait m-machine flow-shop with set-up times separated. Journal of the Operational Research Society, 55(6), 614-621.

[37] Blazewicz, J., Pesch, E., Sterna, M. and Werner, F. (2005). A comparison of solution procedures for two-machine flow shop scheduling with late work criterion. Computers and Industrial Engineering, 49(4), 611-624.

[38] Grabowski, J. and Pempera, J. (2005). Some local search algorithms for no-wait flow-shop problem with makespan criterion. Computers and Operations Research, 32(8), 2197-2212.

[39] França, P. M., Tin Jr, G. and Buriol, L. S. (2006). Genetic algorithms for the no-wait flowshop sequencing problem with time restrictions. International Journal of Production Research, 44(5), 939-957.

[40] Fink, A. and Vob, S. (2003). Solving the continuous flow-shop scheduling problem by meta-heuristics. European Journal of Operational Research, 151(2), 400-414.

[41] Ponnambalam, S. G., Jagannathan, H., Kataria, M. and Gadicherla, A. (2004). A TSP-GA multi-objective algorithm for flow-shop scheduling. International Journal of Advanced Manufacturing Technology, 23(11-12), 909-915.

[42] Lodree Jr, E., Jang, W. and Klein, C. M. (2004). A new rule for minimizing the number of tardy jobs in dynamic flow shops. European Journal of Operational Research, 159(1), 258-263.

[43] Ravindran, D., Selvakumar, S. J., Sivaraman, R. and Haq, A. N. (2005). Flow shop scheduling with multiple objective of minimizing makespan and total flow time. International Journal of Advanced Manufacturing Technology, 25(9-10), 1007-1012.

[44] Ignall, E. and Schrage, L. (1965). Application of the branch and bound technique to some flow-shop scheduling problems. Operations Research, 13(3), 400-412.

[45] Hariri, A. M. A. and Potts, C. N. (1989). A branch and bound algorithm to minimize the number of late jobs in a permutation flow-shop. European Journal of Operational Research, 38(2), 228-237.

[46] Carlier, J. and Rebaï, I. (1996). Two branch and bound algorithms for the permutation flow shop problem. European Journal of Operational Research, 90(2), 238-251.

[47] Fry, T. D., Armstrong, R. D., Darby-Dowman, K. and Philipoom, P. R. (1996). A branch and bound procedure to minimize mean absolute lateness on a single processor. Computers and Operations Research, 23(2), 171-182.

[48] Błazewicz, J., Pesch, E., Sterna, M. and Werner, F. (2005). The two-machine flow-shop problem with weighted late work criterion and common due date. European Journal of Operational Research, 165(2), 408-415.

[49] Gowrishankar, K., Rajendran, C. and Srinivasan, G. (2001). Flow shop scheduling algorithms for minimizing the completion time variance and the sum of squares of completion time deviations from a common due date. European Journal of Operational Research, 132(3), 643-665.

[50] Nowicki, E. and Smutnicki, C. (2006). Some aspects of scatter search in the flow-shop problem. European Journal of Operational Research, 169(2), 654-666.

[51] Su, L. H. and Lee, Y. Y. (2008). The two-machine flow-shop no-wait scheduling problem with a single server to minimize the total completion time. Computers and Operations Research, 35(9), 2952-2963.

[52] Aldowaisan, T. (2001). A new heuristic and dominance relations for no-wait flowshops with setups. Computers and Operations Research, 28(6), 563-584.

[53] Santos, D. L., Hunsucker, J. L. and Deal, D. E. (2001). On makespan improvement in flow shops with multiple processors. Production Planning and Control, 12(3), 283-295.

[54] Wang, J. B., Daniel Ng, C. T., Cheng, T. E. and Liu, L. L. (2006). Minimizing total completion time in a two-machine flow shop with deteriorating jobs. Applied Mathematics and Computation, 180(1), 185-193.

[55] Solimanpur, M., Vrat, P. and Shankar, R. (2004). A neuro-tabu search heuristic for the flow shop scheduling problem. Computers and Operations Research, 31(13), 2151-2164.

[56] Taillard, E. (1990). Some efficient heuristic methods for the flow shop sequencing problem. European Journal of Operational Research, 47(1), 65-74.

[57] Rajendran, C. and Ziegler, H. (2004). Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flow time of jobs. European Journal of Operational Research, 155(2), 426-438.

[58] Shyu, S. J., Lin, B. M. T. and Yin, P. Y. (2004). Application of ant colony optimization for no-wait flow-shop scheduling problem to minimize the total completion time. Computers and Industrial Engineering, 47(2), 181-193.

[59] Lian, Z., Gu, X. and Jiao, B. (2006). A similar particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan. Applied Mathematics and Computation, 175(1), 773-785.

