# Optimization of Makespan for Flexible Job Shop Scheduling Problems using Genetic Algorithms

**By**

**Muhammad Kamal Amjad**

**(Registration Number NUST201490190PSMME2614F)**

**Thesis Supervisor**

**Prof Dr Shahid Ikramullah Butt**

**Department of Design and Manufacturing Engineering**
**School of Mechanical and Manufacturing Engineering**
**National University of Sciences and Technology (NUST)**
**Islamabad, Pakistan**
**(2021)**

# Optimization of Makespan for Flexible Job Shop Scheduling Problems using Hybrid Genetic Algorithms


**By**

**Muhammad Kamal Amjad**

**(Registration Number NUST201490190PSMME2614F)**

A thesis submitted to the National University of Sciences and Technology, Islamabad in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in

Design and Manufacturing Engineering

**Thesis Supervisor**

**Prof Dr Shahid Ikramullah Butt**

**Department of Design and Manufacturing Engineering**
**School of Mechanical and Manufacturing Engineering**
**National University of Sciences and Technology (NUST)**
**Islamabad, Pakistan**
**(2021)**

# Thesis Acceptance Certificate

This is to certify that final copy of PhD thesis written by **Muhammad Kamal Amjad**, Registration No. **NUST201490190PSMME2614F** of **School of Mechanical and Manufacturing Engineering (SMME)** has been vetted by undersigned, found complete in all aspects as per NUST Statutes/ Regulations/ PhD Policy, is free of plagiarism, errors, and mistakes and is accepted as partial fulfillment for award of PhD Degree. It is further certified that necessary amendments as pointed out by GEC members and foreign/local evaluators of the scholar have also been incorporated in the said thesis.

Signature: _____

**Name of Supervisor**: Prof Dr Shahid Ikramullah Butt

Date: _____

**Signature (HOD): _____**

Date: _____

## Countersigned by

**Signature (Principal/ Dean): _____**

Date: _____

**Annex L**
Form PhD-7
DOCTORAL PROGRAM
OF STUDY
(Must be type written)

# National University of Sciences & Technology, Islamabad

## REPORT OF DOCTORAL THESIS DEFENCE

We hereby recommend that the student: **Muhammad Kamal Amjad**, Regn No.: **NUST201490190PSMME2614F** may be accepted for Doctor of Philosophy Degree.

### DOCTORAL DEFENSE COMMITTEE

Doctoral Defense Held on _____

GEC Member 1: Prof Dr Riaz Ahmad Signature: _____

GEC Member 2: Dr Mushtaq Khan Signature: _____

GEC Member 3 (External): Prof Dr Mujtaba Hassan Agha Signature: _____

Supervisor: Prof Dr Shahid Ikramullah Butt Signature: _____

External Evaluator 1: Dr Zareena Kausar Signature: _____
(Local Expert)

External Evaluator 2: Dr Ghulam Hussain Signature: _____
(Local Expert)

External Evaluator 3: Dr Gong Lin Signature: _____
(Foreign Expert*)

External Evaluator 4: Dr Muhammad Fahad Signature: _____
(Foreign Expert*)

### COUNTERSIGNED

Dated: _____ _____
Dean/Commandant/Principal

Distribution: 1 x copy each for Director PGP, Registrar Directorate (Examination Branch), Director Research, Director Academics at Main Office, NUST, HoD, Supervisor, Co-Supervisor (if appointed), one for student's dossier at the Institution and copy each for members of GEC.

Note: * Decision of External Evaluators (Foreign Experts) will be sought through video conference, if possible, on the same date and their decision will be intimated (on paper) to Main Office, NUST at a later date.

# Certificate of Approval

This is to certify that the research work presented in this thesis entitled "**Optimization of Makespan for Flexible Job Shop Scheduling Problems using Genetic Algorithms**" was conducted by **Muhammad Kamal Amjad** under the supervision of **Prof Dr Shahid Ikramullah Butt.**

No part of this thesis has been submitted anywhere else for any degree. This thesis is submitted to the **School of Mechanical and Manufacturing Engineering** in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the field of **Design and Manufacturing Engineering**, Department of **Design and Manufacturing Engineering, School of Mechanical and Manufacturing Engineering, National University of Sciences and Technology, Islamabad, Pakistan**.

Student Name: **Muhammad Kamal Amjad** Signature: _____

Examination Committee:

a) External Examiner 1

Prof Dr Ghulam Hussain Signature: _____
Department of Mechanical Engineering,
Ghulam Ishaq Khan Institute of Engineering Science and Technology,
Topi, KPK, Pakistan

b) External Examiner 2:

Dr Zareena Kausar Signature: _____
Department of Mechatronics Engineering,
Air University,
Islamabad, Pakistan

c) Internal Examiner:

Prof Dr Riaz Ahmad Signature: _____
Directorate of Quality Assurance,
National University of Sciences and Technology,
Islamabad, Pakistan

Supervisor Name: **Prof Dr Shahid Ikramullah Butt** Signature: _____

Name of Dean/ HoD: **Prof Dr Javed Iqbal** Signature: _____

# Author's Declaration

I, Muhamad Kamal Amjad hereby state that my PhD thesis titled **"Optimization of Makespan for Flexible Job Shop Scheduling Problems using Genetic Algorithms"** is my own work and has not been submitted previously by me for taking any degree from "National University of Sciences and Technology (NUST)" or anywhere else in the country/worldwide.

At any time if my statement is found to be incorrect even after my graduation, the university has the right to withdraw my PhD degree.

Name of Student/ Author: **Muhamad Kamal Amjad**

Signature: _____

Date: _____

# Plagiarism Undertaking

I, Muhammad Kamal Amjad, solemnly declare that research work presented in the PhD thesis titled **"Optimization of Makespan for Flexible Job Shop Scheduling Problems using Genetic Algorithms"** is solely my research work with no significant contribution from any other person. Small contribution / help wherever taken has been duly acknowledged and that complete thesis has been written by me.

I understand the zero-tolerance policy of the HEC and National University of Sciences and Technology (NUST) towards plagiarism. Therefore, I as an Author of the above titled thesis declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred / cited.

I undertake that if I found guilty of any formal plagiarism in the above titled thesis even after award of PhD Degree, the University reserves the right to withdraw/revoke my PhD Degree and that HEC and the University has the right to publish my name on the HEC / University website in which the names of the students are placed who submitted plagiarized thesis.

Student/ Author Signature: _____

Name of Student: **Muhammad Kamal Amjad**

# Acknowledgement

First of all, I am thankful to Allah, Who is the source of all knowledge in this world. Indeed, He has taught what all the mankind knows.

I am deeply grateful to my supervisor **Prof Dr Shahid Ikramullah Butt** for his sincere guidance, suggestions and supervision. He has supported me through all the highs and lows of my research tenure and helped me manging my research work along with official commitments. I would also like to thank my GEC **Prof Dr Riaz Ahmad**, **Dr Mushtaq Khan** and **Prof Dr Mujtaba Hassan Agha** for their valuable inputs in conception, conducting the experimentation and supervision at various stages of the research. They helped and guided me sail through the hard times. Special thanks are due to **Prof Dr Imram Ali Chaudhary** for his thorough review and direction during my publications and thesis writing all the way from Kingdom of Saudi Arabia.

I am also gratified to my colleague and one of my very best friends, **Naveed Anjum** for his devotion and all out help during the coding process. I am also thankful to **Dr Umer Asgher** and **Dr Salman Sagheer Warsi** (my PhD fellow colleagues), who have been a source of motivation and guidance and my companion throughout this journey.

Finally, I am indebted to my father **M Amjad Amin** and mother **Meher Un Nisa** who believed in me, built my dream about doctorate and supported through all these years to complete my work. They also supported me out of the way through this task along with my professional responsibilities of job and took extra care of everything throughout these years. Special thanks are also due to my wife **Dr Rubeena Kamal** and children (**M Ahmad Kamal** and **Fatimah Kamal**) for providing me the peace of mind and time to pursue this milestone in my life.

**Muhammad Kamal Amjad**

**December 2021**

*To my parents, wife, and children*

# Abstract

Manufacturing scheduling is one of the most researched areas since its optimality plays an important role in the operation of the shop floor. Manufacturing has a vital contribution in the overall economy of a country as it generates and attracts commercial activities. The whole framework of business has changed in view of the fluctuating global customer demands and fierce opposition from technologically advanced competitors. There is always a pressure on the manufacturer to produce the designed products in the shortest possible time to capture the market. To the challenge of changing product requirements and market demands, flexible manufacturing system is the answer.

Flexible job shop is employed to produce a medium variety of products in a medium volume category. In contrast to the conventional job shop, it offers flexibility in performing operations on different machines; hence providing space for the manufacturing planner / scheduler for arranging parts as per corporate requirements. When seen in the context of optimal operation, this setting while offering such great advantage, also poses the scheduler with the decision regarding assignment of operations to available machines in addition to sequencing of operations. In this way, the complexity of the problem grows exponentially even in the small settings of the shop.

The flexible job shop scheduling is a NP-hard combinatorial optimization problem with regards to complexity and its exact solution requires many lifetimes to reach. Consequently, techniques built around the concepts of artificial intelligence have been popularly used to solve the problem. Genetic Algorithm (GA) is one of the most attempted and widespread technique from this domain. GA can produce good results of the scheduling problems, however when stuck in the local minima, the algorithm normally fails to escape, and solution quality is badly affected.

In this research work, problem is formulated mathematically and insights to a selected benchmark is provided. Problem complexity is then evaluated in a quantitative way through estimation of search space of the selected datasets and an understanding to the actual area of search is developed.

Priority rules are then integrated with the GA (GA-PR) to solve the FJSSP. In this regard, competitive modification in the rule has been proposed in addition to the integration scheme. The algorithm is also equipped with adaptive operators which also contribute to its performance. In addition to this a standalone pure GA (GA-IDT) is also proposed to efficiently solve the target problem. An iterative diversification technique is embedded into the proposed algorithm which proficiently manages the intensification and diversification of the population.

The efficacy of both algorithms is tested against standard benchmark problems and it is concluded that proposed techniques are competitive with other concepts in literature.

***Keywords:*** Genetic algorithm; Flexible job shop scheduling problem; Iterative diversification technique; combinatorial optimization.

# List of Publications

1. M. K. Amjad, S. I. Butt, R. Kousar, R. Ahmad, M. H. Agha, Z. Faping, et al., "Recent Research Trends in Genetic Algorithm Based Flexible Job Shop Scheduling Problems," *Mathematical Problems in Engineering*, vol. 2018, p. 32, 2018. **[IF: 1.009, HJRS Cat W]**

2. M. K. Amjad, S. I. Butt, N. Anjum, I. A. Chaudhry, Z. Faping, and M. Khan, "A layered genetic algorithm with iterative diversification for optimization of flexible job shop scheduling problems," *Advances in Production Engineering & Management*, vol. 15, pp. 377-389, 2020. **[IF: 2.347, HJRS Cat W]**

3. M. K. Amjad, S. I. Butt, N. Anjum, "Improved Genetic Algorithm Integrated with Scheduling Rules for Flexible Job Shop Scheduling Problems," in *5th International Conference on Power, Energy and Mechanical Engineering*, Shinghai, China, 2021. **[Scopus Indexed]**

# List of Abbreviations

| Abbreviation | Description |
|---|---|
| AGVs | Automated Guided Vehicles |
| AIA | Artificial Immune Algorithm |
| Avg | Average |
| CM | Compulsory mutation |
| CP | Constraint Programming |
| FJS | Flexible Job Shop |
| FJSSP | Flexible Job Shop Scheduling Problem |
| FMC | Flexible Manufacturing Cell |
| FMS | Flexible Manufacturing System |
| GA | Genetic Algorithm |
| GA-IDT | Genetic Algorithm - Iterative Diversification Technique |
| GA-PR | Genetic Algorithm - Priority Rules |
| GB | Giga Bytes |
| GDP | Gross Domestic Product |
| GS | Global selection |
| GT | Group Technology |
| JSSP | Job Shop Scheduling Problem |
| LB | Lower bound |
| LPT | Longest processing time |
| LS | Local selection |
| M | Set of machines |
| Max | Maximum |
| MFJS | Medium Flexible Job Shop |
| MILP | Mixed Integer Linear Programming |
| Min | Minimum |
| mMWR | Modified most work remaining |
| MOR | Most operations remaining |
| MS | Machine selection |

| | |
|---|---|
| MWR | Most work remaining |
| NP | Non polynomial |
| NSGA | Nondominated Sorting Genetic Algorithm |
| OS | Operation selection |
| PC | Percentage contribution |
| P-FJSSP | Partial FJSSP |
| PopSize | Population size |
| POX | Precedence preserving order-based crossover |
| PSO | Particle Swarm Optimization |
| RAM | Random Access Memory |
| RIM | Random intelligent mutation |
| RS | Random selection |
| RW | Roulette wheel |
| SA | Simulated Annealing |
| SFJS | Small Flexible Job Shop |
| SM | Swap Mutation |
| SPT | Shortest processing time |
| SPX | Single-point crossover |
| SS | Search space |
| T-FJSSP | Total FJSSP |
| Tk | Machine busy time |
| TPX | Two-point crossover |
| TS | Tabu Search |
| UX | Universal crossover |
| VNS | Variable Neighborhood Search |

# List of Symbols

| Symbol | Description |
|---|---|
| $C_{max}$ | Makespan |
| $E_{ijk}$ | End time |
| J | Set of jobs |
| $J_{io}$ | Total number of operations |
| L | Total number of sequences |
| m | Number of machines |
| M | Set of machines |
| n | Number of jobs |
| O | Operation |
| $O_{ij}$ | $j^{th}$ operation of $i^{th}$ job |
| OS | Operation selection |
| $P_c$ | Probability of crossover |
| $P_{ijk}$ | Processing time |
| $P_m$ | Probability of mutation |
| PopSize | Population size |
| $r_{ijk}$ | Release time |
| SelRatio | Selection Ratio |
| $t_{ijk}$ | Start time |
| $T_k$ | Machine busy time |
| $U_{ij}$ | Flexibility index |
| α | Machine characteristics |
| β | Job characteristics |
| γ | Cost Function |
| $\Omega_{ij}$ | Actual identification of machines available to undertake operation |

# Table of Contents

## List of Figures

## List of Tables

# 1 Chapter 1 - Introduction

## 1.1 Background

Since the beginning of intellectual evolution of mankind, efforts are being made to improve the lifestyle and living standard. The earliest documented human life on earth is divided into stone age, bronze age and iron age [1] which depict this continual struggle. As a result, humanity has seen marvelous growth in each race of everyday life. With the increase in population and the advent of technology, mankind developed machines to ease the availability and accessibility of daily used products. The need of machines for completing the repetitive and laborious work raised and eventually, the industrial age (1760 – 1970) saw unprecedented increase in the manufacturing sector. Today, one cannot imagine daily life without the presence of machines.

As the humans saw the advancement in technology, the need for luxury and comfort increased. In addition, owing to the increase in population, the requirement of necessities was also increased. The amalgamated cultural and occupational requirements generated various types of product requirements. All products to be developed are to be manufactured, hence products and manufacturing go hand-in-hand. However, since products differ in nature according to their use, different machines are required for their manufacturing in order to optimize the production and cost effectiveness. Here comes the balance between the number of machines and number of products to be manufactured on these available machines. Obviously, both the industrialist and the customer want to get minimized production time and cost.

The history of manufacturing can be traced back to the stone age whereby basic tools were invented for reshaping the materials recovered from natural environment [2]. The word “manufacture” itself is derived from Latin background which basically means “made by hand” [3]. The manufacturing process converts raw materials in products of use through use of tools and machines [4]. Hence, every product in our surrounding goes through this process during its creation. Therefore, manufacturing sector always faces challenging demands from inside and outside the engineering industry in view of push-pull system of technology change and market requirements [5].

Over the period of time, manufacturing has emerged as a key indicator of a country’s economic and commercial growth. World Bank has reported that manufacturing activity adds up to 18% of the Gross Domestic Product (GDP) of the world [6], thereby creating wealth.

Moreover, the industry gave employment to 23.083% of total employment share and added 2.7% in the annual growth of the world in year 2018 [7].

Since, the manufacturing industry directly affects the product output of any country, its wellbeing guarantees economic plateau and improved standard of living / comfort. The manufacturing sector responds to market and product development needs and provides a concrete business opportunity [8] as shown in Figure 1.1. Consequently, the earlier the company responds to the market needs, the more profits are ensured and hence a competitive advantage is gained since no competitor is available in the market. This fact has risen aggressive competition in all the manufacturing industry owing to the constantly changing customer demand on one hand; while one the other hand, it has provided an excellent opportunity for a well-equipped industry to attain market supremacy. Effective manufacturing management techniques, therefore, warrant reduced product availability time to the market and hence increases market share of the said company in the longer run [9].

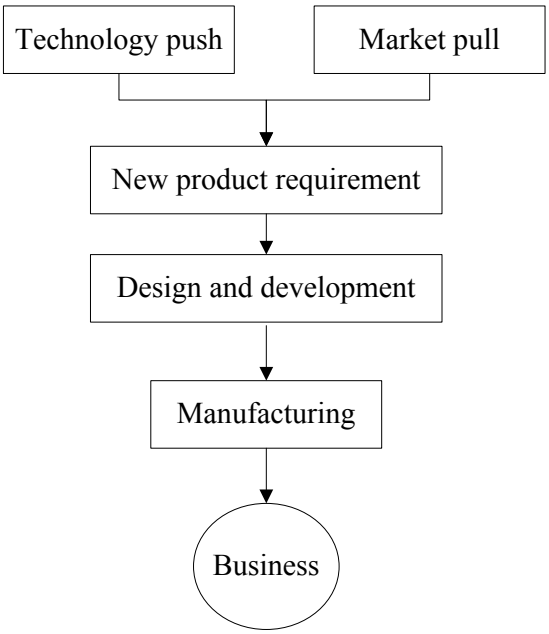


**Figure 1.1: Business opportunity and manufacturing**

Conventional manufacturing operations are carried out in a machine shop where machines are installed in a certain layout. In order to undertake manufacturing in large quantities, production operations are to be planned carefully keeping in view predetermined objectives to ensure that maximum output is achieved with minimum resources. This is important because remaining resources may be assigned to other tasks. Sequencing of operations in a manufacturing

facility directly contributes to its performance. It requires careful production planning to avoid wastage of manhours, materials, cost, and time. Scheduling directly contributes to better operations of a manufacturing facility.

## 1.2 Problem Statement

Manufacturing scheduling deals with planning of operations on machines to obtain intended output with regards to environmental constraints. The problem pertains to the classical scheduling problems generated in the early 20$^{th}$ century and since then enormous amount of research has been carried out to minimize the completion time of process / jobs on the available resources. The minimum completion time will not only utilize the resources in optimal manner but will also release them earlier for future operations with an added advantage of minimized cost over-run and manhours etc.

The advancement in automated manufacturing and production engineering has led to many progressions in the modern manufacturing concern, one of the most important being the Flexible Manufacturing System (FMS). The FMS has been designed to encounter changing demands of different products with regards to operations (e.g., handling, machining operation) and process maximum number of different jobs.

Where the idea feels astounding, it is a challenging task to allocate these advanced resources optimally. Scheduling in the flexible environment is one of the most tough optimization problems and has attracted researchers for over a century. The scheduling field proposes some of the most advanced, complex, and toughest combinatorial optimization problems. The search space of these problems is so huge that one cannot evaluate the complete space even by spending many complete lifetimes. This motivated the current research for actual evaluation of selected benchmark search space. This research is focused on minimization of makespan in a flexible job shop atmosphere. Since exact solution approaches cannot provide solution of these problems in a reasonable time, artificial intelligence techniques have been used as a popular alternative for solving these problems. The current research attempts to solve these problems by use of Genetic Algorithm (GA).

## 1.3 Research objectives

Following are the objectives of this work.

a. To study the GA based FJSSP optimization literature
b. To develop a thorough understanding of scheduling paradigm and its related mathematical intricacies

c. To optimize the processing times of jobs through proposition of efficient techniques
d. To evolve a software-based environment for solving the scheduling problem in an automated manner
e. To develop a heuristic based solution approach
f. To develop a pure GA based approach
g. To evaluate the complexity of scheduling problems on the basis of chromosome representation
h. To solve selected benchmark instances and to identify improvement
i. To identify the advantages of proposed solution architecture and methodologies

## 1.4 Research methodology

This work is organized in three stages as outlined in Figure 1.2. During the preparation phase, initial problem understanding was gathered, and a thorough literature review was conducted to reveal grey areas of the literature. The implementation stage was started with development of an integrated MATLAB-MS Excel based simulation environment to conduct of computational experiments. The phase was designed to achieve following objectives.

a. Development of heuristics-GA based approach
b. Development of a pure GA based approach

The developed algorithms were evaluated thoroughly for correctness through solving small problems available before attempting the large problems. Conceptual or programming bugs were removed through solving the small instances with hand and verification of step-by-step ability of MATLAB bug removal. Moreover, the solution of software generated Gantt charts was also computed and compared with manual solutions for correctness. For awareness of problem complexity, the search space was also quantified. The algorithms were then tested on the selected datasets from literature and different experiments were conducted for the evaluation of proposed improvements. At the end, contributions are summarized, and conclusions are presented along with recommendation for further work. The scope of this work will be restricted to FJSSP.

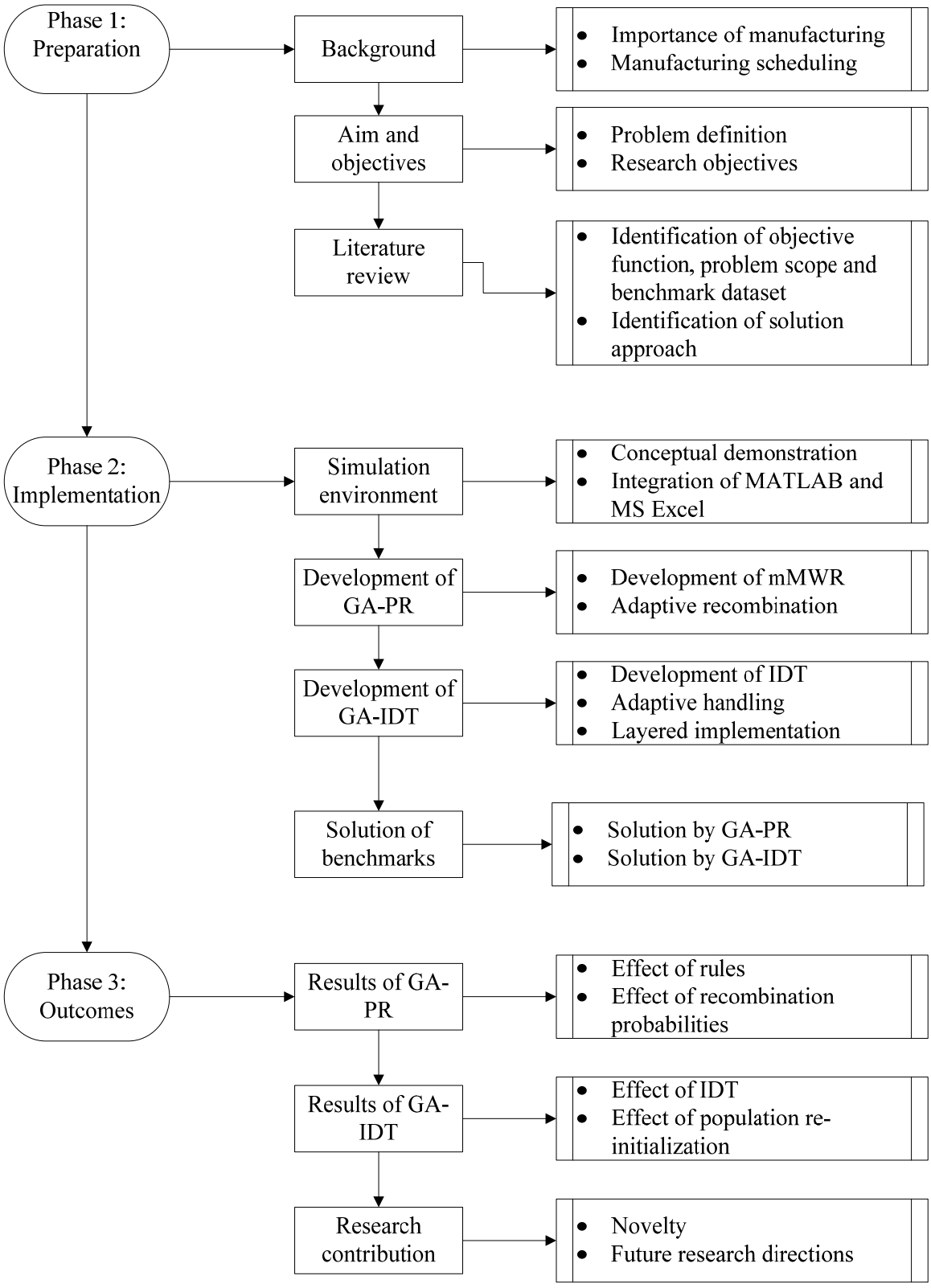


**Figure 1.2: Research methodology**

## 1.5 Organization of thesis

Following is the chapter-wise overview of the current research. Chapter-1 introduces the reader to the background of the research area, the research objectives, and the methodology of the study. Chapter-2 provides literature overview of the research area covering different aspects of the research area. Chapter-3 opens the scheduling problem by performing its mathematical formulation. For a clear understanding, numerical examples are given thoroughly, and each element is explained in a detailed manner. The chapter also covers the design scheme of the proposed solution environment for the modeled problem. Chapter-4 provides detailed explanations of the proposed algorithms for solving the FJSSPs using GA-PR and GA-IDT. Procedures and routines conducted during the algorithms have been illustrated using flowcharts extensively. Meanwhile, improvements in the algorithms have also been highlighted. Chapter-5 deals with the experimental results of the selected benchmarks. Different experiments are conducted to outline the advantages of the proposed techniques and finally comparison has been made with other algorithms to effectively indicate the algorithm performance. Chapter-6 closes the thesis with identification of contributions to the existing knowledge area and future research guidelines.

# 2 Chapter 2 – Literature Review

## 2.1 Introduction

This chapter introduces concept of FMS, machine layouts, classifications and general optimization practice. These topics are necessary to understand and attempt Flexible Job Shops Scheduling Problems (FJSSPs). The literature of FJSSP is thoroughly discussed along with different solution approaches. The chapter has majorly been extracted and modified from the already published review work of the author [10] unless otherwise cited; in which a total of 190 papers have been reviewed.

## 2.2 Flexible Manufacturing Systems

The craft manufacturing techniques (based on skill) has changed to global manufacturing (based on information) in the last century. The manufacturing industry has tailored itself to adapt to the changes and challenges posed by the demanding customer and market needs. Now, a manufacturing enterprise requires concurrent and up-to-date information instead of just skill-based-information. Another huge impact during this change is the advent of automation in place of human operators. Figure 2.1 provides a pictorial layout of the changes that the manufacturing industry has undergone during last decades.

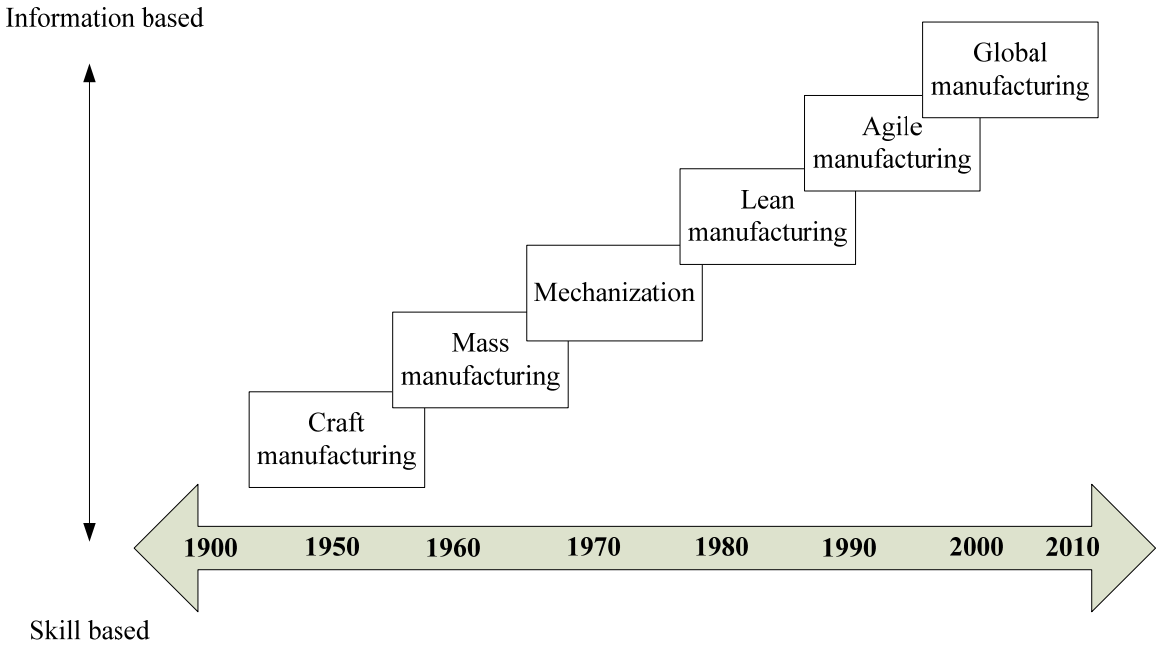


**Figure 2.1: Types of manufacturing in the last century**

Manufacturing systems face the challenge of processing different types of jobs. Since different types of jobs require different machines, increasing the job variety increases machine types and consequently capital investment. Hence, a manufacturing system offers limited number of operations to be processes.

There exists an inverse relation between the volume and variety of products that can be undertaken at a manufacturing concern [3, 11]. The production system can generate a low-volume-high-variety product through a specialized and dedicated machinery to address a specific project need. Such systems are highly specialized and cannot encounter job flexibility. On the other hand, low-variety-high-volume systems produce continuous production in high numbers that mainly constitute the major consumer market products.

Another aspect in this regard is that market share of a company depends upon the time to market for a certain product i.e., the time required to deliver the product to market. Obviously, this entails the efficient development and production of the product. The early the product is made available in the market, the more it gains the market share and customer loyalty accordingly.

Therefore, the relation between the production capacity and production flexibility is inverse in nature and one must keep a balance in both aspects during the design phase of the production system. Since the increment in flexibility entails intermittent process flows and complex / diverse tasks, it cannot be increased infinitely. As a matter of fact, flexibility comes with a penalty of complexity. Moreover, low-volume-low-variety and high-volume-high-variety systems does not lie on the feasible production diagonal as pointed out in Figure 2.2 since they do not pose commercially reasonable. To achieve maximum share of the market, a manufacturing concern must produce a reasonable variety of products with a good volume. This constitutes a medium-volume-medium-flexibility manufacturing setup, commonly known as a Flexible Manufacturing System (FMS).

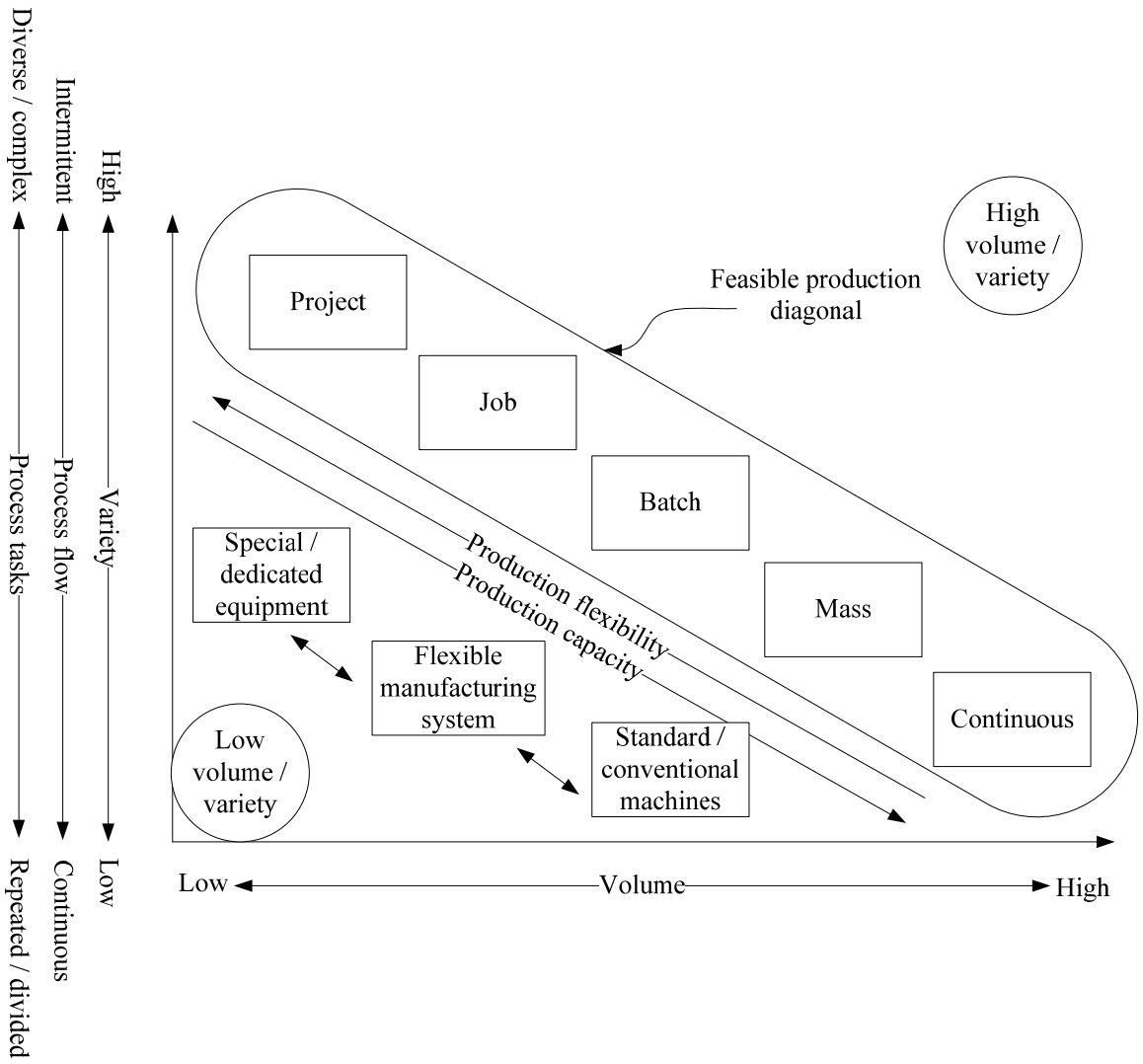


**Figure 2.2: Relation between product volume and variety [3]**

FMS are modern systems which can handle a variety of product types through variable routings [3]. Similar types of operations are grouped together through employment of Group Technology (GT) [12] which generally incorporates a material handling system to produce medium volume and variety products [13].

GT has offered an alternative to the conventional batch production through reduction of downtime for changing jobs [3]. It operates on the concept that different parts contain certain similarities in the form of inherent features. Hence, similar features are processed through same process and tooling. Literature offers several classification schemes for division of parts in GT, e.g., Opitz, CUTPLAN, Brisch, DCLAS and CODE etc. Flexible manufacturing systems include automation in the GT cells.

As described in the section 1.1, customer demands verity of products in a limited time frame. Conventional manufacturing setups give a high production volume but cannot handle different types of products. On the other hands, workshops offer high level of flexibility, but can handle a low production volume. FMS is a trade-off between the conventional high-volume - low-flexibility manufacturing setups and the low-volume-high-flexibility workshops. Major benefits of these systems include; a reduced manufacturing lead time, lower machine requirement, optimal shop floor utilization, medium volume-variety production [3, 14], thereby increasing flexibility in the system [15, 16].

Figure 2.3 presents a typical FMS with Automated Guided Vehicles (AGVs). The FMS is equipped with multiple machines because a certain flexibility has been introduced into the system. Increased number of milling and lathes indicate that this FMS is designed to undertake milling, facing and turning operations more often. The material handling system is used in conjunction with AGVs to load and unload jobs. A computer controller regulates the functioning of the FMS. Similar idea is implemented in the form of Flexible Manufacturing Cell (FMC) where machines are grouped together in order to accommodate a variety of operations. Such cells are designed to undertake similar groups of processes through use of group technology concepts. Different schedules can be executed in the FMS / FMC since it is adaptable to different types and sequences of operations; obviously, until a limitation [17].

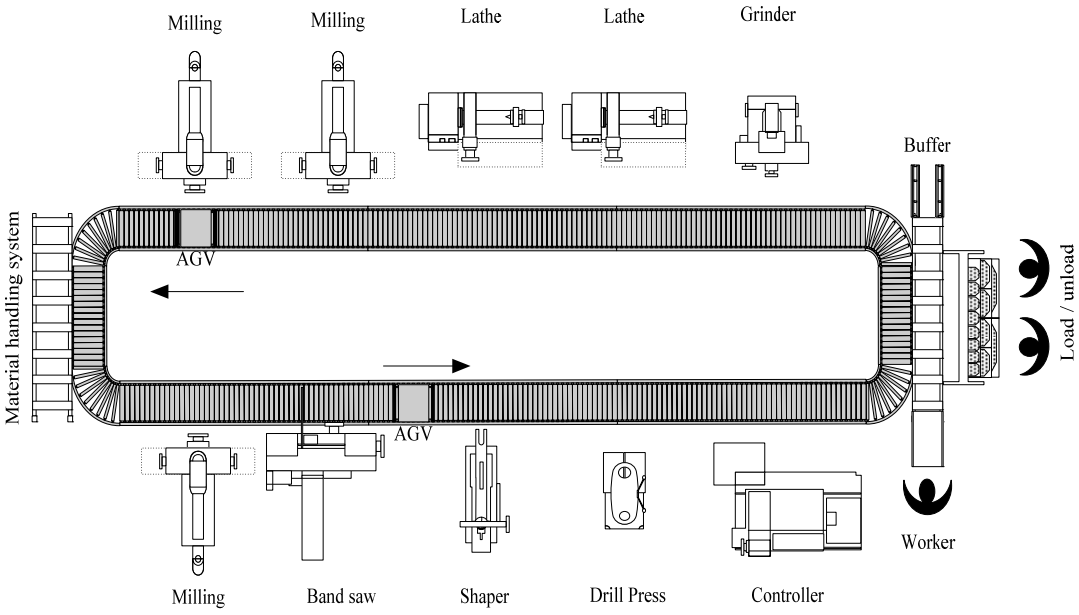


**Figure 2.3: A typical Flexible Manufacturing System**

### 2.2.1 Dimensions of manufacturing flexibility

Flexibility in manufacturing has been described as the ability of a manufacturing system to undertake different tasks of different nature through available resources [15, 18, 19]. It is evident that flexibility eases the operational scenario including many other benefits; however, it adds to the complexity of production management since it requires additional decision making in the process.

The dimensions of flexibility are depicted in Figure 2.4 [20], e.g. flexibility in production means the system can handle variety of parts in production scope. Similarly, flexibility in product means the system can manufacture variety of products. Other dimensions include machine, tooling, and routing flexibility, whereby machines can perform different tasks, can incorporate different tools and can undertake different routing.

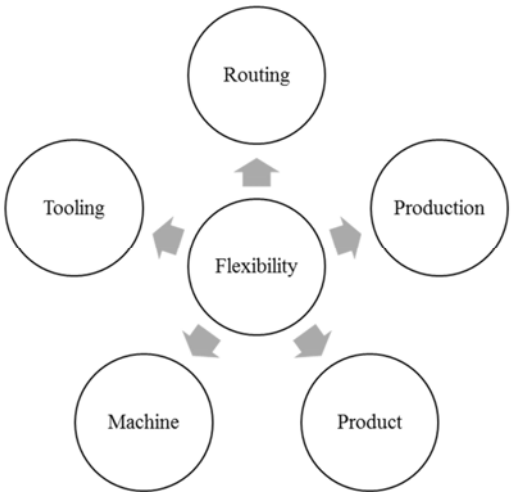


**Figure 2.4: Dimensions of flexibility in a manufacturing environment**

### 2.2.2 The traditional job shop vs flexible job shop

Depending upon the nature of the manufacturing processes and the product to be developed shop layouts have evolved over the period of time. Figure 2.5 presents a limited classification of the shop floor layouts with emphasis on current research problem.

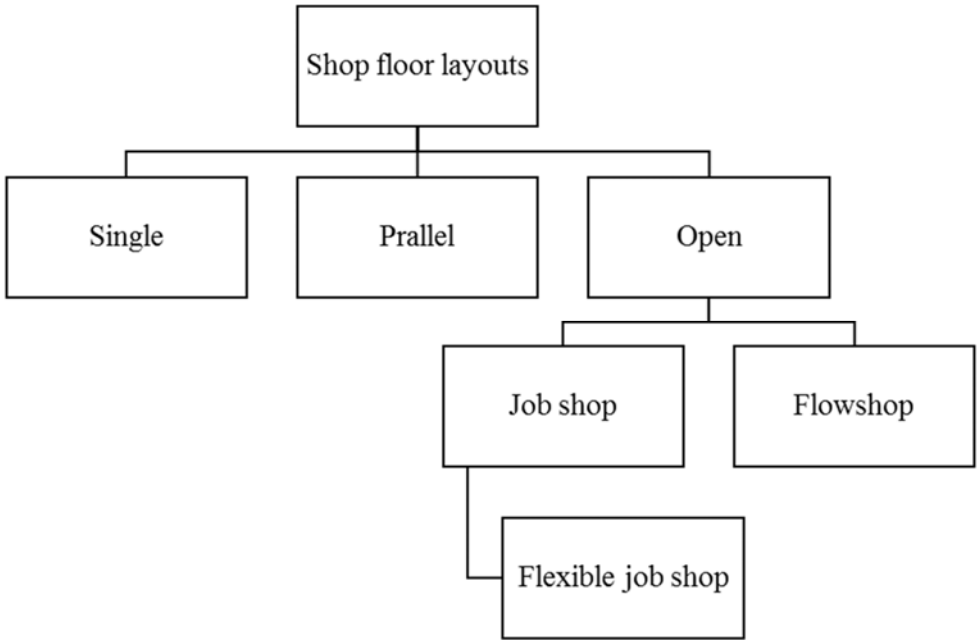


**Figure 2.5: Classification of shop floor layouts**

A typical job shop layout is presented in Figure 2.6. The layout consists of different types of machines, whereby jobs can be processed as per requirement of operations. Different jobs have different sequence of operations; hence this problem deals with routing of each operation of a job on different machines. For example, Job 1 may process from saw-turn-paint-warehouse whereas Job 2 may process as grind-mill-drill-assembly-warehouse. Processing times of each

operation can differ on each machine or it may remain the same. Now, the interest is to sequence the total number of operations of each job on these available machines, since each machine is dedicated for a single operation. Obviously, the scheduler would want to schedule the tasks such that minimum time is consumed to complete all products. This gives rise to the classical Job Shop Scheduling Problem (JSSP). The problem has gained attention since decades, e.g. early attempts to find the schedules date back to early twentieth century [21]. Subsequently, the JSSP has attracted researchers in more recent times e.g., [22-25].

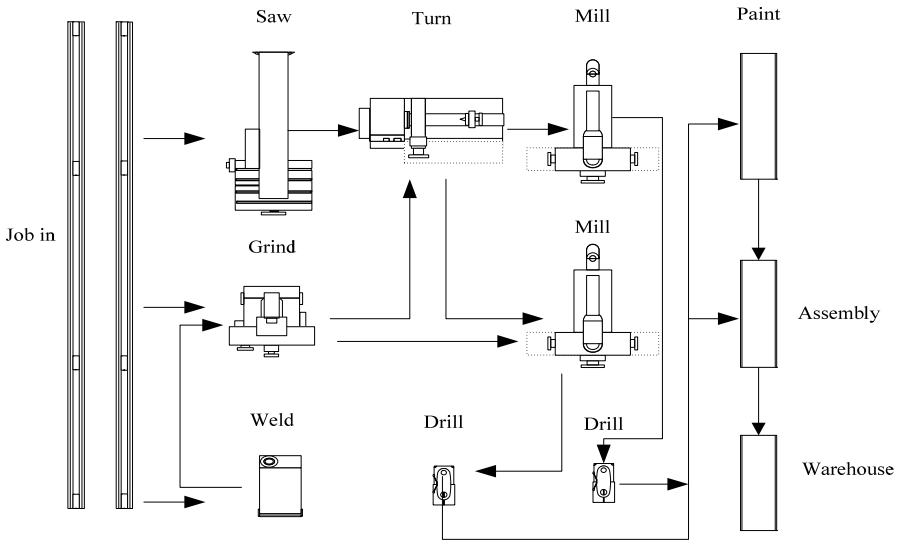


**Figure 2.6: An example of job shop layout**

Hence, the major criterion in JSSP is that every operation must be carried out on a single pre-decided machine. This scenario has been schematically presented in Figure 2.7. Consider a job $J_1$ requiring two operations $O_1$ and $O_2$ for its completion. In a Job Shop Scheduling Problem (JSSP) setting, $O_1$ can only be performed on $M_1$, while $O_2$ can only be performed on $M_2$.