[60] Pan, Q. K., Fatih Tasgetiren, M. and Liang, Y. C. (2008). A discrete particle swarm optimization algorithm for the no-wait flow-shop scheduling problem. Computers and Operations Research, 35(9), 2807-2839.

[61] Kuo, I., Horng, S. J., Kao, T. W., Lin, T. L., Lee, C. L., Terano, T. and Pan, Y. (2009). An efficient flow-shop scheduling algorithm based on a hybrid particle swarm optimization model. Expert Systems with Applications, 36(3), 7027-7032.

[62] Tseng, L. Y. and Lin, Y. T. (2009). A hybrid genetic local search algorithm for the permutation flow-shop scheduling problem. European Journal of Operational Research, 198(1), 84-92.

[63] Salmasi, N., Logendran, R. and Skandari, M. R. (2011). Makespan minimization of a flow-shop sequence-dependent group scheduling problem. International Journal of Advanced Manufacturing Technology, 56(5-8), 699-710.

[64] Muth, J. F. and Thompson, G. L. (1963). Industrial scheduling. New Jersey, Prentice-Hall.

[65] Lenstra, J. K. and Kan, A. H. G. (1981). Complexity of vehicle routing and scheduling problems. Networks, 11(2), 221-227.

[66] Carlier, J. and Pinson, E. (1989). An algorithm for solving the job-shop problem. Management Science, 35(2), 164-176.

[67] Adams, J., Balas, E. and Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. Management Science, 34(3), 391-401.

[68] Singer, M. and Pinedo, M. (1998). A computational study of branch and bound techniques for minimizing the total weighted tardiness in job shops. IIE Transactions, 30(2), 109-118.

[69] Amaral Armentano, V. and Rigao Scrich, C. (2000). Tabu search for minimizing total tardiness in a job shop. International Journal of Production Economics, 63(2), 131-140.

[70] Choi, I. C. and Choi, D. S. (2002). A local search algorithm for job-shop scheduling problems with alternative operations and sequence-dependent setups. Computers and Industrial Engineering, 42(1), 43-58.

[71] Artigues, C. and Roubellat, O. (2002). An efficient algorithm for operation insertion in a multi-resource job-shop schedule with sequence-dependent setup times. Production Planning and Control, 13(2), 175-186.

[72] Low, C., Hsu, C. M. and Huang, K. I. (2004). Benefits of lot splitting in job-shop scheduling. International Journal of Advanced Manufacturing Technology, 24(9-10), 773-780.

[73] Subramaniam, V., Ramesh, T., Lee, G. K., Wong, Y. S. and Hong, G. S. (2000). Job shop scheduling with dynamic fuzzy selection of dispatching rules. International Journal of Advanced Manufacturing Technology, 16(10), 759-764.

[74] Fandel, G. and Stammen-Hegene, C. (2006). Simultaneous lot sizing and scheduling for multi-product multi-level production. International Journal of Production Economics, 104(2), 308-316.

[75] Yang, H. A., Xu, Y. P., Sun, S. D. and Yu, J. J. (2006). A Job Shop Scheduling Heuristic Algorithm Based on Probabilistic Model of the Search Space. Materials Science Forum, 532, 1084-1087

[76] Yamada, T. and Nakano, R. (1997). Genetic algorithms for job-shop scheduling problems. In Proceedings of the Modern Heuristics for Decision Support, 67-81, London (UK), March 1997, 67–81

[77] Jaszkiewicz, A. (2002). Genetic local search for multi-objective combinatorial optimization. European Journal of Operational Research, 137(1), 50-71.

[78] Colorni, A., Dorigo, M., Maniezzo, V. and Trubian, M. (1994). Ant system for job-shop scheduling. Belgian Journal of Operations Research, Statistics and Computer Science, 34(1), 39-53.

[79] Steinhofel, K., Albrecht, A. and Wong, C. K. (1999). Two simulated annealing-based heuristics for the job shop scheduling problem. European Journal of Operational Research, 118(3), 524-548.

[80] Kolonko, M. (1999). Some new results on simulated annealing applied to the job shop scheduling problem. European Journal of Operational Research, 113(1), 123-136.

[81] Zuo, X. Q. and Fan, Y. S. (2005). Solving the job shop scheduling problem by an immune algorithm. In Proceedings of International Conference on Machine Learning and Cybernetics, Guangzhou, China, August 2005, 6, 3282-3287.

[82] Chandrasekaran, M., Asokan, P., Kumanan, S., Balamurugan, T. and Nickolas, S. (2006). Solving job shop scheduling problems using artificial immune system. International Journal of Advanced Manufacturing Technology, 31(5-6), 580-593.

[83] Schuster, C. J. (2006). No-wait job shop scheduling: Tabu search and complexity of sub-problems. Mathematical Methods of Operations Research, 63(3), 473-491.

[84] Essafi, I., Mati, Y. and Dauzere-Peres, S. (2008). A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem. Computers and Operations Research, 35(8), 2599-2616.