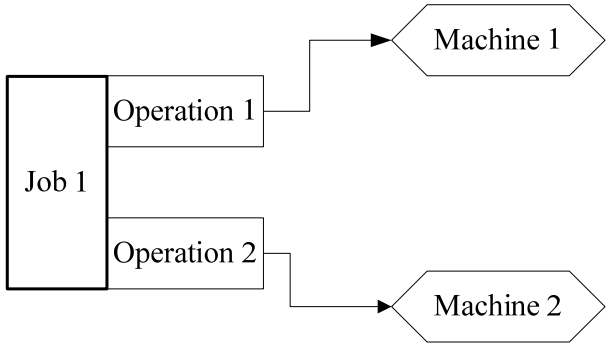


**Figure 2.7: Traditional job shop layout setting**

In actual scenario of the practical shop floors, there are often more than one machines that can perform a single specified task. Therefore, contrary to the scenario of job shop, flexible job shop has inherent flexibility with regards to decision of machine selection form the available pool of machines. This setting provides machines selection problem in addition to the routing problem. A sample FJS schematic is presented in Figure 2.8. Job 1 is to be processed by completing two operations and the shop setting has two available machines. Operation 1 can only be performed on machine 1, while operation 2 has the flexibility to be performed on either of Machine 1 or 2.

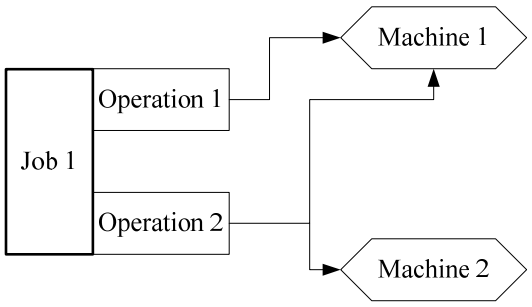


**Figure 2.8: A schematic of Flexible Job Shop**

FJSSP is another branch of the basic JSSP, whereby following two problems are considered simultaneously.

a. Assignment / routing: The decision regarding processing of jobs on available machines.
b. Scheduling / sequencing: The decision regarding the sequence of operations on a said machine.

There exists a built-in flexibility in the FJSSP paradigm. This is due to fact that this setup offers routing and sequencing opportunities to the process planner. Literature has proposed different ways to incorporate flexibility in the manufacturing setup, some of which are summarized in Table 2.1.

**Table 2.1: Some concepts of flexibility in manufacturing**

| Concept | Reference |
|---|---|
| Machines with different tools that can perform multiple operations | [26] |
| Commission more similar machines in case of a bottleneck | [27] |
| One machine for multiple operations | [28] |

## 2.3 Manufacturing Scheduling

Manufacturing scheduling is a decision making problem which involves the sequencing of tasks (i.e. jobs, operations, workers etc.) to available processing resources (machines, work etc.) keeping in view a certain pre-defined objective function in order to complete the tasks [29]. The problem involves sequencing of different jobs, which may contain different operations to different machines. Consequently, manufacturing scheduling becomes a combinatorial optimization problem [30] offering many solutions, only some of which meet the performance requirement, e.g. total completion time of all jobs. Since, infeasible schedules are rejected in a straightforward manner; the feasible schedules are searched for optimality against the objective function. It is evident that all feasible schedules can complete the jobs but will fail to utilize the resources in a good manner as compared to the optimal schedule. As the number of jobs increase, more resources are required which will incur more cost. Hence it is imperative that the interest in finding the optimal schedule grows with the increased number of jobs.

Scheduling decisions are complex in nature since they deal with the problem of resource assignment to the upfront tasking [31]. Generally, the resources are scarce, and tasking becomes larger, so scheduling decisions are to be taken smartly to increase the productivity. Moreover, in view of the various manufacturing sectors involved, companies tend to customize the manufacturing setups, which further increases the complexity of the decision-making problem. In addition, the time available to undertake the scheduling decision is limited because goods are to be produced and forwarded to market. Also, the decision is order-dependent, since production is customer-oriented [32].

### 2.3.1 Classification of scheduling problems

Scheduling can be divided into deterministic and stochastic [31] as shown in Figure 2.9. Deterministic scheduling involves jobs with known processing time and stochastic scheduling refers to real-time changing job processing time which only get confirmed after the job is completely processed [33-35]. Jobs arrive in a known manner in static scheduling, while dynamic scheduling deals with job arrivals at unknown intervals [36-38].

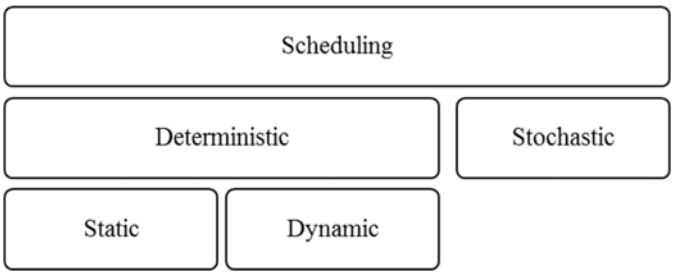


**Figure 2.9: Classification of Scheduling**

Figure 2.10 presents the classification of scheduling problem. Following describes the salient features of each classification.

a. Deterministic: All the variables have known value in the problem e.g., processing time, due date etc.
b. Proactive: This enables the scheduler to undertake unforeseen aspects which can arise because of the stochastic nature of the problem.
c. Real time: This scheme reallocates the resources according to the resources available at the shop floor.
d. Adaptive: The scheduling software changes the scheduling methodology according to the history of the events.
e. Reactive: This embarks on the current state of the shop floor and changes the schedules reactively according to the changing conditions. This scheme aims to change the implemented schedule.
f. Stochastic: Some variable in the problem is not known but is determined through a probability function.
g. Fuzzy: Deals with the fuzzy variables.
h. Robust: The schedule is not affected by the unforeseen events that can occur on the shop floor.

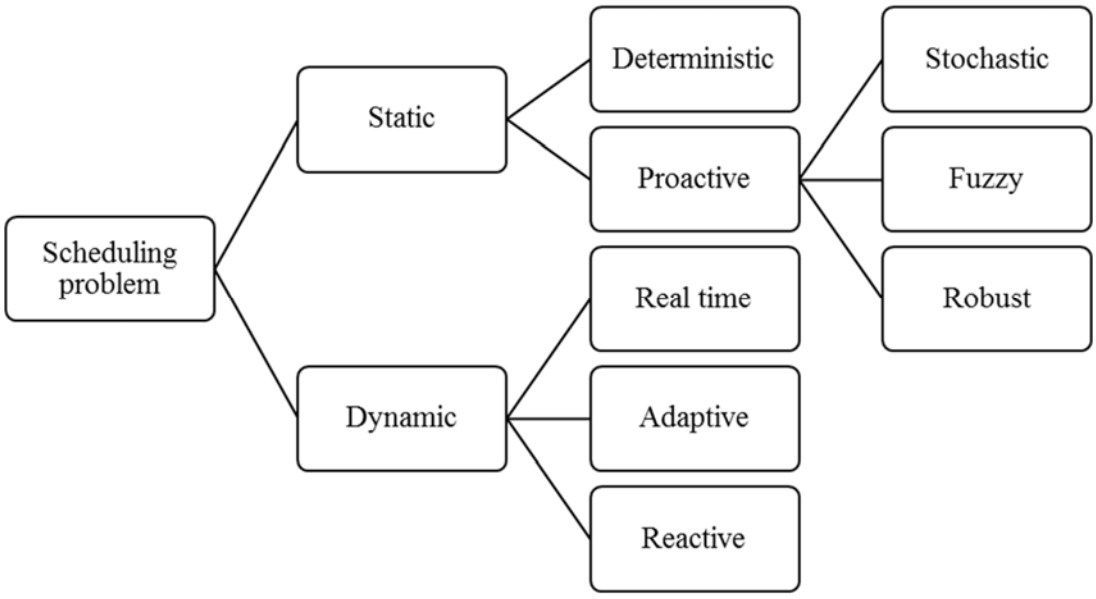


**Figure 2.10: Classification of the scheduling problem**

The scheduling models are conventionally classified through the characteristics of processing layout ($\alpha$), constraints ($\beta$) and cost function ($\gamma$). The notation is summarized as $\alpha \mid \beta \mid \gamma$. This scheme was introduced by Conway et al. [39], which was further extended by Graham et al. [40] and Lawler et al. [41]. The processing layouts are denoted as double index $\alpha_1\alpha_2$ which are further explained in Table 2.2, e.g., $\alpha = J3$ means a job shop layout with 3 machines. Similarly, $\alpha = P$ means a parallel machine setting with no count of machines available. The machine constraints are depicted in Table 2.3

**Table 2.2: Explanation of processing layout ($\alpha$)**

| Notation | Values | Description |
|---|---|---|
| $\alpha_1$ | $\phi$ | Single machine |
| | $P$ | Parallel identical machines |
| | $Q$ | Parallel uniform machines |
| | $R$ | Parallel unrelated machines |
| | $F$ | Flow shop |
| | $J$ | Job shop |
| | $O$ | Open shop |
| $\alpha_2$ | $1, 2, \ldots, m$ | Fixed machines |
| | $\phi$ | No fixed machines |

**Table 2.3: Explanation of constraints ($\beta$)**

| Notation | Description |
|---|---|
| $\beta = prec$ | There exist precedence relations between operations of the jobs, e.g., number of predecessor and successors. |
| $\beta = sd$ | There is sequence dependent setup / removal time |
| $\beta = M_j$ | Machine eligibility i.e., some machines can perform some operations only. |
| $\beta = prmu$ | Permutation flow shop problem. |
| $\beta = brkdwn$ | There can be possibility of breakdown in machines. |
| $\beta = recrc$ | The process contains recirculation i.e., at least one job visits one machine more than once. |
| $\beta = no\text{-}idle$ | There can be no idle machine. |
| $\beta = batch$ | There are batches in jobs. |

**Table 2.4: Examples of objective functions ($\gamma$)**

| Notation | Description |
|---|---|
| $\gamma = C_{max}$ | Minimize makespan |
| $\gamma = max\ F_j$ | Minimize maximum flowtime |
| $\gamma = \sum C$ | Total completion time |

### 2.3.2 Complexity

FJSSP is a NP-hard problem [42] that has received attention from different backgrounds of mechanical engineering, computer sciences and operations research due to its complex nature [43]. The scheduling decision is one of the most complex decision, owing to the various reasons including different objective functions, decision constraints, large number of feasible solutions, and time-domain dynamic nature [29, 44].

The complexity of the problem grows with the number of jobs in a JSSP since each job can have varied number of operations and corresponding different times. A total of $(n!)^m$ sequences can be generated in a JSSP environment [45], where $n$ is the number of jobs and $m$ is the number of machines. The problem complexity is increased manifold because of changing number of jobs, machines, processing times, uncertainties, and apprehended shop floor breakdowns [46]. It has been a major endeavor of this research to actually estimate the search space of the selected benchmark problems in a quantitative manner.

### 2.3.3 Benchmark problems

The schedules generated for FJSSPs are evaluated on benchmark problems which have been developed to assess the efficacy of a said solution approach [47]. Several sets have been proposed in this regard, some of which are summarized below.

a. Brandimarte (MK Data [48]): These instances are generated with different flexibilities of the available machines. This set includes 15 problems with minimum size of 10 jobs x 6 machines and maximum size of 30 jobs x 15 machines. However, first 10 problems from this dataset are generally used in literature.
b. Hurink et al. (HU Data [49]): These instances were generated by using the classical JSSP formulations of Fisher and Thompson [50] with the assumption of multi-purpose machines. This set includes a total of 264 instances with minimum size as 6 job x 6 machine and maximum size as 15 job x 15 machines. HU data is further grouped as follows.
    i. Sdata: Each operation can be performed on one machine only.
    ii. Edata: Some operations can be performed on more than one machine.
    iii. Rdata: Many operations can be performed on more than one machine.
    iv. Vdata: Each operation can be performed on many machines.
c. Dauzère-Pérès and Paulli (DPpaulli Data [51]): These instances were generated using the similar principles of multi-purpose machines with inherent flexibility. A salient feature of these instances is that operations to be scheduled are higher than the available machines in all possible cases. Moreover, these instances also provide variable times for selected machines against a similar problem. A total of 18 problems are presented in this data with minimum size of 10 jobs x 5 machines and maximum size of 20 jobs x 10 machines.
d. Barnes & Chambers (BC Data [27, 52]): In these instances, processing times at relevant machines are not dependent on the selected machine; whereby the basic idea was to duplicate a machine depending upon seven (7) different policies. The instances have been driven from basic data by Fisher and Thompson [50] and Lawrence [53]. A total of 21 instances are included in this set with minimum size of 10 jobs x 11 machines and maximum size of 15 jobs x 17 machines.
e. Kacem et al. (Kacem Data [54]): These instances have been designed on the hypothesis that every process can be performed on a set of machines from the available set such that the number of operations are different for different jobs. A total

of 4 instances were produced with minimum size of 4 jobs x 5 machines and maximum size of 15 jobs x 10 machines.

f. Fattahi et al. (FT data [55]): These instances contain 20 problems with small and medium sizes. Both partial and total flexible instances are available in this set.

g. Industrial instances: In addition to the conventional benchmark problems, literature also proposes specific industrial problems e.g. [56-59]. Generally, these problems are seldom used widely for evaluating algorithm efficiency.

### 2.3.4 An example of the scheduling problem

Here, the basic explanation of a selected scheduling instance is explained. Let's consider the scheduling problem as presented in Table 2.5. The problem consists of a total of 3 jobs $J_1$, $J_2$ and $J_3$, the first two consists of three operations and third job consists of two operations. The first operation of first job is denoted as $O_{11}$, the second operation of third job is denoted as $O_{32}$ and so on. In general, this is denoted as $O_{ij}$. The operations are to be performed on one of the available four machines and all operations can be performed on all machines. The time required for a said operation on a selected machine is provided accordingly. Since all operations can be performed on all machines, the problem is considered total flexible.

**Table 2.5: The SFJS6 Benchmark**

| Job | Operation | Processing Time | | | |
|---|---|---|---|---|---|
| | | $M_1$ | $M_2$ | $M_3$ | $M_4$ |
| *J1* | $O_{11}$ | 1 | 3 | 4 | 1 |
| | $O_{12}$ | 3 | 8 | 2 | 1 |
| | $O_{13}$ | 3 | 5 | 4 | 7 |
| *J2* | $O_{21}$ | 4 | 1 | 1 | 4 |
| | $O_{22}$ | 2 | 3 | 9 | 3 |
| | $O_{23}$ | 9 | 1 | 2 | 2 |
| *J3* | $O_{31}$ | 8 | 6 | 3 | 5 |
| | $O_{32}$ | 4 | 5 | 8 | 1 |

The task of calculating the overall time for completion of all jobs ($C_{max}$) problem can be undertaken through formulation of a Gantt Chart. Time is shown on *x-axis* and machines are shown on *y-axis*. The processes are placed on the chart keeping in view the prescribed constraints such that all processes are completed. In this way, Gantt chart is a graphical solution of the scheduling problem.

A possible solution of the selected instance is shown in Figure 2.11. $O_{11}$ and $O_{32}$ are performed at $M_1$ while $O_{31}$ and $O_{13}$ are performed on $M_3$. Here, $O_{12}$ is performed on $M_4$, however it can also be performed on $M_1$ however it will take more time on it. The makespan comes out to be 7 in case the jobs are scheduled as shown. It is evident that several other possible solutions are available for the considered instance and the makespan will vary accordingly. Optimal solution can only be found if all possible solutions are evaluated for makespan and the number of solutions will increase enormously with size of problem. Of course, it is an aspect of prime importance to the process planner to minimize the overall completion time of the problem.

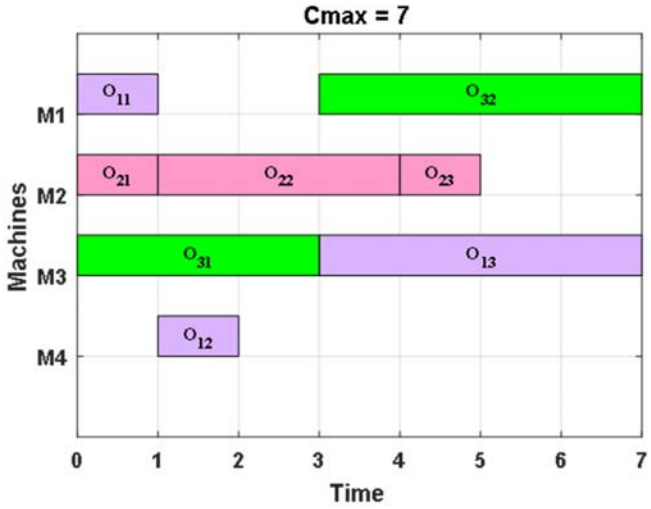


**Figure 2.11: A sample Gantt chart**

### 2.3.5 Optimization of scheduling problems and objective functions

Optimization deals with the generation of best solution from the set of available solutions to an upfront problem [60, 61]. The field is of concrete importance in the modern engineering world where multiple solutions of a single problem are available. In addition, the field also targets to solve problems that can achieve a cost function that addresses multiple objectives. Figure 2.12 chalks out a typical process to be carried out to undertake an optimization problem.

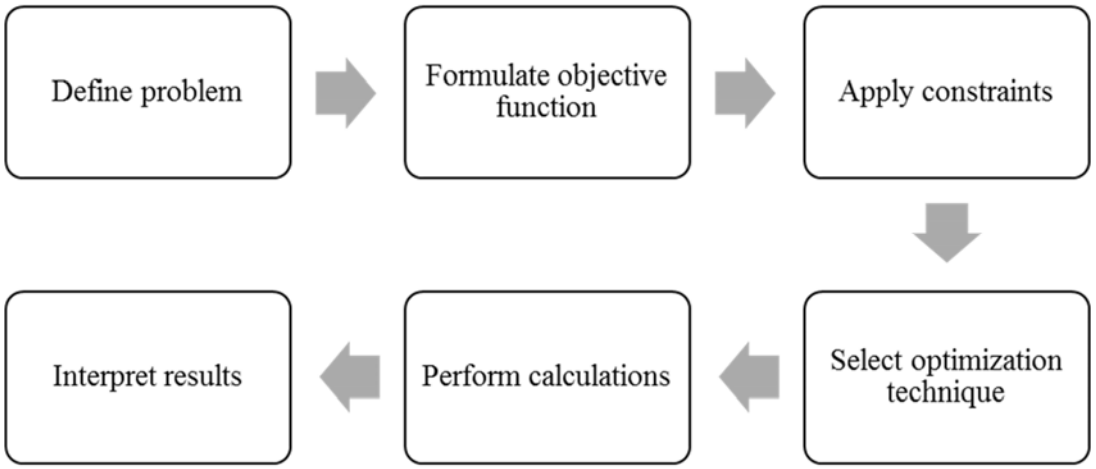


**Figure 2.12: The typical optimization process**

Whereas all feasible schedules can be executed on the shop floor for achieving the production order, only best solution will guarantee the utilization of resources and output efficiency. In view of the complexity of scheduling problems, many feasible solutions are available. The problem thus becomes twofold; one is to find feasible solutions and reject infeasible ones, second is to find optimal solution. A general optimization problem is formulated as follows.

*Minimize*

$$f(x) \tag{1}$$

*w.r.t.*

$$h_i(x) = 0 \; \forall \; i = 1 \ldots n \tag{2}$$

$$g_j(x) = 0 \; \forall \; j = 1 \ldots m \tag{3}$$

Here, *h(x)* and *g(x)* represent the equalities and inequities constraints for the objective function *f(x)*. The objective function defined in Eq. (1) can take the multi-objective form as $f(x) = f_1(x) + f_2(x) + f_3(x) + \cdots f_n(x)$. Here $f_1(x), f_2(x), f_3(x), \ldots f_n(x)$ are the sub-functions of the objective function describing different objectives. Objective functions are set in an optimization problem to achieve a pre-determined goal. In the current high business flux and customer-oriented market, it has become inevitable that the operations are performed on the

production shop floor in an optimal manner to reduce cost, wastage of resources and time. Table 2.6 presents the popular objective functions that have been studied in the FJSSPs.

**Table 2.6: Popular Objective Functions for FJSSP**

| Function | Description | Calculation |
|---|---|---|
| Mean completion time | Mean time required to complete a single job | $\bar{C} = \frac{\sum_{j=1}^{n} C_j}{n}$ |
| Total Tardiness | Difference between due date and completion time for all jobs | $T = \sum_{j=1}^{n} T_j$ |
| Makespan | Total completion time for all jobs | $C_{max} = \max_{1 \le j \le n} C_j$ |
| Maximum flow time | Total time taken by the job to get processed from the shop floor (including wait) | $F_j = \max_{1 \le j \le n} F_j$ |
| Total workload of machines | Total time for which all machines are working | $W_T = \sum_{j=1}^{n} W_j$ |

## 2.4 Algorithms for scheduling

Scheduling problems need a structural approach for achieving solution [62]. There are many techniques available to solve the scheduling problems and a holistic classification is presented in Table 2.7.

**Table 2.7: Classification of scheduling algorithms**

| Class | Sub-class | Algorithm | Reference |
|---|---|---|---|
| Exact | Constructive | Johnson's algorithm | [63] |
| | | Moore's algorithm | [64] |
| | | Lawler's algorithm | [65] |
| | Enumerative | Integer programming | [66-68] |
| | | Branch and bound | [69, 70] |
| | | Dynamic programming | [40, 71] |
| Approximate | Heuristics | NEH heuristic | [72] |
| | | Shifting bottleneck | [24, 73] |
| | Metaheuristics | Simulated annealing | [25, 28] |
| | | Tabu search | [74, 75] |
| | | Greedy search | [76, 77] |
| | | Genetic algorithm | [78, 79] |
| | | Ant colony optimization | [80, 81] |
| | | Artificial immune system | [82-85] |

| | | | |
|---|---|---|---|
| | | Differential evolution | [86, 87] |
| | | Harmony search | [88-90] |
| | | Particle swarm optimization | [91-93] |
| | | Artificial bee colony | [94, 95] |
| | | Neighborhood search | [96, 97] |
| | | Frog leaping algorithm | [98-100] |
| | | Biogeography-based optimization | [101] |
| | | Firefly algorithm | [102, 103] |
| | | Invasive weed optimization | [104] |

Generally scheduling techniques are classified as exact and approximate [105]. Recently, the approximate techniques have gained exceptional interest due to the increased complexity of the problems. Approximate techniques provide good solutions within acceptable time frame in comparison to the exact methods which can provide guaranteed optimal solutions for smaller instances but may tend to take infinite solution times on larger problems [106]. Heuristics provide a straight-forward rule-based solution for the problem at hand which may not be optimal. In the context of scheduling problems, priority rules are classical example of heuristics. In contrast, metaheuristics generate sequential neighborhood solutions through stochastic techniques or take inspiration from any other nature / technical process.