[85] Baptiste, P., Flamini, M. and Sourd, F. (2008). Lagrangian bounds for just-in-time job-shop scheduling. Computers and Operations Research, 35(3), 906-915.

[86] Roshanaei, V., Naderi, B., Jolai, F. and Khalili, M. (2009). A variable neighborhood search for job shop scheduling with set-up times to minimize makespan. Future Generation Computer Systems, 25(6), 654-661.

[87] Bozejko, W., Uchronski, M. and Wodecki, M. (2010). Parallel hybrid metaheuristics for the flexible job shop problem. Computers and Industrial Engineering, 59(2), 323-333.

[88] Pan, J. C. H. and Huang, H. C. (2009). A hybrid genetic algorithm for no-wait job shop scheduling problems. Expert Systems with Applications, 36(3), 5800-5806.

[89] Gu, J., Gu, M., Cao, C. and Gu, X. (2010). A novel competitive co-evolutionary quantum genetic algorithm for stochastic job shop scheduling problem. Computers and Operations Research, 37(5), 927-937.

[90] Kachitvichyanukul, V. and Sitthitham, S. (2011). A two-stage genetic algorithm for multi-objective job shop scheduling problems. Journal of Intelligent Manufacturing, 22(3), 355-365.

[91] Mati, Y., Dauzere-Peres, S. and Lahlou, C. (2011). A general approach for optimizing regular criteria in the job-shop scheduling problem. European Journal of Operational Research, 212(1), 33-42.

[92] Arthanari, T. S. and Ramamurthy, K. G. (1971). An extension of two machines sequencing problem. European Journal of Operations Research, 8(1), 10-22.

[93] Guinet, A., Solomon, M. M., Kedia, P. K. and Dussauchoy, A. (1996). A computational study of heuristics for two-stage flexible flow-shop. International Journal of Production Research, 34(5), 1399-1415.

[94] Gupta, J. N. and Tunc, E. A. (1994). Scheduling a two-stage hybrid flow-shop with separable setup and removal times. European Journal of Operational Research, 77(3), 415-428.

[95] Liu, C. Y. and Chang, S. C. (2000). Scheduling flexible flow shops with sequence-dependent setup effects. IEEE Transactions on Robotics and Automation, 16(4), 408-419.

[96] Botta-Genoulaz, V. (2000). Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness. International Journal of Production Economics, 64(1), 101-111.

[97] Neron, E., Baptiste, P. and Gupta, J. N. (2001). Solving hybrid flow shop problem using energetic reasoning and global operations. Omega, 29(6), 501-511.

[98] Gupta, J. N., Kruger, K., Lauff, V., Werner, F. and Sotskov, Y. N. (2002). Heuristics for hybrid flow shops with controllable processing times and assignable due dates. Computers and Operations Research, 29(10), 1417-1439.

[99] Su, L. H. (2003). A hybrid two-stage flow-shop with limited waiting time constraints. Computers and Industrial Engineering, 44(3), 409-424.

[100] Hmida, A. B., Huguet, M. J., Lopez, P. and Haouari, M. (2007). Climbing depth-bounded discrepancy search for solving hybrid flow shop problems. European Journal of Industrial Engineering, 1(2), 223-243.

[101] Kurz, M. E. and Askin, R. G. (2004). Scheduling flexible flow lines with sequence-dependent setup times. European Journal of Operational Research, 159(1), 66-82.

[102] Low, C. (2005). Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. Computers and Operations Research, 32(8), 2013-2025.

[103] Oguz, C. and Ercan, M. F. (2005). A genetic algorithm for hybrid flow-shop scheduling with multiprocessor tasks. Journal of Scheduling, 8(4), 323-351.

[104] Ruiz, R. and Maroto, C. (2006). A genetic algorithm for hybrid flow-shops with sequence dependent setup times and machine eligibility. European Journal of Operational Research, 169(3), 781-800.

[105] Janiak, A., Kozan, E., Lichtenstein, M. and Oguz, C. (2007). Meta-heuristic approaches to the hybrid flow shop scheduling problem with a cost-related criterion. International Journal of Production Economics, 105(2), 407-424.

[106] Ying, K. C. and Lin, S. W. (2006). Multiprocessor task scheduling in multistage hybrid flow-shops: an ant colony system approach. International Journal of Production Research, 44(16), 3161-3177.

[107] Tseng, C. T. and Liao, C. J. (2008). A particle swarm optimization algorithm for hybrid flow-shop scheduling with multiprocessor tasks. International Journal of Production Research, 46(17), 4655-4670.

[108] Kahraman, C., Engin, O., Kaya, I. and Kerim Yilmaz, M. (2008). An application of effective genetic algorithms for solving hybrid flow shop scheduling problems. International Journal of Computational Intelligence Systems, 1(2), 134-147.