Conventional optimization methods include calculus based techniques, e.g. Karuch-Khun Tucker methods, gradient based methods [107]. These methods reply on the availability of a mathematical objective function which can be optimized using calculus based or numerical techniques. Further, regularity and convexity checks are the hallmark requirement of these methods due to which they cannot undertake NP-hard problems. On the other hand, enumerative methods provide a step-by-step solution procedure which can check the best solution during the process. Depending upon the size of search space, these methods become inefficient and may cause huge delays in schedule generation.

The basic ingredient of a metaheuristic technique is a guidance mechanism for the underlying heuristic to effectively evolve the solution until an acceptable termination criteria [108]. The search space is explored in this process through diversification (further evaluation of the unexplored areas) and intensification (exploration in the already explored area), whereby both techniques are used side-by-side. These methods can solve the NP-hard problems in an effective manner and hence they have gained extreme popularity and research interest during the last few

decades [109]. Metaheuristics methods have various techniques, which are generally inspired from nature-based systems, sometimes termed as nature-inspired-algorithms. The number of techniques are becoming more and more diverse in response to the No-Free-Lunch proposition [110] which demands different algorithms for different situations. A limited discussion for some selected metaheuristic methods is presented below.

a. Tabu search: It is a neighborhood based local search method which uses a memory list to prevent a revisiting previously evaluated solution. This method is proven to be very useful in avoiding local minima traps. However, in a larger search space, the number of iterations to reach an acceptable solution increase enormously along with the length of tabu list [111].
b. Simulated annealing: The method is named after the famous heat treatment process. The method can deal with non-linear models and can find a good approximation to the global minima. The methods relies on the quality of initial startup solution for the final solution quality [112].
c. Ant colony optimization: It is one of the most popular nature-inspired-algorithm that relates to the social conduct of ant colonies. It imitates real ants when they search for food and tend to follow the shortest available path based upon pheromone trials. The method can evaluate multiple solutions at a time through various paths [113]. It relies on the probability calculations for path selection which ultimately govern the solution quality.
d. Particle swarm optimization: This method adopts the actions of birds in flocks when they search for food. The method depends upon the current state, presumed best state and the velocity. The algorithm has proven difficulties in dealing with scattering problems [93].
e. Artificial bee colony: The method is inspired from the scavenging behavior of honeybees. As in all other metaheuristic approaches, the method has three agents (i.e., employed / unemployed bees and sources of food) which are used to incorporate search based on feedback mechanisms.
f. Integer programming: These approaches attempt to solve the FJSSP through formulation of mathematical models.
g. Artificial immune system: The method is derived from the behavior of immune system of living things that tries to drive the body back to normal i.e., optimal condition in case of any disease.

h. Harmony search: operates inspired from operation of orchestra that tries to produce melodious sound.
i. Memetic algorithms: two algorithms are combined.
j. Neighborhood search: these methods find the optimal solutions about the current solution through a strategy to explore the neighborhood.

## 2.5 Dispatching rules

Schedules can be generated using dispatching rules, which can undertake the problem under certain objective and return the schedule [114]. It has been reported that scheduling rules can be employed with ease for obtaining acceptable solutions [115] in a stand-alone manner. Conventionally, these rules have been used for producing schedules on the production line in a manual manner. Since these rules target to optimize a single objective, they may not produce an overall optimal schedule, hence many combinations are used accordingly. Some of the important techniques are listed in Table 2.8.

**Table 2.8: Some important dispatching rules**

| Rule | Calculation | Description |
|---|---|---|
| Shortest processing time | $P_{ik} = min_{j=1}^{\zeta}[p_{ij}]$ | Operation requiring minimum time is scheduled |
| Longest processing time | $P_{ik} = max_{j=1}^{\zeta}[p_{ij}]$ | Operation requiring maximum time is scheduled |
| Earliest due date | $d_k = min_{j=1}^{\zeta}[d_j]$ | Operation with smallest due date is scheduled |

## 2.6 Genetic algorithm

Genetic algorithm (GA) is the most popular evolutionary algorithms for finding near-optimal global solutions. It was introduced by Holland [116] and further developed and popularized by Goldberg [117].

GA has been developed in line with the natural evolution process and many advancements have been incorporated since its inception. It is a global optimization technique that searches the best solution by recombining the available solutions. Due to the unmatched ease of implementation and adaption to different sets of problems, GA has been applied to diversified fields of engineering [118, 119], data mining [120], demand estimation [121] and medicine [122]; only to mention a few. A great deal of literature has been published to describe the applications of GA e.g., [123-126].

### 2.6.1 Theoretical background

The algorithm is based on the well-known principles of "survival-of-the-fittest" and "natural-selection" introduced by Darwin [127]. He claims that the species of living being are evolving over the period of time through generations such that each generation exhibits more better individuals than the previous through adaptation to the prevalent environment. He further argues that nature, itself, tends to choose better individuals over time and the inheritance characteristics of parents are transferred to the offspring. Figure 2.13 presents a comparison of similarities between the natural evolutionary process and the terminology of GA.

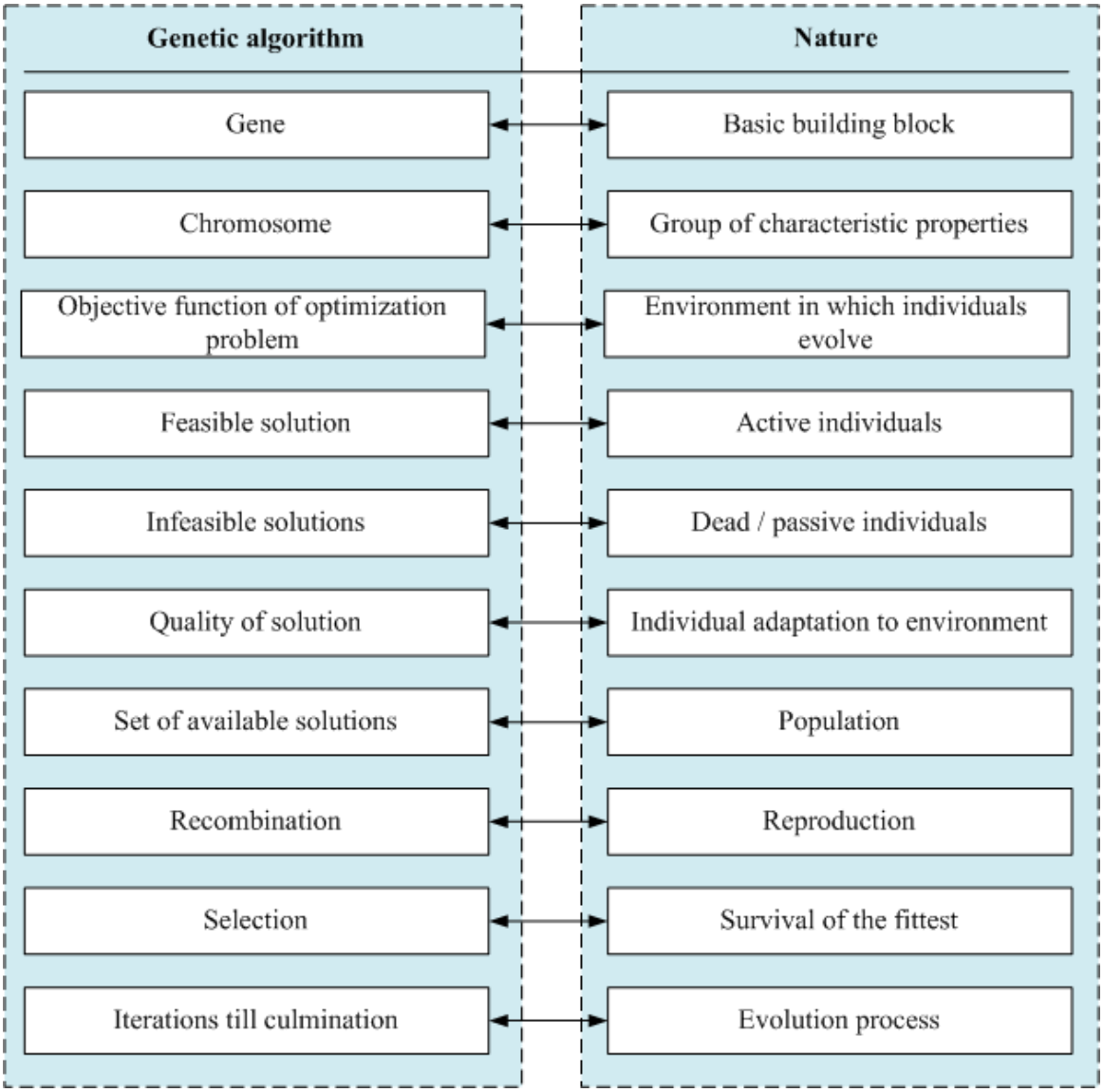


**Figure 2.13: Analogies between genetic algorithm and natural evolution process**

### 2.6.2 Explanation of GA

The basic building block of GA is a 'gene' which contains relevant information regarding the optimization problem. Genes are stacked together to formulate a chromosome, which describe the characteristic information in the chromosome, i.e., essentially a candidate solution is represented in a holistic manner for further processing by the algorithm. General representation methods include strings, rules permutations etc. Every chromosome is different due to the inherent properties and information contained in the relevant genes. The intrinsic properties, either good or bad, depict the quality of the individual chromosome, i.e., it can be the potentially best solution, or it may be an infeasible one. Figure 2.14 presents two different chromosomes with every gene represented in a different box. It is evident that the 3rd gene differs in the two chromosomes.


**Figure 2.14: Two different chromosomes**

Several chromosomes group together to formulate a population of individuals. As different chromosomes are generated, a population has a diverse set of chromosomes. Population diversity plays a vital role in the exploration of best solution in the evolutionary algorithms [128]. An important consideration in this part is that generation methods must prevent the generation of infeasible solutions, otherwise the same will have to be evaluated at a later stage for rejection.

The population has parent chromosomes which reproduce through recombination methods to generate offspring [129]. Since the offspring are generated through the parents, they typically include characteristic information from their parents; however, they are different owing to the sequence of the genes.

GA has two distinct recombination operators, crossover and mutation which basically aim to explore the search space by generating neighborhood solutions. Crossover occurs between two parents such that the information exchanges between the parents. Conventionally, a single-point crossover is employed whereby a random position for crossover is selected and the relative parent parts are swapped to generate two further offspring. Figure 2.15 presents a single-point crossover. In mutation, a single parent gene 'mutates' to generate a new offspring. Figure 2.16 presents a representation of a typical swap mutation. Obviously, there are other formulations for recombination operators; however, they are algorithm specific.

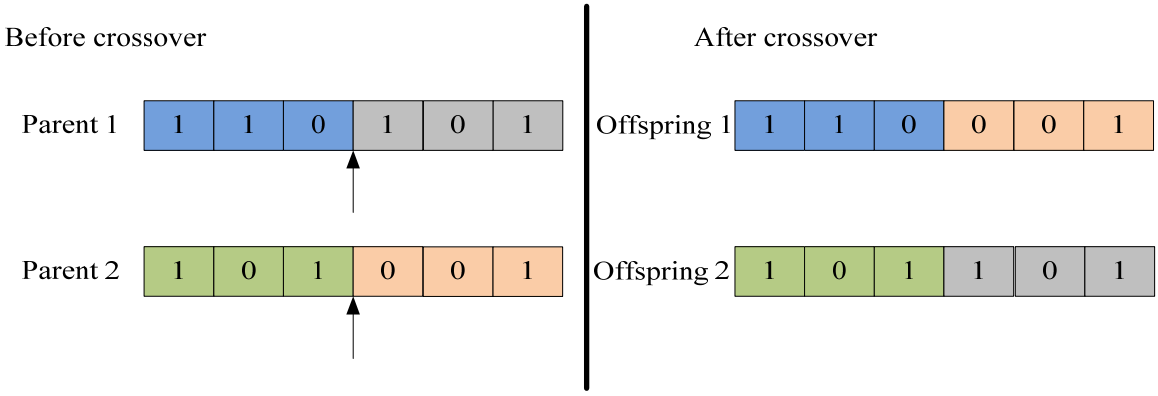


**Figure 2.15: A typical one-point crossover**

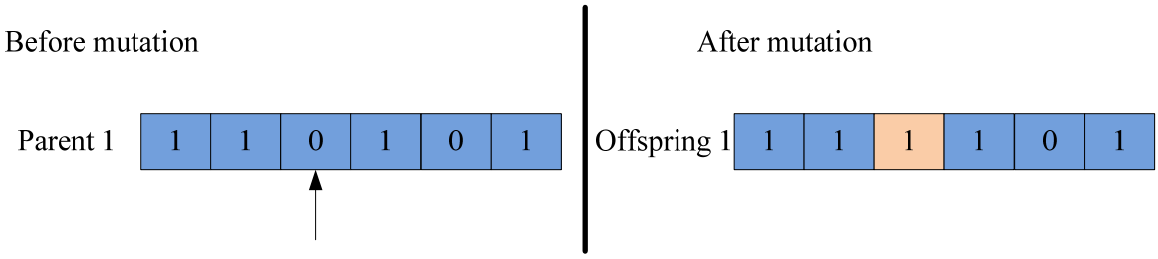


**Figure 2.16: A typical swap mutation**

A new population group is then formulated with the inclusion of offspring with the parents. All chromosomes are then evaluated for 'fitness', i.e. the criteria against which the chromosomes are judged for being better individuals. This is the objective function of the optimization problem. The chromosomes are then ranked according to the fitness.

Natural selection phenomenon is then performed to bring better individuals to the next population. Since GA requires several generations in order to explore the complete search space effectively and to achieve good solutions, the process continues as per requirement. In this regard, elite chromosomes are generally carried forward as they represent the best solution. Moreover, several chromosomes with compromised fitness are also carried forward in order to maintain population diversity. Finally, the algorithm terminates after achieving the prescribed criteria. The general termination criteria are the achievement of a predetermined number of generations or no solution improvement over a said number of generations.

GA is an evolutionary process and the quality of solution provided by it depends on several things. However, one of the major considerations is to ensure the evaluation of complete search space. In this regard, following concepts are employed:

a. Exploration / Diversification: New areas of the search space in distant environments are evaluated in addition to the already explored areas. Crossover is an example of this technique.
b. Exploitation / Intensification: An already searched solution is exploited further to yield a possibly better solution. Mutation is an example of this technique.

Since, it becomes impossible in to completely evaluate the extremely huge search spaces, it is always a trade-off between maintaining the population diversity and convergence. An overly diverse population may fail to converge, and an under-diverse population may converge prematurely. In both cases, solution quality is badly hampered.

Figure 2.17 presents a schematic with regards to different steps of GA. Here, four genes are generated with color-based coding to provide and idea of the evolutionary process.

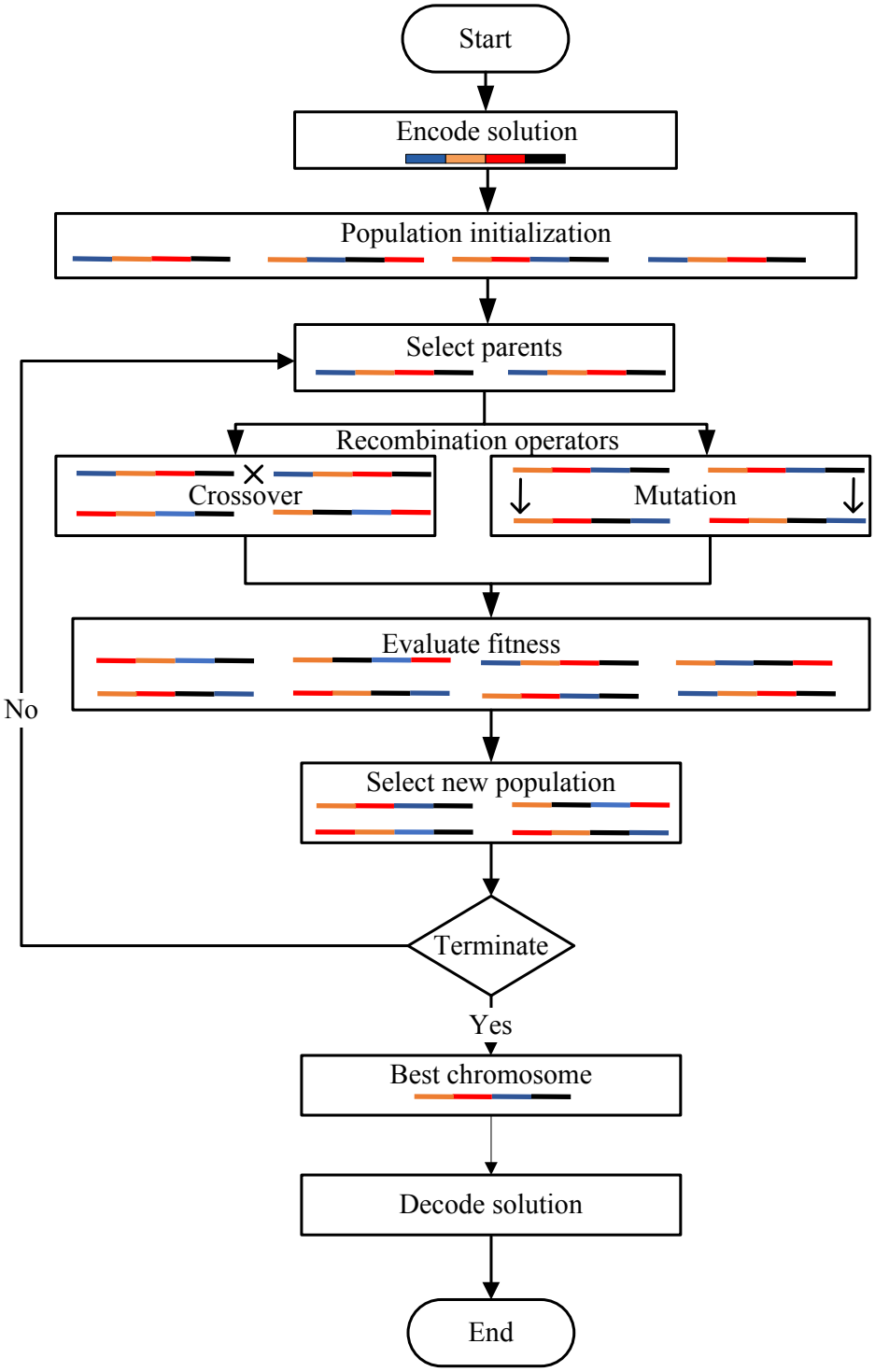


**Figure 2.17: Flowchart of GA with gene representations**

### 2.6.3 Advantages and disadvantages

Following are the major advantages of GA.

a. The algorithm has been developed through taking inspiration from nature and it is easy to comprehend.
b. Enables multiple search points at a single time depending upon the population which provides the facility of multi-dimensional search.
c. Derivative free method.
d. Based upon probabilistic techniques instead of deterministic techniques.
e. Can undertake different types of optimization problems e.g. stochastic, continuous, and noise.
f. GAs have been applied to vast nature of problems including stock prediction [130], financial planning [131], image processing [132], engineering design [133], trajectory planning [134], data fitting [135] etc.
g. The algorithm has received special attention in solving the sequencing problems in various domains (e.g. flight schedules [136], operation room scheduling [137], traveling salesman problem [138], plant layout [139]).

Following are some of the drawbacks of GA.

a. Chromosome representation is to be changed for specific problem, even within the same nature of the problem.
b. Similarly, the recombination operators are problem specific and may severely jeopardize the overall algorithm efficiency if not correctly defined.
c. Definition of fitness function is an uphill task.
d. Decision regarding different parameters e.g. population size, generation size, termination criteria etc. is extremely important to ensure solution quality and to prevent premature convergence.
e. Each solution must be evaluated for fitness.
f. Premature convergence is generally encountered when a chromosome with far better solution is found since the elitist criteria takes it to new populations and it reproduces to generate many neighborhood solutions with better fitness. This situation ultimately results on local minima trap.

## 2.7 GA for FJSSP

### 2.7.1 Literature summary

Davis [140] and Brucker [26] are the pioneers to use the adaptations of GA for FJSSPs. Since then, an enormous amount of literature has been published to solve various aspects of the problem. This is due to fact that the problems are truly challenging and provide many horizons

for optimization endeavors. Consolidated research surveys have also been published in the literature to address the application of GA to FJSSPs; some of which are presented in Table 2.9.