[109] Allahverdi, A. and Al-Anzi, F. S. (2009). The two-stage assembly scheduling problem to minimize total completion time with setup times. Computers and Operations Research, 36(10), 2740-2747.

[110] Al-Anzi, F. S. and Allahverdi, A. (2009). Heuristics for a two-stage assembly flow-shop with bi-criteria of maximum lateness and makespan. Computers and Operations Research, 36(9), 2682-2689.

[111] Mirsanei, H. S., Zandieh, M., Moayed, M. J. and Khabbazi, M. R. (2011). A simulated annealing algorithm approach to hybrid flow shop scheduling with sequence-dependent setup times. Journal of Intelligent Manufacturing, 22(6), 965-978.

[112] Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. Annals of Operations research, 41(3), 157-183.

[113] Saidi-Mehrabad, M. and Fattahi, P. (2007). Flexible job shop scheduling with tabu search algorithms. International Journal of Advanced Manufacturing Technology, 32(5-6), 563-570.

[114] Brucker, P., Hurink, J. and Werner, F. (1996). Improving local search heuristics for some scheduling problems. Discrete Applied Mathematics, 65(1), 97-122.

[115] Dauzere-peres, paulli E. (1997). An integrated approach for modeling and solving the general multi-processor job shop scheduling problem using tabu search. Annals of operation research,70, 281-306

[116] Mastrolilli, M. and Gambardella, L.M. (2000). Effective neighborhood functions for the flexible job shop problem. Journal of Scheduling, 3(1), 3-20.

[117] Kacem, I., Hammadi, S. and Borne, P. (2002). Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 32(1), 1-13.

[118] Zhang H.P, Gen M. (2005). Multistage-based genetic algorithm for flexible job-shop scheduling problem. Journal of Complexity, 11, 223–232.

[119] Fattahi, P., Mehrabad, M. S. and Jolai, F. (2007). Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. Journal of Intelligent Manufacturing, 18(3), 331-342.

[120] Ho, N. B., Tay, J. C. and Lai, E. M. K. (2007). An effective architecture for learning and evolving flexible job-shop schedules. European Journal of Operational Research, 179(2), 316-333.

[121] Pezzella, F., Morganti, G. and Ciaschetti, G. (2008). A genetic algorithm for the flexible job-shop scheduling problem. Computers and Operations Research, 35(10), 3202-3212.

[122] Gao, J., Sun, L. and Gen, M. (2008). A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Computers and Operations Research, 35(9), 2892-2907.

[123] Xing, L. N., Chen, Y. W., Wang, P., Zhao, Q. S. and Xiong, J. (2010). A knowledge-based ant colony optimization for flexible job shop scheduling problems. Applied Soft Computing, 10(3), 888-896.

[124] Bagheri, A., Zandieh, M., Mahdavi, I. and Yazdani, M. (2010). An artificial immune algorithm for the flexible job-shop scheduling problem. Future Generation Computer Systems, 26(4), 533-541.

[125] Yazdani, M., Amiri, M. and Zandieh, M. (2010). Flexible job-shop scheduling with parallel variable neighborhood search algorithm. Expert Systems with Applications, 37(1), 678-687.

[126] Defersha, F. M. and Chen, M. (2010). A parallel genetic algorithm for a flexible job-shop scheduling problem with sequence dependent setups. International Journal of Advanced Manufacturing Technology, 49(1-4), 263-279.

[127] Ben-Hmida, A., Haouari, M., Huguet, M. J. and Lopez, P. (2010). Discrepancy search for the flexible job shop scheduling problem. Computers and Operations Research, 37(12), 2192-2201.

[128] Jurisch, B. (1995). Lower bounds for the job-shop scheduling problem on multi-purpose machines. Discrete Applied Mathematics, 58(2), 145-156.

[129] Zhang, G., Gao, L. and Shi, Y. (2011). An effective genetic algorithm for the flexible job-shop scheduling problem. Expert Systems with Applications, 38(4), 3563-3573.

[130] Vieira, G. E., Herrmann, J. W. and Lin, E. (2003). Rescheduling manufacturing systems: a framework of strategies, policies, and methods. Journal of Scheduling, 6(1), 39-62.

[131] Gholami, M., Zandieh, M. and Alem-Tabriz, A. (2009). Scheduling hybrid flow shop with sequence-dependent setup times and machines with random breakdowns. International Journal of Advanced Manufacturing Technology, 42(1-2), 189-201.

[132] Lee, C. Y. (1997). Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint. Operations Research Letters, 20(3), 129-139.

[133] Allahverdi, A. and Mittenthal, J. (1998). Dual criteria scheduling on a two-machine flow-shop subject to random breakdowns. International Transactions in Operational Research, 5(4), 317-324.

[134] Mehta, S. V. and Uzsoy, R. M. (1998). Predictable scheduling of a job shop subject to breakdowns. IEEE Transactions on Robotics and Automation, 14(3), 365-378.