**Table 2.9: Surveys published to review FJSSP literature**

| Reference | Survey Focus |
|---|---|
| Chaudhry et al [141] | Classified the literature on FJSSP with solution approaches with statistical data |
| Xie et al [142] | Future research directions along with real world applications and development trends |
| Gen et al [143] | Multi-objective optimization of the FJSSP |
| Lei et al [144] | |
| Lal et al [145] | |
| Genova et al [146] | |
| Amjad et al. [10] | Literature classification for FJSSP solution approaches using GA |
| Zhang et al. [147] | Viewpoint of industry 4.0 |
| Çaliş et al. [148] | Artificial intelligence-based strategies |
| Fan et al. [115] | Review of scheduling rules |
| Allahverdi [149] | Setup times consideration |

Since there are two sub-problems encountered for the FJSSP, they can be solved in parallel (integrated method) [150, 151] or one by one (hierarchical method) [152, 153]. Figure 2.18 [10] presents the trend of FJSSP publications conducted by using GA based approached over the last few decades. It is evident that the trend is increasing which indicates the popularity, efficacy and adequacy of the algorithm. Amjad et al. [10] have classified the literature on following basis.

a. Pure GA
b. Advanced GA
c. Hybrid GA

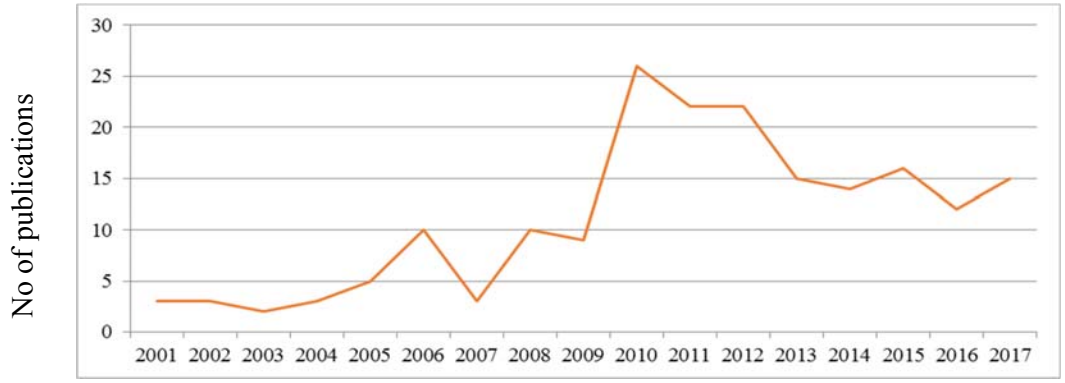


**Figure 2.18: Trend of FJSSP publications with GA approaches**

### 2.7.2 Chromosome encoding

Chromosomes are representatives of a candidate solution and hence researchers have endeavored to generate representations that can handle the problem in an effective manner. Some of the chromosome encoding / representation schemes are summarized below.

a. Mesghouni et al. [154]: Developed for parallel machine problems, whereby machines are placed in parallel and operations are sequenced on them.
b. Chen et al. [155]: Consists of two-part integer-based representation where the first part is for machine indexing and second is for operation sequencing.
c. Ho et al. [156]: Consists of two-part representation where first part represents the operation sequence corresponding to other operations and second part deals with machine assignment.
d. Kacem et al. [54]: Consists of coding based on start time and end time of a certain operation in conjunction with the assignment table.
e. Zhang et al. [157]: Consists of machine selection and operations selection parts scheme whereby the whole chromosome interprets the schedule.

### 2.7.3 Population initialization

Since, GA needs several chromosomes, population initialization methods have also been evolved. The most popular in this regard is the random population initialization method [158, 159]. Other important methods include global and local initialization [160].

### 2.7.4 Recombination operators

The recombination operators play an important role in intensification and diversification of the initial population [77, 161]. Since these operators are used to generate new offspring, an

important factor is to ensure that feasible solutions are evolved during the process. Moreover, the operators are designed in accordance with the chromosome definition such that offspring are generated as per the selected definition. The operators are applied on the parent population through pre-defined probability; hence only some selected parents go through crossover. This can generate following scenarios:

a. Elite (best) chromosomes may only be selected for recombination, which will eventually lead to premature convergence as all population will concentrate against the best chromosome.
b. The chromosomes with compromised fitness will lead the convergence nowhere and a result with poor fitness will be obtained.
c. Since the above-mentioned options will not generate good results, a selected population undergoes recombination in which both elite and non-elite parents are available. The selection is generally made through random criteria.

The crossover operators are mainly used to inculcate diversification through recombination process in a random manner [162]. Traditionally, points are selected from the parents and swapping is performed to generate two offspring. Therefore, single-point [58, 163-171], two-point [172-184], multi-point [37, 185-189] and uniform crossover [190-193] have been used. A major consideration is to avoid generation of infeasible chromosomes during this process [194, 195]. In this regard, precedence preserving order-based crossover (POX) and its improved forms [196-208] have been used tremendously. Moreover, multiple methods have also been combined in a single algorithm to enhance solution quality as every method has its own pros and cons [153, 157, 209-223]. Figure 2.19 depicts the frequency of use for different types of crossover.

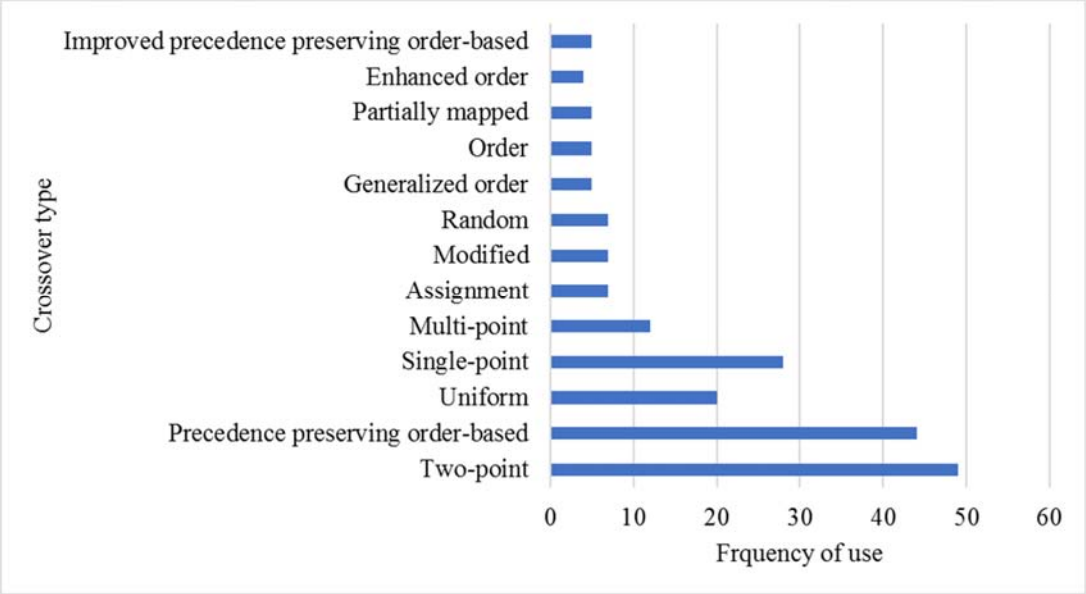


**Figure 2.19: Frequency of use for different crossover types**

Mutation operators also inculcate the concept of search space evaluation, but they entail single gene change and hence promote more of a local search phenomenon since it generates offspring in the neighborhood. The process consists of selecting a random gene from the parent chromosome and changing it to generate new offspring which is termed as swap mutation [58, 164, 169, 178-180, 182, 188, 190, 192-194, 201, 210, 215, 219, 220, 224-233]. Random mutation [37, 184-186, 207, 208, 234-237] is another similar nature of operator. As pointed out in crossover, mutation process has also to be developed carefully in order to avoid generation of infeasible solutions. Additionally, different combinations of mutation operators have also been used some of which are identified in [163, 208, 223, 238-246]. Figure 2.20 presents frequency of different types of mutation used in literature.

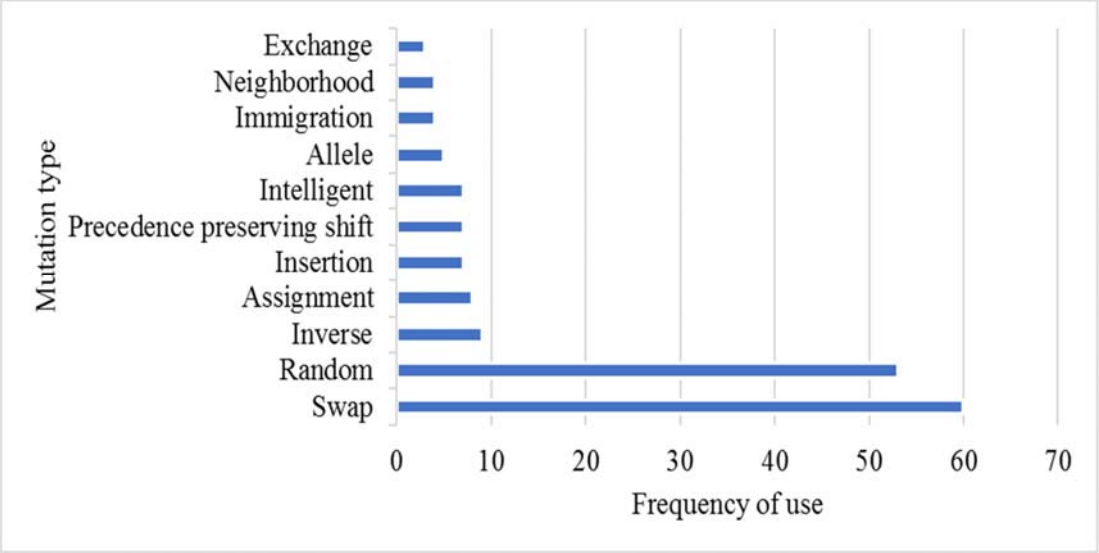


**Figure 2.20 Frequency of use for different mutation types**

The selection process is employed to identify the most suitable individuals for inclusion in the population to improve the fitness function as much as possible. Following are the major types of selection used in the literature.

a. Elitism [169, 191, 205, 218, 231, 233, 247-251]: Only the fittest individuals are selected for next generation.
b. Roulette wheel [59, 150, 157, 180, 185, 186, 194, 197, 213, 224, 227, 252-256]: Individuals are selected based upon the probability such that fit offspring are selected more and fewer are selected from lesser fit offspring, however all types of individuals are permitted to pass.
c. Random [162, 163, 257, 258]: Individuals are selected randomly.

### 2.7.5 Classification of GA approaches for FJSSP

The available literature reporting attempts for solving FJSSP instances with GA has been classified as pure GA, advanced forms of GA and hybrid GA. Pure GA has been used to attempt the FJSSP most of the times as shown in Figure 2.21 with hybrid and advanced GA in the second and third places respectively. NSGA has been used to attempt the multi-objective functions which have been attempted more than 3 times as compared to single objective functions.

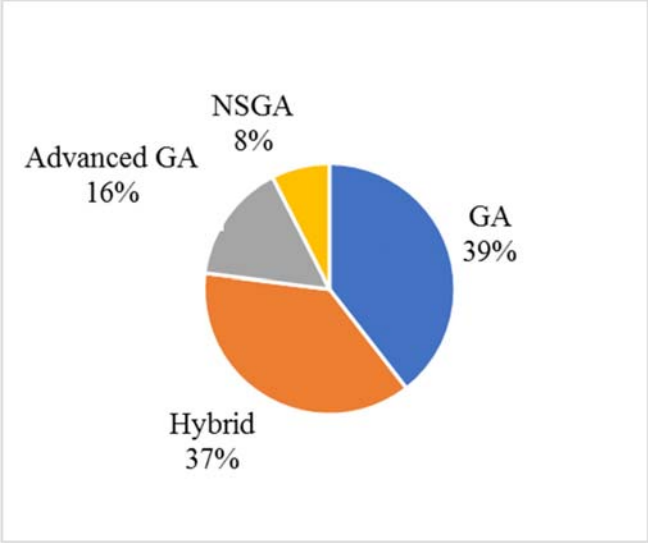


**Figure 2.21: Classification of GA based FJSSP literature**

Different GA based methods have been identified and presented in Figure 2.22 along with their frequency of use. It is noticeable that pure GA has been used the most for attempting the FJSSPs, while GA + TS [36, 170, 217, 221, 236, 245, 248, 251, 259-263] has been used the most in the hybrid methods. Other hybrid applications include local search GA + LS [160, 204-206, 208, 222, 264-266], SA [244, 246, 250, 267, 268] and VNS [189, 207, 216, 269-271].

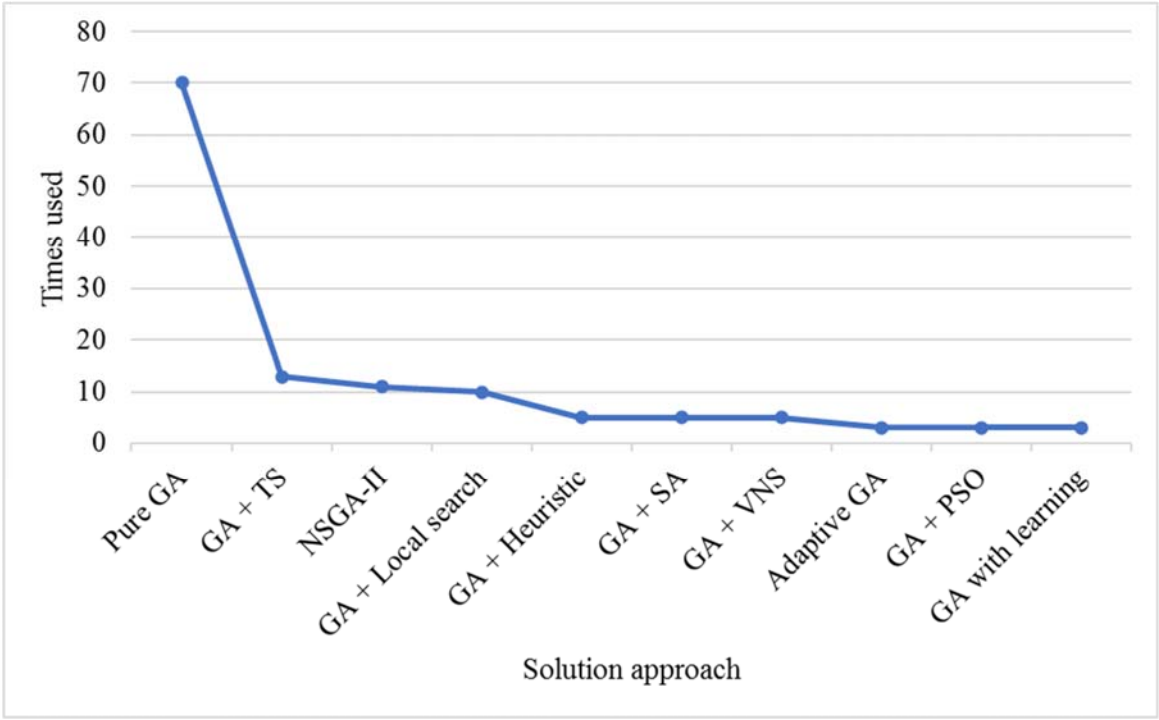


**Figure 2.22: Different GA based approaches for FJSSP and their application**

Figure 2.23 presents the statistics for the use of different objective functions in the optimization process. It is evident that makespan is the most attempted single objective function while total workload has been attempted through multi-objective approach along with other functions.

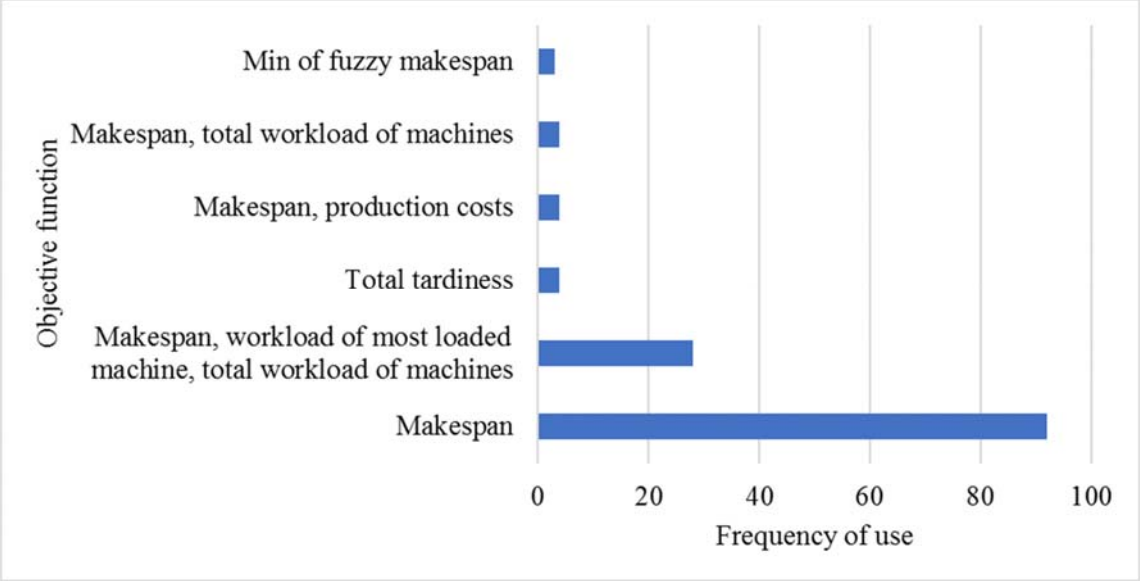


**Figure 2.23: Frequency of objective function attempts (single / multi)**

## 2.8 Gap Analysis

It is concluded from the sections presented above that an enormous amount of literature is available with GA based solution approaches for solving FJSSPs. In this regard, pure GA has been used for solution and dispatching rules have also been integrated. Hence, this research aims to formulate two separate solution approaches one with dispatching rules and other with pure GA. Following summarizes the gap in literature which has provided motivation for this research.

### 2.8.1 GA integrated with scheduling rules (GA-PR)

Hand-schedulers have traditionally been using the dispatching rules for generation of schedules for onward implementation at the shop floor. Since they are purely heuristic methods, they do not guarantee an optimal solution and GA has been integrated with them in order to improve their efficacy. Following is a summary of literature reporting these approaches:

a. Hybrid dispatching rules have been utilized for machine assignment problem [171].
b. A procedure based upon priority rules has been suggested, however results were not comparable [206].

c. A quantitative study has been conducted on 36 rules and their combinations for sequencing an industrial unit and it has been concluded that SPT provides best results [272].

The GA-PR algorithm attempts to further evaluate the performance of selected priority rules against the benchmark problems and various improvements have been incorporated in comparison to the available methods.

### 2.8.2 GA with iterative diversification technique (GA-IDT)

The evolution of GA is dependent upon the selection of best-fit individuals according to the optimization function / environment. To evolve properly, diversity of population is an essential phenomenon because it ensures search space exploration in an effective manner. Consequently, intensified population in a concentric point of the search space indicates convergence on one hand, but that may also occur due to local minima trap. This is a known issue of GA approach [273, 274] which hinders the solution quality. Alternatively, a needlessly diverse population induces convergence barrier and the algorithm may continue to run infinitely. It is therefore a matter of great interest to propose a method which provides balance between the two opposite extremes. Moreover, the fine tuning of mutation and crossover probabilities is also an essential requirement for exploration of search space. In this regard following relevant literature has been identified to maintain population diversity:

a. The preservation of an elite individual has been used and other individuals have been selected through other selection techniques [201].
b. A specifically designed crowding distance has been proposed [275].
c. A modification in the mutation approach has been proposed [276].

Further, following literature reports efforts for different variations of GA for achieving / improving good quality results by improving exploration of search space:

a. Adaptive recombination operators have been proposed in accordance with varied fitness values [277]. The reference algorithm was tested against a non-popular instance of the size of 10 x 10.
b. Adaptive mutation technique has been proposed for smaller rates at the initialization stage of the algorithm and bigger rates at the end to forcefully increase the efficiency of search space exploration [278].
c. A mixture of elitist and tournament selection techniques have been proposed with a linear interpolation method for recombination operator probabilities [247].

d. A strategy for 3 level parametric setting (high-medium-low) has also been proposed for recombination operator probabilities in view of the convergence pattern of the algorithm [189].

As indicated above, limited literature reports the methodologies to manage the diversity in a holistic manner. Hence this aspect is a matter of further research. GA-IDT proposes a framework to maintain the population diversity and manage intensification and diversification in an effective manner to generate solutions of better quality.

## 2.9 Summary

This chapter offers an introductory preamble to the flexible manufacturing systems and different options of flexibility. An in-depth analysis of the literature on approaches for solving the FJSSP with an emphasis on genetic algorithm is then presented. The trends of literature are explored through statistical data and prospective research opportunities have been identified which contribute to the motivation of this research.

# 3 Chapter 3 – Problem Formulation and Simulation Environment

## 3.1 Problem formulation

FJSSP benchmark instances present a set of deterministic number of jobs, operations, machines and respective operation times. Consequently, the FJSSP is modeled as *M* set of machines which have to process *N* set of jobs, where;

$$M = [M_1, M_2, M_3, \dots, M_m] \tag{4}$$

$$J = [J_1, J_2, J_3, \dots, J_N] \tag{5}$$

Every job *j* from the set of jobs $j \in N$, consists of a pre-determined number of operations which have to be carried out on a machine to be selected from the available set of machines.

$$O_i = [O_{i1}, O_{i2}, O_{i3}, \dots, O_{ij}] \tag{6}$$

Here, the $j^{th}$ operation $i^{th}$ job is denoted as $O_{ij}$ and $i^{th}$ job will have a total number of operations as $J_{io}$. Now, every operation is to be completed on a machine $M_k$ with following conditions.

$$M_k \in M_{ij} \tag{7}$$

$$M_{ij} \subset M \quad \forall \quad P - FJSSP \tag{8}$$

$$M_{ij} = M \quad \forall \quad T - FJSSP \tag{9}$$

Where, P-FJSSP present the partial class of flexibility i.e. some operations cannot be performed on all machines and T-FJSSP belong to the total class of flexibility i.e. each operation can be performed on all available machines.

## 3.2 Problem constraints and assumptions

Since every operation must be scheduled, it will have a start time $t_{ijk}$ and end time $E_{ijk}$ on machine $M_k$. The reference processing time on machine $M_k$ is denoted as $P_{ijk}$ during which the machine will be bust and will not be able to take any other jobs. There is a total number of sequences available for a specific instance of the problem, *L*. Now, for ease of sequencing, a number can be allocated to every operation as shown in Eq (11).

$$L = \sum_{i=1}^{N} J_{io} \tag{10}$$

$$n_{ij} = \sum_{x=1}^{i-1} J_{xo} + j \tag{11}$$

Consequently, the operation $O_{ij}$ can only start at machine $M_k$ such that following conditions are fulfilled.

a. The operation at hand has no previous operations pending i.e. its preceding operations have already been completed and it is in waiting state for the present operation.
b. The machine on which the operation $O_{ij}$ is to be allotted is available to carry out the operation [Eq. (11)] at the time of release $r_{ijk}$ of the operation.

$$r_{ijk} = \max\left(C_k, t_{ij'k'} + P_{ij'k'}\right) \tag{12}$$

Following are the integral problem specific variables which can vary in the ranges as mentioned in Eq. (13) through Eq. (16).

$$1 \leq i \leq N \tag{13}$$
$$1 \leq j \leq J_{io} \tag{14}$$
$$1 \leq k \leq M \tag{15}$$
$$1 \leq n_{ij} \leq L \tag{16}$$

Since we are considering the general form of FJSSP, its universal constraints are described as follows.

$$t_{ijk} \geq 0 \qquad \forall\, O_{ijk} \epsilon N \tag{17}$$
$$r_{ijk} \geq 0 \qquad \forall\, O_{ijk} \epsilon N \tag{18}$$
$$t_{ijk} - t_{i'j'k} \geq P_{i'j'k} \qquad \forall\left(O_{ij}, O_{i'j'}\right) \epsilon S_k \tag{19}$$
$$t_{ijk} - t_{ij'k'} \geq P_{ij'k'} \qquad \forall\left(O_{ijk}, O_{ij'k'}\right) \epsilon\, J_{io} \tag{20}$$
$$E_{ijk} - t_{ijk} = P_{ijk} \qquad \forall\, O_{ij} \epsilon\, J_{io}\ \&\ M_k \epsilon\, M \tag{21}$$
$$t_{ijk} = \max\left(C_k, r_{ijk}\right) \tag{22}$$

In general setting of FJSSP, all machines are accessible and available when the jobs are to be started i.e. $t = 0$ [Eq. (17) & Eq. (18)]. The multi-purpose machines are allowed in the formulation however only operation can be undertaken on a said machine at a specific instance of time [Eq. (19)]. Referring to the physical restriction of the FJSSP, all operations are to be undertaken in a pre-decided sequence; hence $O_{12}$ cannot be performed before $O_{11}$ [Eq. (20)]. Machine breakdown are not considered hence all operations will be undertaken and completed

continuously without any break [Eq. (21)]. Similarly, there is no-wait shop scheduling system and operations will be undertaken as soon as the machines are available [Eq. (22)].