[135] Holthaus, O. (1999). Scheduling in job shops with machine breakdowns: an experimental study. Computers and Industrial Engineering, 36(1), 137-162.

[136] Sabuncuoglu, I. and Bayız, M. (2000). Analysis of reactive scheduling problems in a job shop environment. European Journal of Operational Research, 126(3), 567-586.

[137] Jensen, M. T. (2003). Generating robust and flexible job shop schedules using genetic algorithms. IEEE Transactions on Evolutionary Computation, 7(3), 275-288.

[138] Allaoui, H. and Artiba, A. (2004). Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints. Computers and Industrial Engineering, 47(4), 431-450.

[139] Rangsaritratsamee, R., Ferrell Jr, W. G. and Kurz, M. B. (2004). Dynamic rescheduling that simultaneously considers efficiency and stability. Computers and Industrial Engineering, 46(1), 1-15.

[140] Kasap, N., Aytug, H. and Paul, A. (2006). Minimizing makespan on a single machine subject to random breakdowns. Operations Research Letters, 34(1), 29-36.

[141] Suwa, H. and Sandoh, H. (2007). Capability of cumulative delay based reactive scheduling for job shops with machine breakdowns. Computers and Industrial Engineering, 53(1), 63-78.

[142] Goren, S. and Sabuncuoglu, I. (2009). Optimization of schedule robustness and stability under random machine breakdowns and processing time variability. IIE Transactions, 42(3), 203-220.

[143] Yahyaoui, A., Fnaiech, N. and Fnaiech, F. (2009). New shifting method for job shop scheduling subject to invariant constraints of resources availability. In proceeding of 35th Annual Conference of IEEE on Industrial Electronics, Porto, Portugal, November 2009, 3387-3392.

[144] Zandieh, M. and Adibi, M. A. (2010). Dynamic job shop scheduling using variable neighborhood search. International Journal of Production Research, 48(8), 2449-2458.

[145] Lei, D. (2011). Scheduling stochastic job shop subject to random breakdown to minimize makespan. International Journal of Advanced Manufacturing Technology, 55(9-12), 1183-1192.

[146] Al-Hinai, N. and ElMekkawy, T. Y. (2011). An efficient hybridized genetic algorithm architecture for the flexible job shop scheduling problem. Flexible Services and Manufacturing Journal, 23(1), 64-85.

[147] Hasan, S. K., Sarker, R. and Essam, D. (2011). Genetic algorithm for job-shop scheduling with machine unavailability and breakdowns. International Journal of Production Research, 49(16), 4999-5015.

[148] Ponnambalam, S. G., Ramkumar, V. and Jawahar, N. (2001). A multi-objective genetic algorithm for job-shop scheduling. Production Planning and Control, 12(8), 764-774.

[149] Parsopoulos, K. E. and Vrahatis, M. N. (2002). Particle swarm optimization method in multi-objective problems. In Proceedings of the 2002 ACM Symposium on Applied Computing, Spain, March 2002, 603-607.

[150] Chang, P. C., Hsieh, J. C. and Lin, S. G. (2002). The development of gradual-priority weighting approach for the multi-objective flow-shop scheduling problem. International Journal of Production Economics, 79(3), 171-183.

[151] Coello, C. A. C., Pulido, G. T. and Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. IEEE Transactions on Evolutionary Computation, 8(3), 256-279.

[152] Xia, W. and Wu, Z. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. Computers and Industrial Engineering, 48(2), 409-425.

[153] Suresh, R. K. and Mohanasundaram, K. M. (2006). Pareto archived simulated annealing for job shop scheduling with multiple objectives. International Journal of Advanced Manufacturing Technology, 29(1-2), 184-196.

[154] Lei, D. and Wu, Z. (2006). Crowding-measure-based multi-objective evolutionary algorithm for job shop scheduling. International Journal of Advanced Manufacturing Technology, 30(1-2), 112-117.

[155] Gao, J., Gen, M., Sun, L. and Zhao, X. (2007). A hybrid of genetic algorithm and bottleneck shifting for multi-objective flexible job shop scheduling problems. Computers and Industrial Engineering, 53(1), 149-162.

[156] Ho, N. B. and Tay, J. C. (2008). Solving multiple-objective flexible job shop problems by evolution and local search. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 38(5), 674-685.

[157] Tay, J. C. and Ho, N. B. (2008). Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. Computers and Industrial Engineering, 54(3), 453-473.

[158] Yagmahan, B. and Yenisey, M. M. (2008). Ant colony optimization for multi-objective flow shop scheduling problem. Computers and Industrial Engineering, 54(3), 411-420.

[159] Lei, D. (2008). A Pareto archive particle swarm optimization for multi-objective job shop scheduling. Computers and Industrial Engineering, 54(4), 960-971.

[160] Zhang, G., Shao, X., Li, P. and Gao, L. (2009). An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. Computers and Industrial Engineering, 56(4), 1309-1318.

[161] Wang, X., Gao, L., Zhang, C. and Shao, X. (2010). A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem. International Journal of Advanced Manufacturing Technology, 51(5-8), 757-767.

[162] Ruiz, R. and Vazquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. European Journal of Operational Research, 205(1), 1-18.

[163] Jin, Z. H., Ohno, K., Ito, T. and Elmaghraby, S. E. (2002). Scheduling hybrid flow-shops in printed circuit board assembly lines. Production and Operations Management, 11(2), 216-230.

[164] Kyparisis, G. J. and Koulamas, C. (2006). A note on makespan minimization in two-stage flexible flow shops with uniform machines. European Journal of Operational Research, 175(2), 1321-1327.

[165] Wittrock, R. J. (1988). An adaptable scheduling algorithm for flexible flow lines. Operations Research, 36(3), 445-453.

[166] Grabowski, J. and Pempera, J. (2000). Sequencing of jobs in some production system. European Journal of Operational Research, 125(3), 535-550.

[167] Guinet, A. (1991). Textile production systems: a succession of non-identical parallel processor shops. Journal of the Operational Research Society, 42(8), 655-671.

[168] Tsubone, H., Ohba, M., Takamuki, H. and Miyake, Y. (1993). A production scheduling system for a hybrid flow shop - A case study. Omega, 21(2), 205-214.

[169] Aghezzaf, E. H. and Van Landeghem, H. (2002). An integrated model for inventory and production planning in a two-stage hybrid production system. International Journal of Production Research, 40(17), 4323-4339.

[170] Yang, T., Kuo, Y. and Chang, I. (2004). Tabu-search simulation optimization approach for flow-shop scheduling with multiple processors - a case study. International Journal of Production Research, 42(19), 4015-4030.

[171] Lin, H. T. and Liao, C. J. (2003). A case study in a two-stage hybrid flow shop with setup time and dedicated machines. International Journal of Production Economics, 86(2), 133-143.

[172] Deal, D. E., Yang, T. and Hallquist, S. (1994). Job scheduling in petrochemical production: two-stage processing with finite intermediate storage. Computers and Chemical Engineering, 18(4), 333-344.

[173] Bertel, S. and Billaut, J. C. (2004). A genetic algorithm for an industrial multiprocessor flow shop scheduling problem with recirculation. European Journal of Operational Research, 159(3), 651-662.

[174] Adler, L., Fraiman, N., Kobacker, E., Pinedo, M., Plotnicoff, J. C. and Wu, T. P. (1993). BPSS: a scheduling support system for the packaging industry. Operations Research, 41(4), 641-648.

[175] Dror, M. and Mullaseril, P. A. (1996). Three stage generalized flow-shop: Scheduling civil engineering projects. Journal of Global Optimization, 9(3-4), 321-344.

[176] Allahverdi, A. and Al-Anzi, F. S. (2006). Scheduling multi-stage parallel-processor services to minimize average response time. Journal of the Operational Research Society, 57(1), 101-110.

[177] Chen, L., Bostel, N., Dejax, P., Cai, J. and Xi, L. (2007). A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal. European Journal of Operational Research, 181(1), 40-58.

[178] Ziaeifar, A., Tavakkoli-Moghaddam, R. and Pichka, K. (2012). Solving a new mathematical model for a hybrid flow shop scheduling problem with a processor assignment by a genetic algorithm. International Journal of Advanced Manufacturing Technology, 61(1-4), 339-349.

[179] Engin, O., Ceran, G. and Yilmaz, M. K. (2011). An efficient genetic algorithm for hybrid flow shop scheduling with multiprocessor task problems. Applied Soft Computing, 11(3), 3056-3065.

[180] Sun, J., Liu, J. and Xu, W.B. (2007) 'Using quantum-behaved particle swarm optimization algorithm to solve non-linear programming problems', International Journal of Computer Mathematics,84(2),261-272.

[181] Clerc, M., and Kennedy, J.(2002) The particle swarm: explosion, stability and convergence in a multi-dimensional complex space. IEEE Transactions on Evolutionary Computation, 6 (1), 58-73.

[182] Caponetto, R., Fortuna, L., Fazzino, S. and Xibilia, M. G. (2003). Chaotic sequences to improve the performance of evolutionary algorithms. , IEEE Transactions on Evolutionary Computation, 7(3), 289-304.

[183] Xiang, T., Liao, X. and Wong, K. W. (2007). An improved particle swarm optimization algorithm combined with piecewise linear chaotic map. Applied Mathematics and Computation, 190(2), 1637-1645.

[184] Prakash, A., Khilwani, N., Tiwari, M. K. and Cohen, Y. (2008). Modified immune algorithm for job selection and operation allocation problem in flexible manufacturing systems. Advances in Engineering Software, 39(3), 219-232.