## 3.3 Insight to the problem formulation

Let us consider the benchmark problem of Fattahi 12 i.e. MFJS 2 as shown in Table 3.1. The problem has a size of 5 jobs x 7 machines and each job has further three operations. Hence the problem has an overall size of 15 operations x 7 machines. This is a partial flexible instance as $M_{ij} \subset M$; e.g. $O_{11}$ can only be performed on $M_1$, $M_2$ and $M_3$ while it cannot be performed on $M_4 - M_7$. Moreover, the operation times of $O_{11}$ are different on all the possible machines, whereby the machines not available for a said operation are marked with infinity (∞).

**Table 3.1: MFJS 2 benchmark problem**

| Job | Operation | Processing Time | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ | $M_6$ | $M_7$ |
| $J_1$ | $O_{11}$ | 147 | 123 | 145 | ∞ | ∞ | ∞ | ∞ |
| | $O_{12}$ | 123 | 130 | ∞ | 140 | ∞ | ∞ | ∞ |
| | $O_{13}$ | ∞ | ∞ | ∞ | 150 | 160 | ∞ | 200 |
| $J_2$ | $O_{21}$ | 214 | ∞ | 150 | ∞ | ∞ | ∞ | ∞ |
| | $O_{22}$ | ∞ | 66 | 87 | 99 | ∞ | ∞ | ∞ |
| | $O_{23}$ | ∞ | ∞ | ∞ | ∞ | 178 | 95 | 150 |
| $J_3$ | $O_{31}$ | 87 | 62 | ∞ | ∞ | ∞ | ∞ | ∞ |
| | $O_{32}$ | ∞ | ∞ | 180 | 105 | ∞ | ∞ | 145 |
| | $O_{33}$ | ∞ | ∞ | ∞ | 190 | 100 | 153 | ∞ |
| $J_4$ | $O_{41}$ | 87 | 65 | ∞ | ∞ | ∞ | ∞ | ∞ |
| | $O_{42}$ | ∞ | ∞ | 250 | ∞ | 173 | ∞ | ∞ |
| | $O_{43}$ | ∞ | ∞ | ∞ | 145 | ∞ | 136 | ∞ |
| $J_5$ | $O_{51}$ | 128 | 123 | 145 | ∞ | ∞ | ∞ | ∞ |
| | $O_{52}$ | ∞ | ∞ | 86 | 65 | 47 | ∞ | ∞ |
| | $O_{53}$ | ∞ | ∞ | ∞ | ∞ | 110 | 85 | ∞ |

Following are the salient mathematical interpretations of the selected problem.

i. Since there are 5 jobs and 7 machines the first two parameters of the instance are $N = 5$ and $M = 7$.

ii. All jobs have 3 operations each and total number of operations are $L = 15$.

iii. The index $i$ assumes a value from [1 – 5] since there are five jobs and all jobs are required to be addressed.

iv. Similarly, $j$ assumes a value from [1 – 3] since maximum number of operations of any job are three.

v. There are total seven machines available in this problem i.e. $k$ can take any value from [1 – 7].

vi. According to the notation, index of machines is formulated as Eq. (23). This gives a complete row vector list of available machines. Furthermore, there are three operations for all jobs, hence the parameter $J_{io}$ is formulated as Eq. (24).

vii. In view of the reasoning explained in the nomenclature of $i$ and $j$, $O_{ij}$ is formulated as a matrix of 5 rows and 3 columns where the rows denote the jobs and columns represent the relevant number of operation e.g. $O_{42}$ means the 2nd operation of 4th job as $i = 4$ and $j = 2$.

viii. $U_{ij}$ is a measure of flexibility of each machine with regards to the said operation. Only two machines are available to take the operation $O_{53}$, however three machines can undertake $O_{52}$. This aspect builds flexibility in the machine assignment because only the machines capable to undertake the operations are allowed to carry out the operation.

ix. A sequence number ($n_{ij}$) for each operation is generated to track the sequence identification and is generated for problem under consideration in Eq. (27). This number is simply incremented for each operation and the value is stored in the corresponding matrix index.

x. Since the index of $U_{ij}$ only provides the flexibility level of each operation and not the exact identification of machines which can undertake a selected operation $\Omega_{ij}$ is populated to resolve this aspect and presented in Eq. (28). Again, the index is mapped as per the already populated $O_{ij}$ scheme to impart a universal understanding. It is evident that $O_{53}$ can be performed on $M_5$ and $M_6$, while $O_{51}$ can only be performed on $M_1$, $M_2$ and $M_3$. This matrix can be searched to find out the exact information of machine which can undertake the relevant operation.

$$M_k = [M_1 \quad M_2 \quad M_3 \quad M_4 \quad M_5 \quad M_6 \quad M_7] \tag{23}$$

$$J_{io} = [3 \quad 3 \quad 3 \quad 3 \quad 3] \tag{24}$$

$$O_{ij} = \begin{bmatrix} O_{11} & O_{12} & O_{13} \\ O_{21} & O_{22} & O_{23} \\ O_{31} & O_{32} & O_{33} \\ O_{41} & O_{42} & O_{43} \\ O_{51} & O_{52} & O_{53} \end{bmatrix} \tag{25}$$

$$U_{ij} = \begin{bmatrix} 3 & 3 & 3 \\ 2 & 3 & 3 \\ 2 & 3 & 3 \\ 2 & 2 & 2 \\ 3 & 3 & 2 \end{bmatrix} \tag{26}$$

$$n_{ij} = \sum_{x=1}^{i-1} J_{xo} - j = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \\ 13 & 14 & 15 \end{bmatrix} \tag{27}$$

$$\Omega_{ij} = \begin{bmatrix} 1,2,3 & 1,2,4 & 4,5,7 \\ 1,3 & 2,3,4 & 5,6,7 \\ 1,2 & 3,4,7 & 4,5,6 \\ 1,2 & 3,5 & 4,6 \\ 1,2,3 & 3,4,5 & 5,6 \end{bmatrix} \tag{28}$$

Since FJSSP is a deterministic problem, its single lowest solution exists in each situation which is termed as the Lower bound (LB). This is the point beyond which no feasible solution exists. This aspect is supported by Ho et al. [234], which states that precedence constraints indicate the sequence of the operations of a job. This is one of the basic constraints of the FJSSP which formulate the very core of the optimization paradigm. The calculations of LB is fruitful gain insight to the possible solutions of the problem. Thus, completion time of each job is minimum if all operations are assigned with minimum possible times with subjective machine assignments as pointed out in Eq. (29). Moreover, any job cannot be completed before the completion of all of its operations and minimum time in this regard can be obtained just by adding the processing times of all operations without adding any delay as pointed out in Eq. (30). In order to calculate the makespan, it cannot be lower than the longest cumulative processing time of any job as shown in Eq. (31). Calculations for LB of MFJS-2 are shown in Table 3.2, which shows that lowest possible makespan is 396. For LB calculations, minimum possible operation time is taken for each operation and entered sequentially to obtain the completion time of the pertinent job. Accordingly, the maximum time of job completion is taken as LB since problem cannot be solved before that. A Gantt chart in this regard is presented in Figure 3.1.

$$P_i' = \sum_{x=1}^{J_{io}} P_{ix}' \qquad \forall\, O_{ij} \tag{29}$$

$$P_{ij}' = \min\left(P_{ijk}\right) \quad \forall\, M_k \epsilon \Omega_{ij} \tag{30}$$

$$LB = \max_{1 \le i \le N} P_i' \tag{31}$$

**Table 3.2: Calculation of LB for MFJS-2**

| | | M1 | M2 | M3 | M4 | M5 | M6 | M7 | $P'_{ij}$ | $P'_i$ | LB |
|---|---|---|---|---|---|---|---|---|---|---|---|
| J1 | $O_{11}$ | 147 | 123 | 145 | | | | | **123** | **396** | **396** |
| | $O_{12}$ | 123 | 130 | | 140 | | | | **123** | | |
| | $O_{13}$ | | | | 150 | 160 | | 200 | **150** | | |
| J2 | $O_{21}$ | 214 | | 150 | | | | | **150** | **311** | |
| | $O_{22}$ | | 66 | 87 | 99 | | | | **66** | | |
| | $O_{23}$ | | | | | 178 | 95 | 150 | **95** | | |
| J3 | $O_{31}$ | 87 | 62 | | | | | | **62** | **267** | |
| | $O_{32}$ | | | 180 | 105 | | | 145 | **105** | | |
| | $O_{33}$ | | | | 190 | 100 | 153 | | **100** | | |
| J4 | $O_{41}$ | 87 | 65 | | | | | | **65** | **374** | |
| | $O_{42}$ | | | 250 | | 173 | | | **173** | | |
| | $O_{43}$ | | | | 145 | | 136 | | **136** | | |
| J5 | $O_{51}$ | 128 | 123 | 145 | | | | | **123** | **255** | |
| | $O_{52}$ | | | 86 | 65 | 47 | | | **47** | | |
| | $O_{53}$ | | | | | 110 | 85 | | **85** | | |

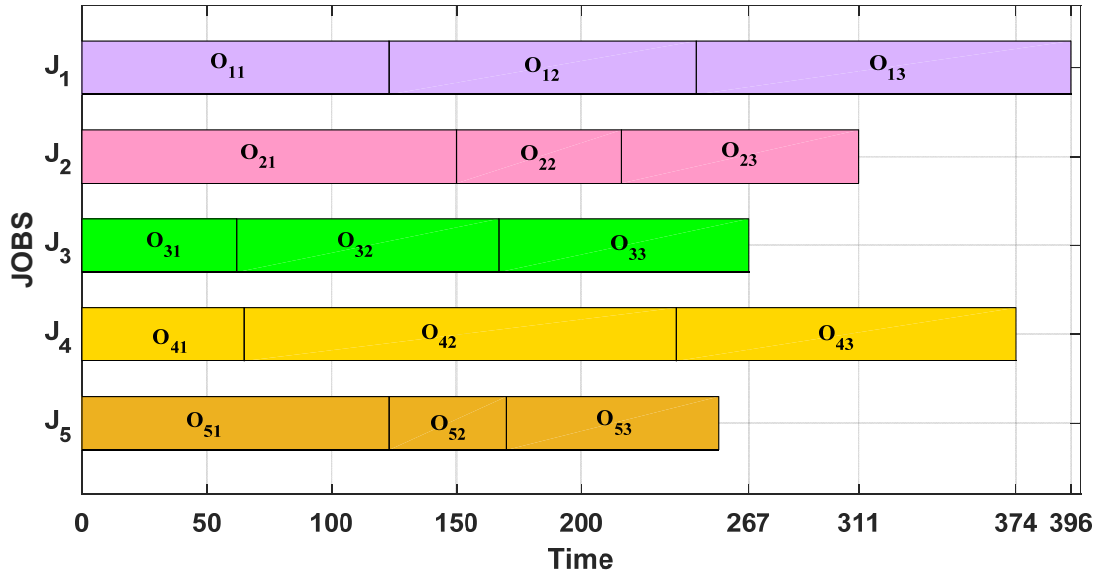


**Figure 3.1: Makespan of MFJS-2 as per LB**

## 3.4 Simulation environment

Since the problem requires extensive computational routines, MATLAB® has been used to build the simulation environment for attempting the solution. MATLAB® is released by MathWorks and is one of the most popular computational programming languages that has been used in various engineering fields. It offers smooth integration with Microsoft Excel for read / write function in an autonomous mode and this functionality has been used to automate the algorithm execution. Benchmark problems have been obtained from the online repository of Mastrolilli [279].

The simulation environment for the current study is built in following different parts as shown in Figure 3.2.

i. *Scheduler*: The scheduler reads the problem from .fjs file and transforms it into an initial schedule through the predefined chromosome structure. The parametric setting of the algorithm is also fed into the solver through Excel worksheet.

ii. *GA solver*: The solver intakes a seed chromosome and generates initial population through stipulated techniques. The GA routine then continues to converge until the termination criteria is met. Once the convergence is achieved the best chromosome is sent back to the scheduler and the same is translated into a Gantt Chart.

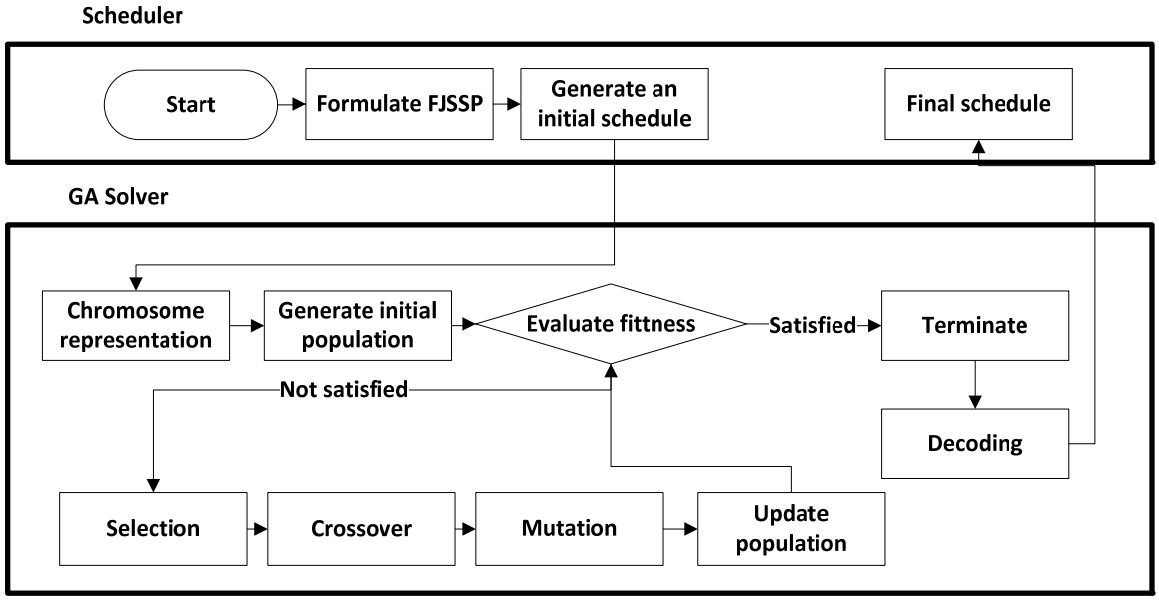


**Figure 3.2: Simulation environment for solving FJSSP**

# 4 Chapter 4 – Proposed Algorithms

## 4.1 Introduction

It has been established adequately through extensive literature review that GA has been used as an effective technique to solve the combinatorial optimization problems. Although lot of research has been conducted to evolve different GA routines for improving the solution quality, areas for further research have been identified in section 2.8. This chapter provides details of the proposed algorithms for solving FJSSPs for optimization of makespan on selected benchmarks. The work presented in this chapter has been expanded from already published papers of the author [280, 281].

This work proposes two different approaches for solving the FJSSP as follows.

i. GA integrated with scheduling rules: The two-part problem of FJSSP is addressed using a hybrid approach of GA and scheduling / dispatching rules.
ii. Pure GA based technique: A pure GA based approach is proposed for solving the FJSSP in a holistic manner.

## 4.2 GA with Priority Rules (GA-PR)

Since there are two parts of the FJSSP, GA-PR is proposed, whereby the assignment part is solved by GA and scheduling part is solved by using five different dispatching rules. GA is used to search for the assignment problem after which the dispatching rules are used for the selected assignment scheme for finding the best makespan. The dispatching rule that gives best makespan is used finally to generate the solution. Figure 4.1 provides a flowchart of GA-PR and the following sections describe the algorithm in detail.

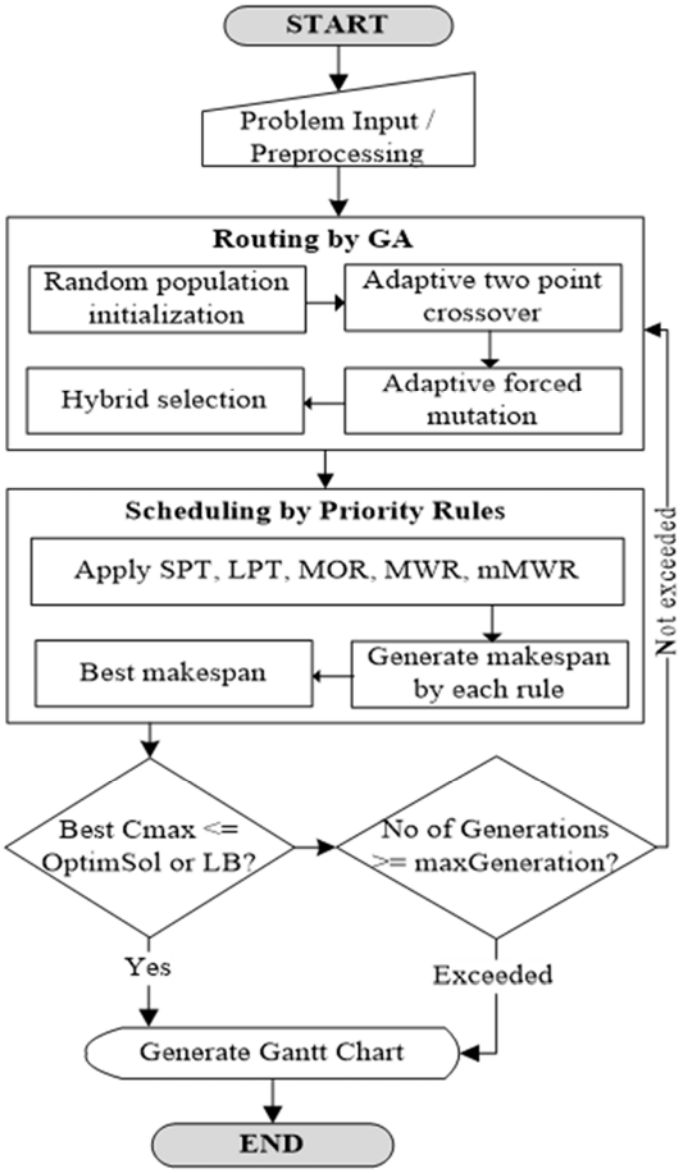


**Figure 4.1: The flowchart of GA-PR**

### 4.2.1 Solution of assignment problem by GA

The decision regarding assignment of operations to the available machines is made through application of GA. The chromosome is encoded using the scheme of Zhang et al. [157]. Considering the SFJS 6 problem as presented in Table 4.1. This is a partial flexible problem e.g. $O_{11}$ can only be performed on $M_1$ and $O_{12}$ can be performed on both $M_1$ and $M_2$, but not on $M_3$. A sample encoding is presented in Figure 4.2, where only $M_1$ is shown below $O_{11}$ since it can only be performed on this machine. The operation $O_{32}$ is performed on $M_3$ (blue gene) which is the second machine from the available set of possible machines i.e. $M_2$ and $M_3$. The operation which is assigned to a said machine is shown in black (circled) and other machines are shown in

red. In this way, the assignment of operations to available machines can be encoded in the form of a chromosome.

**Table 4.1: Fattahi SFJS6**

| Job | Operation | Processing Time | | |
|---|---|---|---|---|
| | | $M_1$ | $M_2$ | $M_3$ |
| $J_1$ | $O_{11}$ | 17 | ∞ | ∞ |
| | $O_{12}$ | 40 | 130 | ∞ |
| | $O_{13}$ | ∞ | 50 | 60 |
| $J_2$ | $O_{21}$ | 30 | ∞ | ∞ |
| | $O_{22}$ | 150 | 160 | ∞ |
| | $O_{23}$ | ∞ | ∞ | 70 |
| $J_3$ | $O_{31}$ | 50 | 60 | ∞ |
| | $O_{32}$ | ∞ | 170 | 180 |
| | $O_{33}$ | ∞ | 90 | 100 |

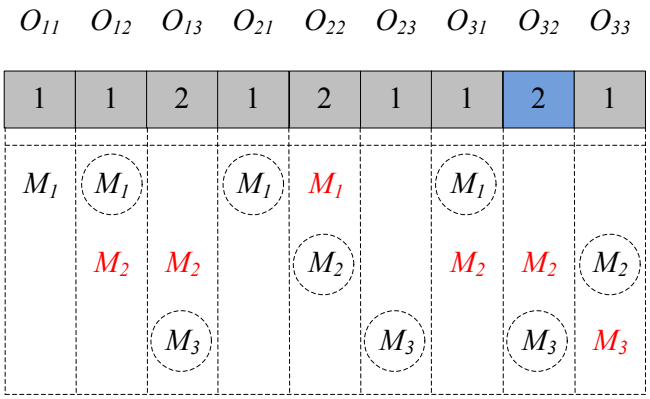


**Figure 4.2: A sample chromosome encoding**

Figure 4.3 presents another chromosome following similar encoding scheme, whereby $O_{32}$ will now be performed on first of the available machines i.e. $M_2$. Here, all assignment options have been triggered to provide clarity of the representation, e.g, for $O_{12}$, the available option of $M_2$ is now selected from the available options of $M_1$ and $M_2$. This will change the spectrum of the scheduling problem since the operation will now be completed in 130 units of time instead of 40 units.