[185] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In Proceedings of IEEE International Conference on Neural Network, Washington, USA, Nov/Dec 1995, 4, 1942-1948.

[186] Yao, X., Liu, Y. and Lin, G. (1999). Evolutionary programming made faster. IEEE Transactions on Evolutionary Computation, 3(2), 82-102.

[187] Wong, K. W., Man, K. P., Li, S. and Liao, X. (2005). A more secure chaotic cryptographic scheme based on the dynamic look-up table. Circuits, Systems and Signal Processing, 24(5), 571-584.

[188] Song, Y., Chen, Z. and Yuan, Z. (2007). New chaotic PSO-based neural network predictive control for nonlinear process. IEEE Transactions on Neural Networks, 18(2), 595-601.

[189] Pareek, N. K., Patidar, V. and Sud, K. K. (2006). Image encryption using chaotic logistic map. Image and Vision Computing, 24(9), 926-934

[190] Alatas, B. (2010). Chaotic bee colony algorithms for global numerical optimization. Expert Systems with Applications, 37(8), 5682-5687.

[191] Coelho, L. D. S. (2008). A quantum particle swarm optimizer with chaotic mutation operator. Chaos, Solitons and Fractals, 37(5), 1409-1418.

[192] Cai, J., Ma, X., Li, L. and Haipeng, P. (2007). Chaotic particle swarm optimization for economic dispatch considering the generator constraints. Energy Conversion and Management, 48(2), 645-653

[193] Hong, W. C. (2009). Chaotic particle swarm optimization algorithm in a support vector regression electric load forecasting model. Energy Conversion and Management, 50(1), 105-117.

[194] Alatas, B., Akin, E. and Ozer, A. B. (2009). Chaos embedded particle swarm optimization algorithms. Chaos, Solitons and Fractals, 40(4), 1715-1734.

[195] Wang, H., Sun, H., Li, C., Rahnamayan, S. and Pan, J. S. (2013). Diversity enhanced particle swarm optimization with neighborhood search. Information Sciences, 223, 119-135.

[196] Carlier, J. and Néron, E. (2000). An exact method for solving the multi-processor flow-shop. RAIRO-Operations Research, 34(01), 1-25.

[197] Artigues, C. and Feillet, D. (2008). A branch and bound method for the job-shop problem with sequence-dependent setup times. Annals of Operations Research, 159(1), 135-159.

[198] Lorigeon, T., Billaut, J. C. and Bouquard, J. L. (2002). A dynamic programming algorithm for scheduling jobs in a two-machine open shop with an availability constraint. Journal of the Operational Research Society, 53(11), 1239-1246.

[199] Potts, C. N. and Van Wassenhove, L. N. (1987). Dynamic programming and decomposition approaches for the single machine total tardiness problem. European Journal of Operational Research, 32(3), 405-414.

[200] Wu, W., Cai, H. and Jiang, L. (2009). Integrated genetic algorithm for flexible job-shop scheduling problem. Computer Engineering and Applications, 45 (22), 183-86.

[201] Stoop, P. P. and Wiers, V. C. (1996). The complexity of scheduling in practice. International Journal of Operations and Production Management, 16(10), 37-53.

[202] Suresh, V. and Chaudhuri, D. (1993). Dynamic scheduling: a survey of research. International Journal of Production Economics, 32(1), 53-63.

[203] Xiong, J., Xing, L. N. and Chen, Y. W. (2013). Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns. International Journal of Production Economics, 141(1), 112-126.

[204] Horng, S. C., Lin, S. S. and Yang, F. Y. (2012). Evolutionary algorithm for stochastic job shop scheduling with random processing time. Expert Systems with Applications, 39(3), 3603-3610.

[205] He, W. and Sun, D. H. (2013). Scheduling flexible job shop problem subject to machine breakdown with route changing and right-shift strategies. International Journal of Advanced Manufacturing Technology, 66(1-4), 501-514.

[206] Li, J. Q., Pan, Q. K. and Liang, Y. C. (2010). An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. Computers and Industrial Engineering, 59(4), 647-662.

[207] Frutos, M., Olivera, A. C. and Tohmé, F. (2010). A memetic algorithm based on a NSGAII scheme for the flexible job-shop scheduling problem. Annals of Operations Research, 181(1), 745-765.

[208] Moslehi, G. and Mahnam, M. (2011). A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. International Journal of Production Economics, 129(1), 14-22.

[209] Li, J. Q., Pan, Q. K. and Gao, K. Z. (2011). Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems. International Journal of Advanced Manufacturing Technology, 55(9-12), 1159-1169.

[210] Rabiee, M., Zandieh, M. and Ramezani, P. (2012). Bi-objective partial flexible job shop scheduling problem: NSGA-II, NRGA, MOGA and PAES approaches. International Journal of Production Research, 50(24), 7327-7342.

[211] Li, J. Q., Pan, Q. K. and Chen, J. (2012). A hybrid Pareto-based local search algorithm for multi-objective flexible job shop scheduling problems. International Journal of Production Research, 50(4), 1063-1078.

[212] Wang, Y. M. (1998). Using the method of maximizing deviations to make decision for multi-indices. System Engineering and Electronics, 7, 24-26.

[213] Mostaghim, S. and Teich, J. (2003). Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO). In Proceedings of the IEEE Swarm Intelligence Symposium, Indiana, USA, April 2003, 26-33.

[214] Wang, L. and Singh, C. (2007). Environmental/economic power dispatch using a fuzzified multi-objective particle swarm optimization algorithm. Electric Power Systems Research, 77(12), 1654-1664.

[215] Raquel, C. R. and Naval Jr, P. C. (2005). An effective use of crowding distance in multi-objective particle swarm optimization. In Proceedings of the Genetic and Evolutionary Computation Conference, Washington, USA, June 2005, 257-264

[216] Reyes-Sierra, M. and Coello, C. C. (2006). Multi-objective particle swarm optimizers: A survey of the state-of-the-art. International Journal of Computational Intelligence Research, 2(3), 287-308.

[217] Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2), 182-197.

[218] Ebrahimipour, V., Haeri, A., Sheikhalishahi, M. and Asadzadeh, S.M. (2012). Application of multi-objective particle swarm optimization to solve a fuzzy multi-objective reliability redundancy allocation problem. Journal of Safety Engineering, 1(2), 26-38.

[219] Basu, M. (2008). Dynamic economic emission dispatch using non-dominated sorting genetic algorithm-II. International Journal of Electrical Power and Energy Systems, 30(2), 140-149.

[220] Ghiasi, H., Pasini, D. and Lessard, L. (2011). A non-dominated sorting hybrid algorithm for multi-objective optimization of engineering problems. Engineering Optimization, 43(1), 39-59.

[221] Bandyopadhyay, S. and Bhattacharya, R. (2013). Solving multi-objective parallel machine scheduling problem by a modified NSGA-II. Applied Mathematical Modeling, 37(10), 6718-6729.

[222] Karimi, N., Zandieh, M. and Karamooz, H. R. (2010). Bi-objective group scheduling in hybrid flexible flow-shop: a multi-phase approach. Expert Systems with Applications, 37(6), 4024-4032

Appendix A1

List of publications

List of Publications in International Journals

1. **Singh, M. R.** and Mahapatra, S. S. (2012). A swarm optimization approach for flexible flow shop scheduling with multiprocessor tasks. International Journal of Advanced Manufacturing Technology, 62(1-4), 267-277.
2. **Singh, M. R**., Mahapatra, S. S. and Mishra, K. (2013). A novel swarm optimizer for flexible flow shop scheduling. International Journal of Swarm Intelligence, 1(1), 51-69.

Papers Accepted for Publication

3. **Singh, M. R**., Mahapatra, S. S. and Mishra, R.K. (2013). Robust scheduling for flexible job shop problems with random machine breakdowns using a quantum behaved particle swarm optimization. International Journal of Services and Operations Management, (In Press)

Communicated Papers

4. **Singh, M. R**., Singh, M., Mahapatra, S. S. and Jagedev, N. (2014). Particle swarm optimization algorithm embedded with maximum deviation theory for solving multi-objective flexible job shop scheduling problem. Optimization and Engineering (Springer).(under review)
5. Bathrinath, S., Saravanasankar, S., Mahapatra, S. S., **Singh, M. R.** and Ponnambalame, S.G. (2014). An improved meta-heuristic approach for solving identical parallel processor scheduling problem. Part B: Journal of Engineering Manufacture (Sage).(under review)
6. **Singh, M. R.** and Mahapatra, S. S. (2013). A quantum behaved particle swarm optimization for flexible job shop scheduling. Computers and Industrial Engineering (Elsevier).(under review)

International Conferences

1. **Singh, M. R.** and Mahapatra, S. S. (2013). A quantum particle swarm optimizer for multi-objective flexible flow shop scheduling problem. International Conference on Industrial Engineering (ICIE 2013), at S.V. National Institute of Technology, Surat, during 20th to 22th November 2013.
2. **Singh, M. R.** and Mahapatra, S. S. (2013). A quantum behaved particle swarm optimization for flexible job shop scheduling with random machine breakdowns.

International Conference on Smart Technologies for Mechanical Engineering (STME-2013), at Delhi Technological University, New Delhi, during 25th to 26th October 2013.

3. **Singh, M. R.** and Mahapatra, S. S. (2013). A quantum particle swarm optimizer with chaotic mutation operator for flexible flow shop scheduling. International Conference on Advanced Manufacturing and Automation (INCAMA 2013), at Kalasalingam University, Tamil Nadu, during 28th-30th March, 2013.