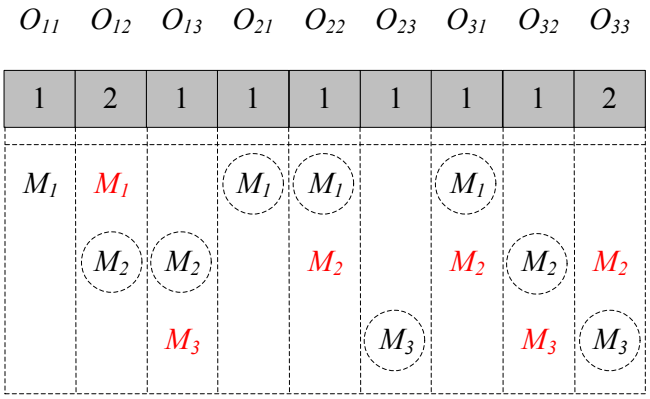


**Figure 4.3: Another sample chromosome**

In order to generate a population, random initialization of chromosomes is done as presented in Figure 4.4. The random initialization evaluates whether all operations of all jobs are assigned to the available machines and terminates when this condition is met. The loop continues to run until the required number of chromosomes are generated.

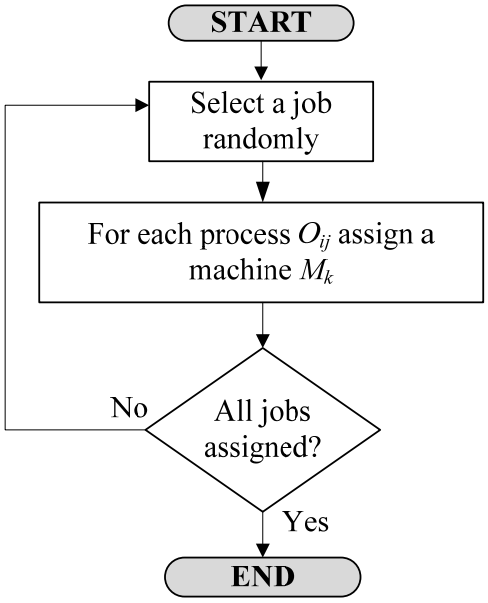


**Figure 4.4: Random population initialization**

Two-point crossover has been used for generating new offspring. First, two parent chromosomes are selected randomly from the current population and two crossover points are selected randomly. The parents are then swapped against these two points to generate two children. Since the assignment on the machines remains the same after crossover, this method restricts generation of infeasible children. Figure 4.5 provides an example of TPX where color-coded parents are shown to produce children having inherited characteristics.

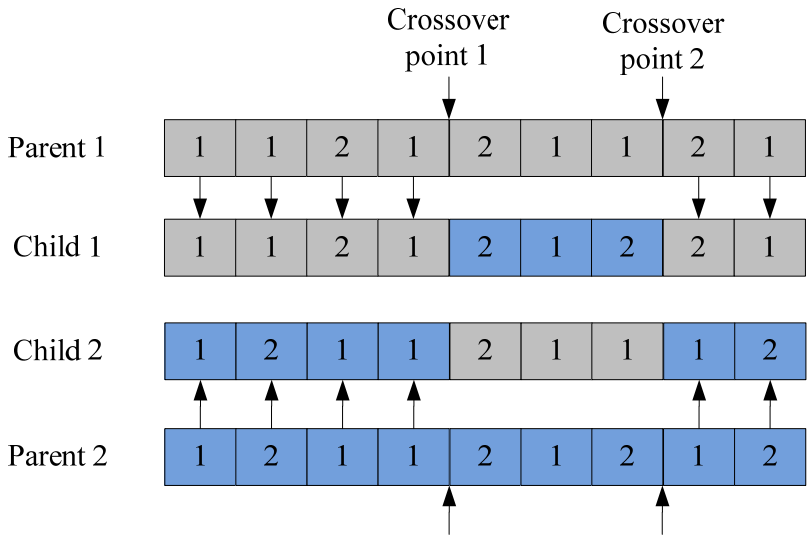


**Figure 4.5: An example of TPX**

Crossover is performed over a selected population which is controlled by crossover probability ($P_c$). This value is traditionally taken as 0.8 – 0.9 to ensure a good evaluation throughout the search space. However, since the probability is selected at the start of the algorithm, the operator tends to remain stagnant during the algorithm execution. Hence, adaptive probability is proposed in GA-PR, whereby the probability starts increasing from the initial value as the convergence is achieved in order to ensure population diversity as depicted in Eq. 32. This helps the algorithm to reduce the chances of trapping in local minima. A flowchart of the TPX is presented in Figure 4.6.

$$P_c = \frac{\text{avg } F_t}{\max F_t} \tag{32}$$

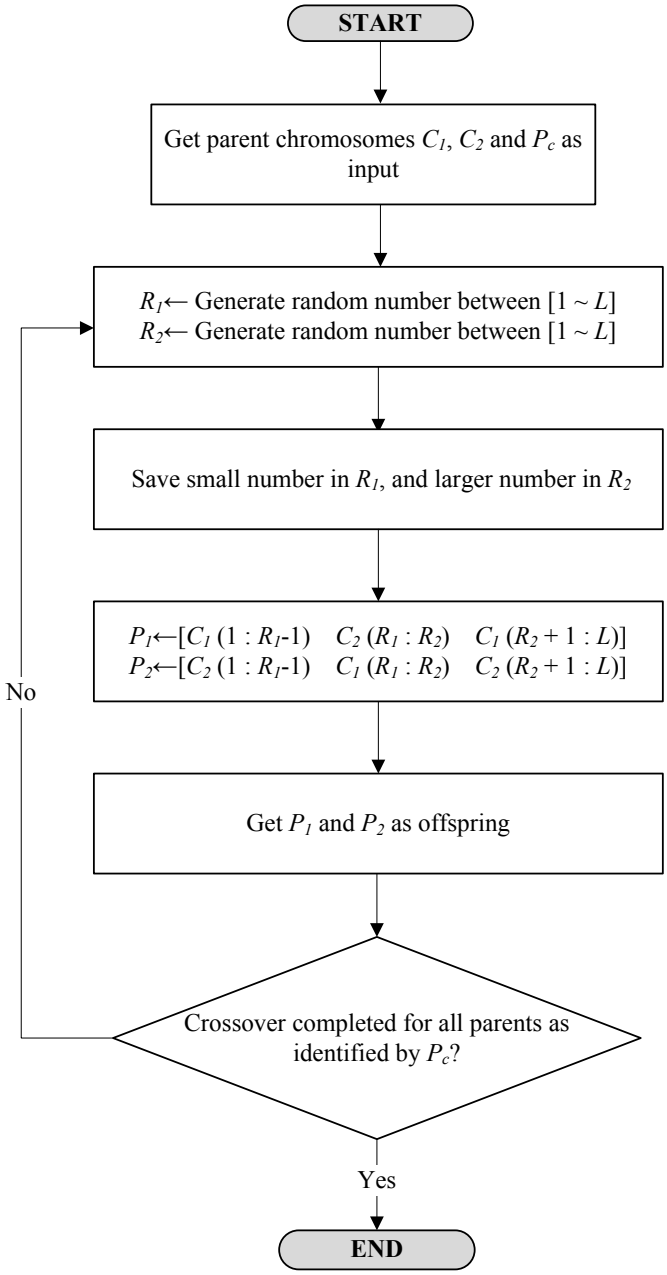


**Figure 4.6: Flowchart of TPX**

A compulsory mutation (CM) technique is proposed in GA-PR which has been evolved through the concept of flexibility in machine assignment. The flexibility in genes depends upon the number of operations a machine can undertake i.e. the machine must be capable to complete at least two operations ($U_{ij} > 1$). This level of flexibility has also been indicated in the standard benchmarks and in corresponding .fjs files and the same is calculated as follows.

$$f = \frac{\sum_{i=1}^{N} \sum_{j=1}^{Jio} U_{ij}}{L} \tag{33}$$

The problem flexibility introduces computational complexity on one side and possibility of producing better solutions on the other side. Hence it is important to evaluate the overall options offered by problem flexibility in an adequate manner. In view of this fact, following propositions are made for the CM operator.

i. *CM Rule 1:* Only flexible genes must be mutated (if available), instead of random genes; since random gene mutation will return same value as no other option is available or otherwise return an infeasible solution.

ii. *CM Rule 2:* Once the gene with flexibility is selected, the already stored option in that gene must be changed to another value and same value return will not be allowed.

Once the combination of above schemes is implemented, flexibility of mutated gene is evaluated in a better manner. Figure 4.7 provides an example of the CM operator. The operation $O_{31}$ is selected for mutation out of the possible options of $O_{12}$, $O_{13}$, $O_{22}$, $O_{31}$, $O_{32}$ and $O_{32}$ (shown in blue). Rest of the genes will not be selected for mutation as per CM Rule 1 since they don't offer flexibility. Now, $O_{31}$ can be performed on $M_1$ and $M_2$. As per CM Rule 2, $M_2$ cannot be selected again since it is already selected, hence $M_1$ is selected and the updated gene is shown in red.

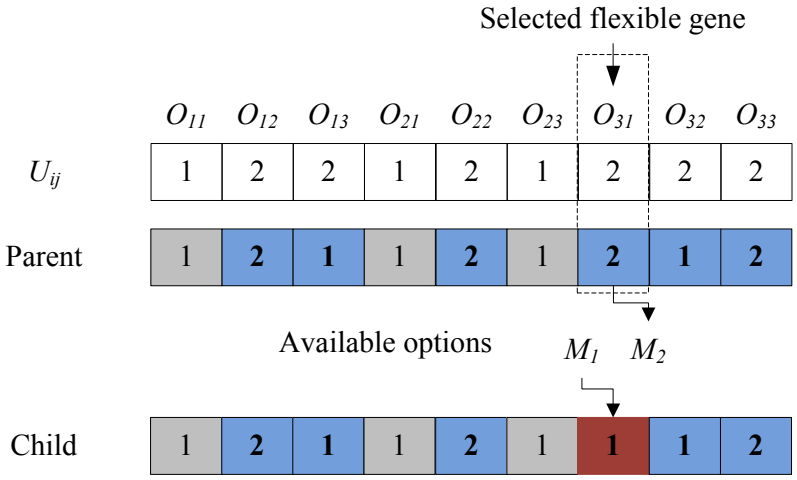


**Figure 4.7: An example of CM**

As also pointed out in TPX discussion, mutation operator is also performed on some randomly selected parents which are similarly controlled through probability of mutation ($P_m$). Traditionally, this vale is taken as 0.1 – 0.2. Since the value is selected in the start of the algorithm, it remains the same throughout the generations and loses effectivity regarding search

space exploration towards convergence. A similar adaptive mutation probability is proposed here as per Eq. 25 which increases the mutation probability as per the maximum population fitness value and increases the $P_m$ value as convergence is achieved. This procedure incorporates further diversity in the population. A flowchart of CM procedure is presented in Figure 4.7.

$$P_m = \frac{\min F_t}{\max F_t} \tag{34}$$

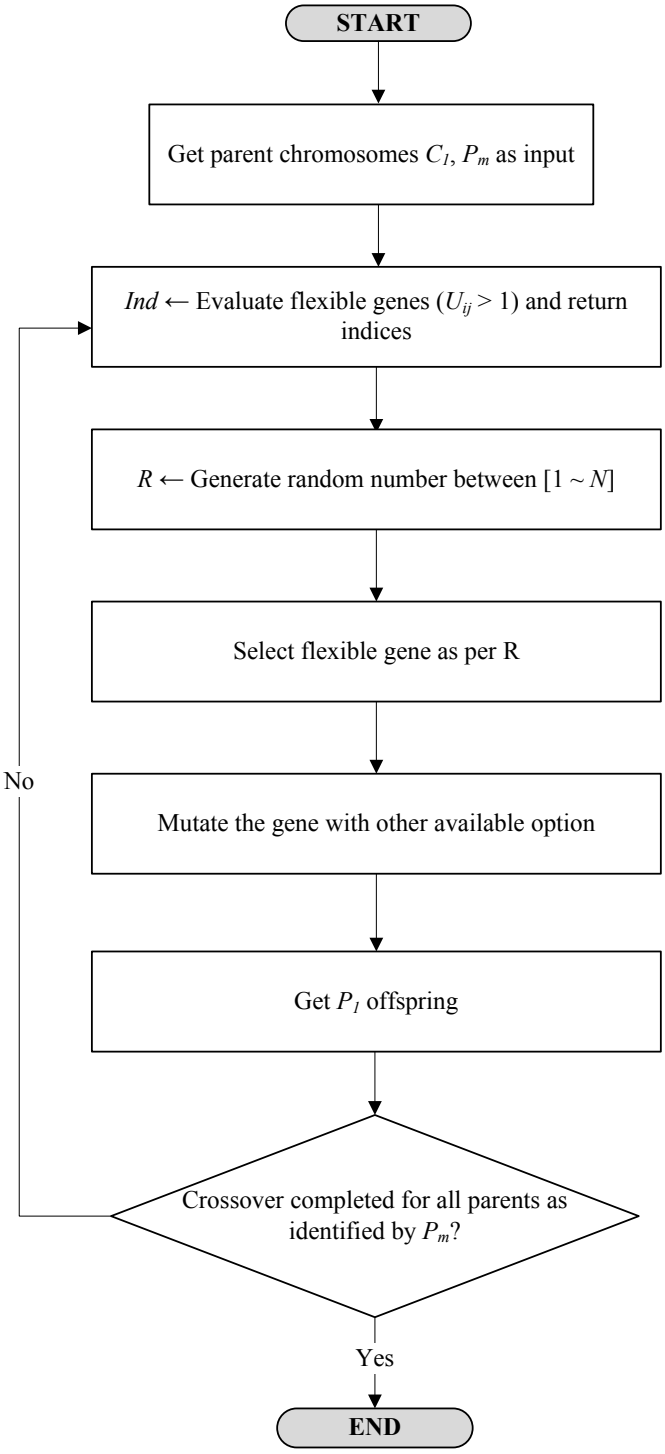


**Figure 4.8: Flowchart of CM**

The algorithm continues to execute the recombination operators until the required number of parents get crossover as per $P_c$ and mutation as $P_m$. This generates a new population pool which is greater than the allowed population and which contains both parents and the offspring. The rule of fitness survival is now applied on the whole population. The selection mechanism is

hybrid mixture of elitism and roulette wheel techniques in a ratio of 40:60, respectively. These procedures are described below.

a. *Elitism*: This procedure inputs population comprising of parents / offspring and sorts it in descending order. Best chromosomes are then selected depending upon the fitness value. The number of chromosomes selected through elitism is generally kept low because their increased number forces the algorithm to converge in premature manner and / or trap in local solution. Its flowchart is presented in Figure 4.9.

b. *Roulette wheel*: This procedure intakes cumulative fitness of whole population and generates a random number in its stipulated range. The corresponding chromosome is then transported to the next generation irrespective of its own fitness. This operator ensures population diversity, and a higher number of chromosomes are selected through this procedure. Its flowchart is presented in Figure 4.10.

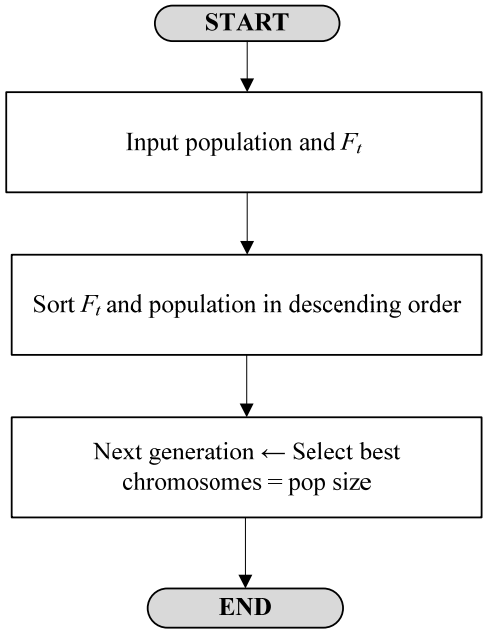


**Figure 4.9: Flowchart of elitism**

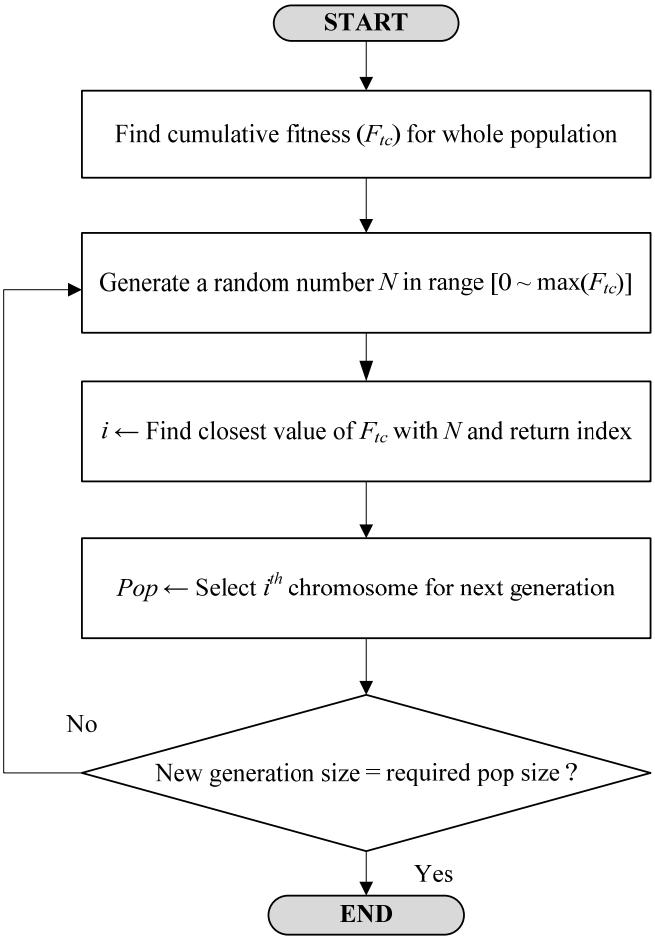


**Figure 4.10: Flowchart of roulette wheel**

### 4.2.2 Solution of scheduling problem by priority rules

Once the assignment problem is addressed by GA, the scheduling problem is undertaken by using priority rules. The selected rules for implementation are explained below.

a. *SPT*: This rule uses the processing time of jobs to schedule them based upon the least time required i.e., the job requiring minimum time will be scheduled first. This is modeled as defined in Eq. (26). Considering the SFJS-6 (Table 4.1), assume that all operations which can be completed on $M_1$ are assigned to it (i.e. $O_{11}$ = 17, $O_{12}$ = 40, $O_{21}$ = 30, $O_{22}$ = 150, $O_{31}$ = 50). Since, SPT is to be applied, the operation with minimum $P_{ik}$ i.e., $O_{11}$ will be scheduled at priority.

$$P_{ik} = min_{j=1}^{\zeta}[p_{ij}] \quad (35)$$

b. *LPT*: This rule uses the processing time of jobs to schedule them based upon the longest time required i.e., the job requiring maximum time will be scheduled first. This is modeled as defined in Eq. (27). Considering the SFJS-6 (Table 4.1),

assume that all operations which can be completed on $M_2$ are assigned to it (i.e. $O_{12}$ = 130, $O_{13}$ = 50, $O_{22}$ = 160, $O_{31}$ = 60, $O_{32}$ = 170, $O_{33}$ = 90). Since, LPT is to be applied, the operation with minimum $P_{ik}$ i.e., $O_{32}$ will be scheduled at priority.

$$P_{ik} = max_{j=1}^{\zeta}[p_{ij}] \quad (36)$$

c. *MOR*: This rule uses the remaining number of operations of the job as the decisive factor and schedules the job with the greatest number of remaining operations first. The remaining operations in this regard are calculated as defined in Eq. (28). Considering the SFJS-6 (Table 4.1), it is clear that all jobs have equal number of operations, i.e., 3. Let's assume that an assignment arises such that all three operations of $J_1$ and 2 operations of $J_2$ have already been scheduled. Since the greatest number of operations will be from $J_3$, it will be scheduled based upon MOR.

$$OR_{ij} = J_{io} - j + 1 \quad (37)$$

d. *MWR*: This rule uses the most remaining work for scheduling and prioritizes the job with most work. The remaining work in this regard is evaluated as defined in Eq. (29). Here, *k'* is the index assigned to the machine processing operation $O_{ix}$. Irrespective of the operations, the remaining work in terms of time is given priority in MWR. Let's assume that an assignment arises such that all three operations of $J_1$ and 2 operations of $J_2$ have already been scheduled. Even though all operations of $J_3$ are remaining, this rule will schedule next operation of $J_2$ since [(130 + 160 + 170 + 90 = 550) > (60 + 70 + 180 + 100 = 410)].

$$WR_{ij} = \sum_{x=j}^{J_{io}} P_{ixk'} \quad (38)$$

e. *mMWR*: The time required to process any operation on a said machine can be identified once the operation is assigned, because the operation time differs depending upon the machines. Thus, one must decide the assignment to execute the MWR. Here, proposition is made to modify the MWR to use average time of all process of an operation which can be undertaken on different machines. In this way, the rule becomes autonomous of the assignment. This is calculated as described in Eq. (30). An explanation in this regard is presented in Table 4.2. $O_{12}$ and $O_{13}$ are scheduled by MWR as 11, however the mMWR generates an answer of 8.25.

$$mWR_{ij} = \sum_{x=j}^{J_{io}} avgP_{ix} \quad (39)$$

**Table 4.2: Example Instance for mMWR**

| Job | | $M_1$ | $M_2$ | $M_3$ | $M_4$ | avgT |
|---|---|---|---|---|---|---|
| $J_1$ | $O_{11}$ | 1 | 3 | 4 | 1 | **2.25** |
| | $O_{12}$ | 3 | 8 | 2 | 1 | **3.50** |
| | $O_{13}$ | 3 | 5 | 4 | 7 | **4.75** |
| $J_2$ | $O_{21}$ | 4 | 1 | 1 | 4 | **2.50** |
| | $O_{22}$ | 2 | 3 | 9 | 3 | **4.25** |
| | $O_{23}$ | 9 | 1 | 2 | 2 | **3.50** |
| $J_3$ | $O_{31}$ | 8 | 6 | 3 | 5 | **5.50** |
| | $O_{32}$ | 4 | 5 | 8 | 1 | **4.50** |

**Table 4.3: Explanation of mMWR**

| **Operation** | $O_{11}$ | $O_{12}$ | $O_{13}$ | $O_{21}$ | $O_{22}$ | $O_{23}$ | $O_{31}$ | $O_{32}$ |
|---|---|---|---|---|---|---|---|---|
| Assignment | $M_1$ | $M_2$ | $M_1$ | $M_4$ | $M_1$ | $M_1$ | $M_3$ | $M_2$ |
| $T_{ijk}$ | 1 | **8** | **3** | 4 | 2 | 9 | 3 | 5 |
| MWR→ | | **11** | | | | | | |
| avg $T_{ijk}$ | 2.25 | **3.50** | **4.75** | 2.50 | 4.25 | 3.50 | 5.50 | 4.50 |
| mMWR→ | | **8.25** | | | | | | |

The algorithm calculates makespan based upon all five rules for each assignment and returns best makespan along with the used rule. The termination criteria are set as follows:

a. Best known solution as identified by [282] or less is obtained.

b. Total generations are exhausted.

The output of GA-PR is evolved as a solution of the scheduling problem. Fitness function is used for the purpose of evaluating whether the solution is good or bad. Figure 4.11 presents a process flow of fitness function used in GA-PR. The fitness is the total minimum time ($C_{max}$) to finish all operations of all jobs which is found by taking inverse of $C_{max}$.

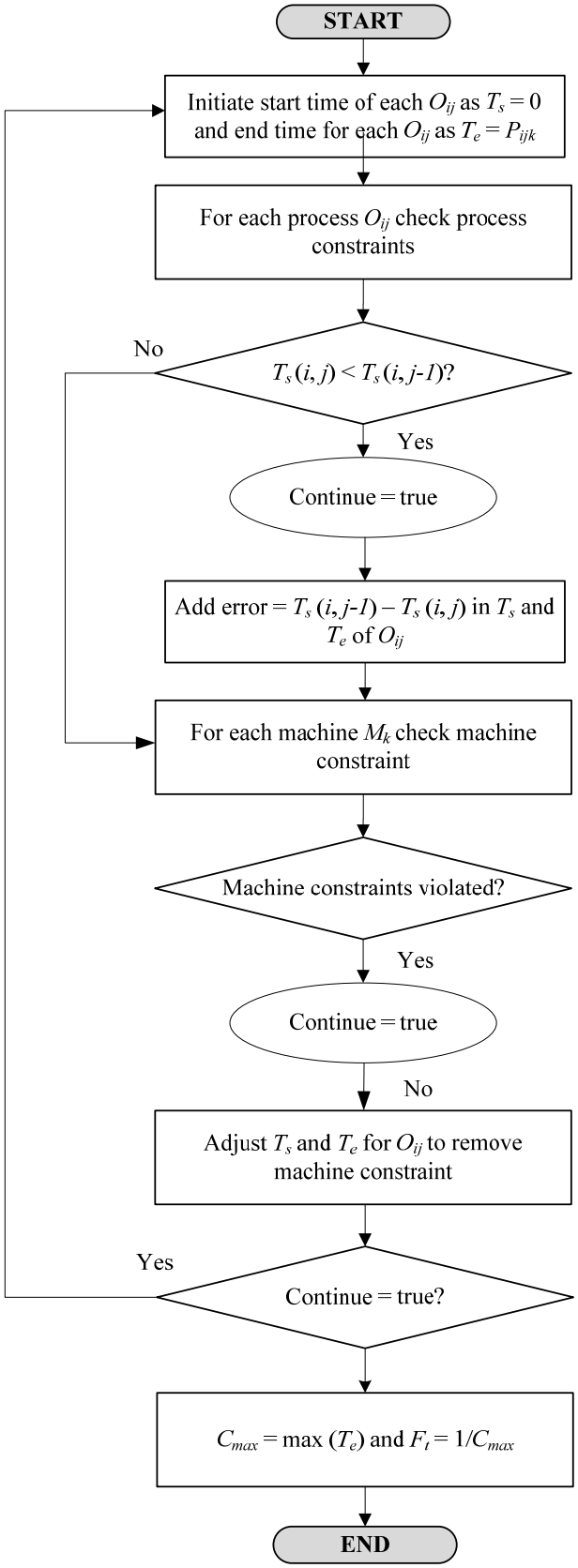


**Figure 4.11: Fitness function**

## 4.3 GA with iterative diversification technique (GA-IDT)

Although FJSSP can be adequately handled with GA-PR, the approach is half-heuristic and half-metaheuristic. One hand, this architecture enables to evaluate the assignment part of the problem in an effective way, the setting also limits the scheduling part of the problem to be handled with priority rules. As evaluated earlier, the priority rules can address only a specific nature of instance in an optimal way. This setting, while providing ease of implementation and simplification in interpretation, has pitfalls in the solution of bigger problems as evident from the experiments.

Apropos, a pure GA based approach is now proposed for addressing the complete landscape of the complex FJSSP in a holistic manner. This approach will undertake the assignment and routing problem in a parallel manner and will attempt to solve the problem simultaneously, such that space of both parts of the problem is evaluated by means of a meta-heuristic.

### 4.3.1 The need for IDT

All population-based algorithms require a sufficient set of candidate solutions to evolve and produce possible best solutions and this basic ingredient of population is common amongst this family of algorithms, although described and implemented in conceptually different schemes. As an example, the PSO algorithm may be considered [283, 284] where a population of particles is maintained in a similar way, however fitness is estimated in a different manner.

Population can be restrained in a certain area of the search space or it can be dispersed in the far spread areas. When concentrated in a certain area, it generally reflects a region of better solutions after convergence, but it may also be a local optimum very far away from the global one. On the other hand, well spread population indicates that the algorithm convergence is not met, and solution of equally good quality are being found in the far away areas.

This is explained schematically in the Figure 4.12. The figure shows a search space for an arbitrary multi-solution optimization problem. The black dots present the candidate solutions, and the red dot represents the optimal solution. It is evident that the population is spread in the whole search space, however there are more solutions in the left side of the search space. The search algorithm (which in case is GA) initializes its population in a random manner and searches for a solution in four areas of the space. The concept of diversity starts right here, since increasing the number of initial solutions will increase the possibility of evaluating more search space in a

single time, however it will come at the cost of computational effort and will also lead to many similar solutions.

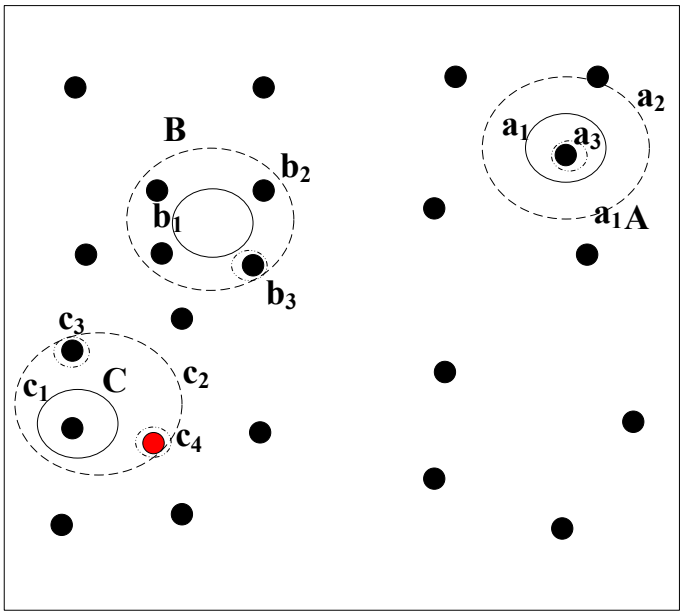


**Figure 4.12: A schematic representation of the search space**

Let's consider the situation at reference location 'A'. The algorithm starts to search ($a_1$), finds only one solution, and then expands search to evaluate more space around its area ($a_2$) , but doesn't find anything. Similarly, it also sees no solution in intensification ($a_3$). The solution presented as ($a_3$) is saved as best in this case. This situation offers equal chances of finding solutions for both cases of intensification and diversification.

In case 'B', the initial search location ($b_1$) has no solution. When the solution space is expanded to ($b_2$), four solutions are found, and the algorithm converges to the solution selected as ($b_3$). In this scenario, if diversification is not carried out, no solution could be found inside the initial space. The best solution in this case is saved as ($b_3$).

In case 'C', the initial search location finds one solution ($c_1$). In order to evaluate further, the search location is expanded ($c_2$). Lets consider that the maximum number of solutions that can be saved in an iteration in this example are two and we find ($c_3$). However, as evident from the figure we also have the optimal solution ($c_4$) but cannot find it since the memory register has already been filled with best solutions. Now, to overcome this aspect, we intensify the search and find ($c_4$) which is the better solution form ($c_3$) so it is discarded. On comparison of the all solutions mentioned above, it is found that ($c_4$) is the optimal solution.

A well-spread population is required to evaluate the enormous search space offered by the FJSSPs. This diverse population guarantees that all possible locations of solutions are evaluated in an assertive manner; however, spreading the population unconventionally requires exceptional computational resources and it may also restrain the algorithm to converge since the algorithm will continue to evaluate the search space. On the other hand, restraining the population to a smaller number will result in insufficient evaluation of the search space and there will a possibility that better solutions are not found. It is therefore a factor of prime importance in the paradigm of evolutionary computation that population diversity is managed in a balanced manner such that search space is evaluated in an efficient way and unnecessary / recursive search is prohibited.

With regards to GA, it is also a known fact that the algorithm tends to get trapped in local minima. This is again due to fact that the population tends to converge towards some elitist solutions, and it seems that the optimal point has been achieved; however, better solutions are available in the neighborhood or overall search space. It is pertinent to mention here that meta-heuristic evolutionary algorithm does not guarantee a global optimum, but they provide a good solution that is near optimal.

It is a matter of research to propose ways whereby the algorithm can produce near optimal solutions as close as possible to the global optima. With regards to the local minima trap, the algorithm is traditionally coupled with local search techniques of Tabu Search (TS) [236] or Simulated Annealing (SA) [285]. Although these are proven efficient techniques, they require additional implementation effort and corresponding computational cost. Another possibility in this regard to run the algorithm in extended generations [159]; however this option also requires computational efforts over extended periods of time.

GA-IDT is purposefully built to propose mechanisms to restrict the above-mentioned challenges in the conventional GA approach. The algorithm proposes a strategy to manage diversification (to evaluate the whole search space in a sufficient manner) and intensification (to dig down a promising area of search space). Figure 4.13 presents the basic information and logical flowchart of IDT.

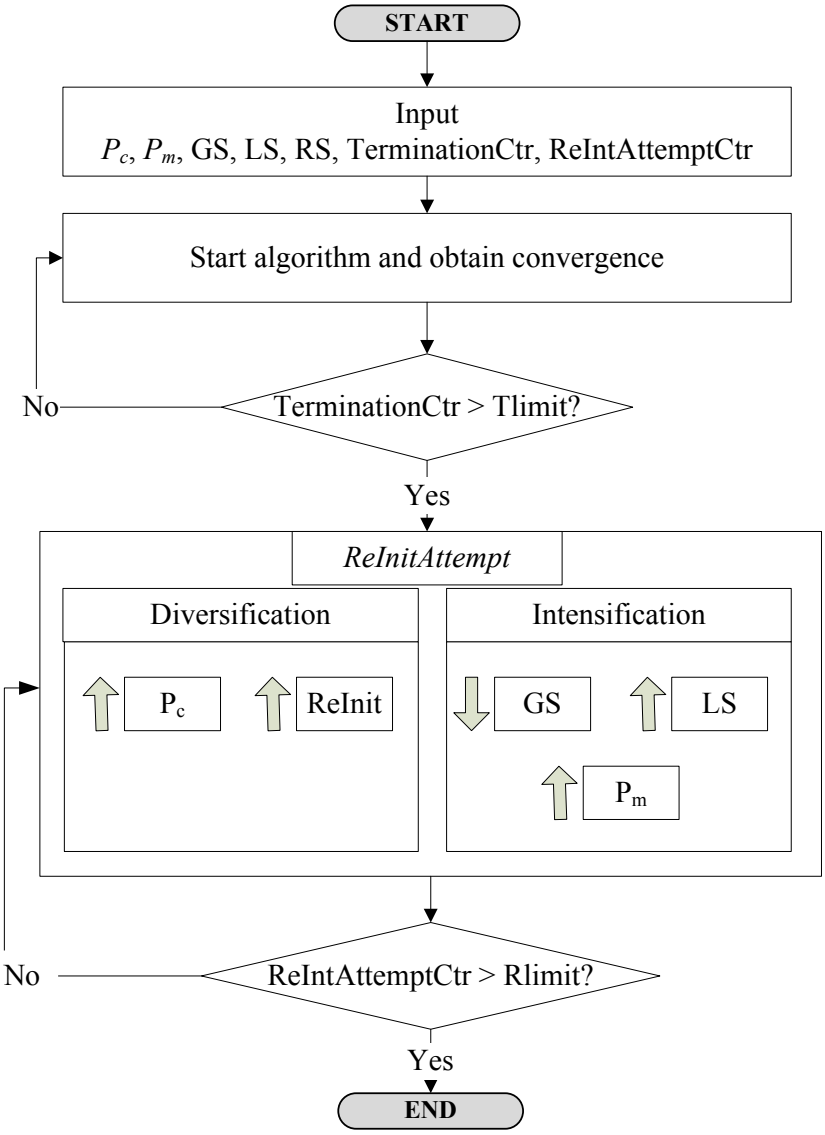


**Figure 4.13; Flowchart of IDT**

### 4.3.2 Architecture of GA-IDT

GA-IDT is built in four layers on the similar scheme of Figure 3.2. The simulation for solution search is executed in following steps.

a. Step-1: Input parameters in MS Excel file: Different parameters are preset into the MS Excel sheet. Details are provided in Input Layer of the algorithm

b. Step-2: MATLAB ® is opened and source code file is loaded and executed.

c. Step-3: The algorithm reads the configuration settings and executes the algorithm accordingly. It converges as per stipulated criteria and terminates to produce $C_{max}$ and Gantt chart.

d. Step-4: The algorithm saves the results in same MS Excel file for future use.

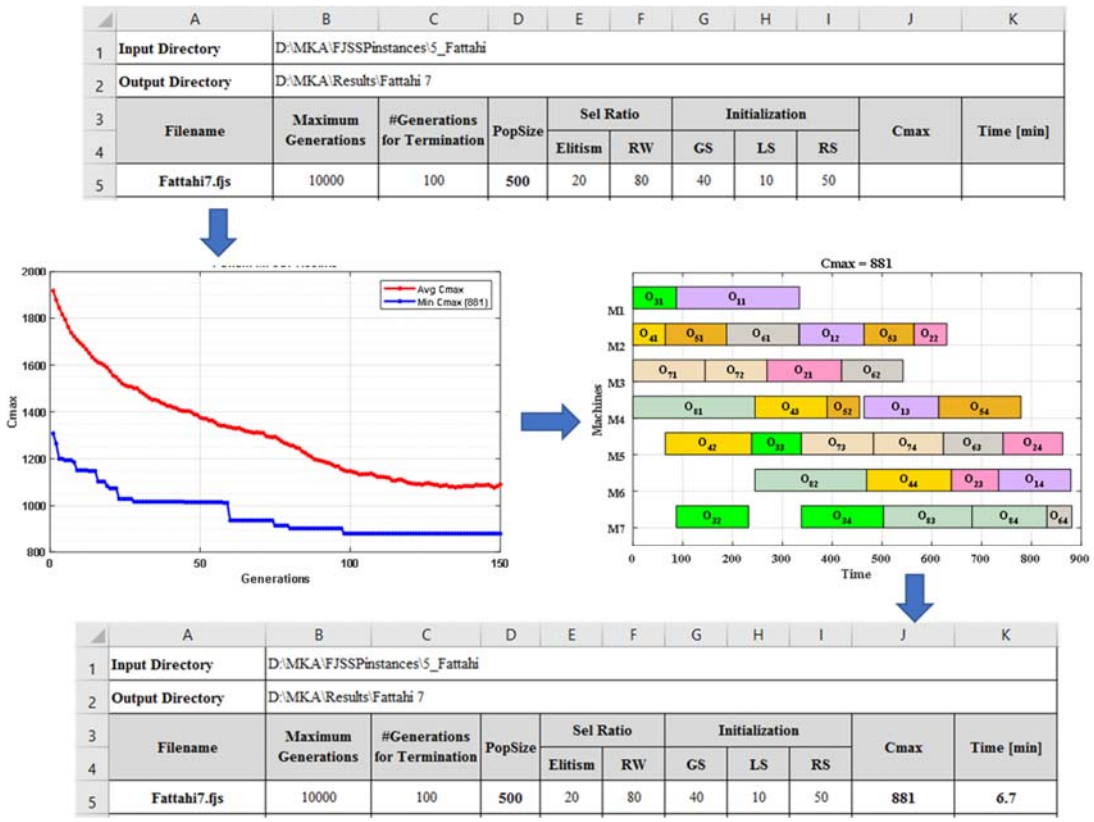


**Figure 4.14: Procedure for GA-IDT execution**

The algorithm is presented in Figure 4.15 and its functioning is presented in the following sections.

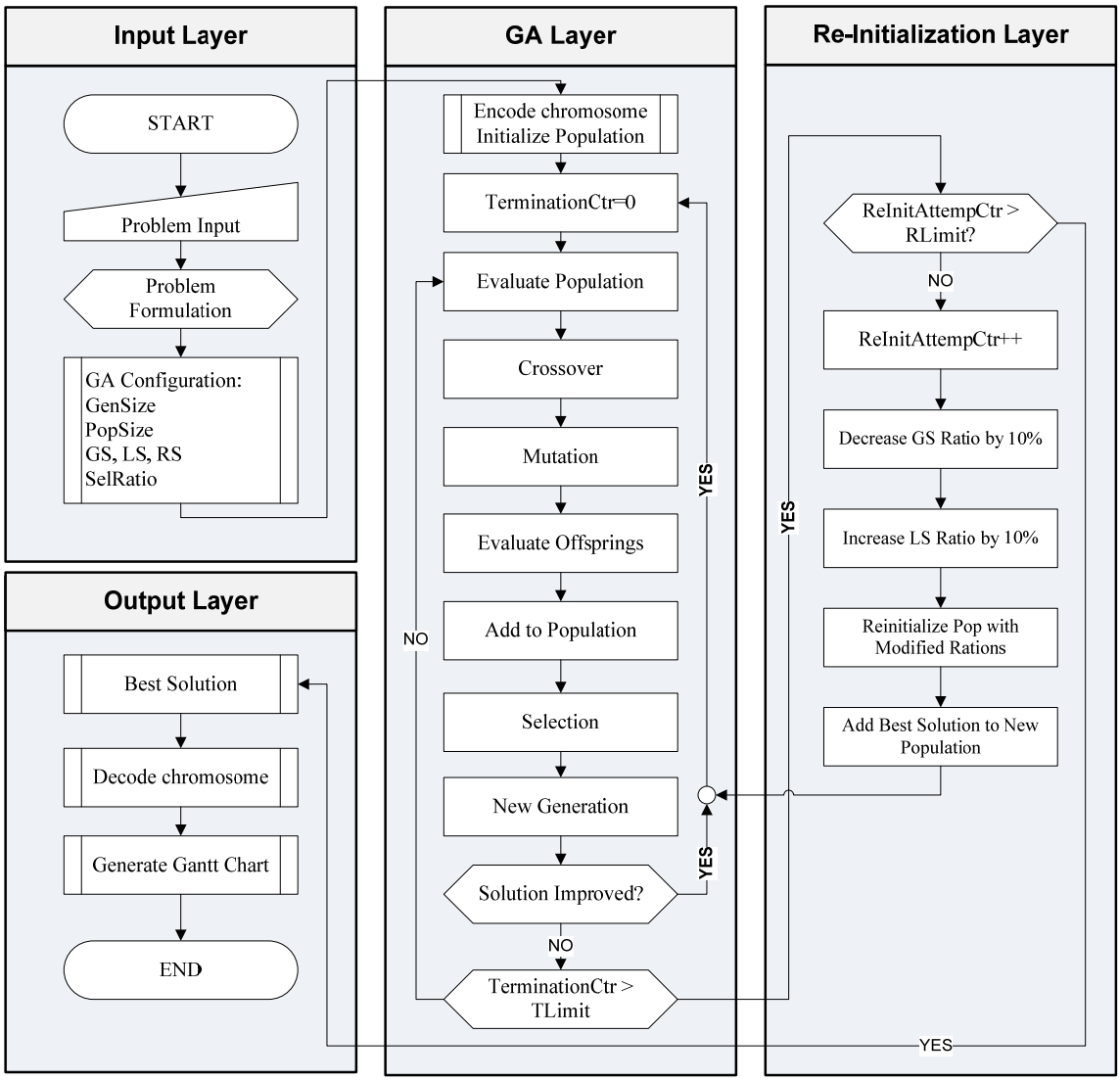


**Figure 4.15: Flowchart of GA-IDT**

### 4.3.3 Layer 1: input

The input layer takes parametric settings of the algorithm which are required for algorithm running through a pre-saved MS Excel file. This instrument is extremely helpful to conduct experiments over several settings of the algorithm such that settings / problems can be fed into next rows and the algorithm reads the settings before the start of each problem.

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **Input Directory** | D:\MKA\FJSSPinstances\5_Fattahi | | | | | | | | |
| 2 | **Output Directory** | D:\MKA\Results\Fattahi 7 | | | | | | | | |
| 3 | **Filename** | **Maximum Generations** | **#Generations for Termination** | **PopSize** | **Sel Ratio** | | **Initialization** | | | **Cmax** |
| 4 | | | | | **Elitism** | **RW** | **GS** | **LS** | **RS** | |
| 5 | **Fattahi7.fjs** | 10000 | 100 | **500** | 20 | 80 | 40 | 10 | 50 | |

Figure 4.16: A sample input MS Excel sheet

presents a snapshot of input file and its different sections are explained below.

a. *Input directory*: contains the path of '.fjs' files to be read before algorithms start

b. *Output directory*: contains the path of the folder where the convergence pattern and Gantt chart is saved in '.fig' format. The algorithm is capable of reading the problem from '.fjs' file and produce population.

c. *Filename*: Contains the name of '.fjs' file which is to be solved. Different problems can be fed into different rows and different parametric setting can be saved for experimentation.

d. *Maximum generations*: The maximum number of generations are saved here, after the completion of which, the algorithm will terminate.

e. *Number of generations for termination*: This counter continues to grow if no improvement is found in subsequent generations and is reset to zero once an improvement is found. The algorithm will terminate once the upper limit of this counter is hit.

f. *PopSize*: The population size to retain in each iteration.

g. *SelRatio*: The ratio of elitism and roulette wheel strategies to be used in the algorithm.

h. *Initialization*: The percentage of global, local and random (GS, LS, RS) are saved here.

i. $C_{max}$: The algorithm saves the best solution here after convergence.

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **Input Directory** | D:\MKA\FJSSPinstances\5_Fattahi | | | | | | | | |
| 2 | **Output Directory** | D:\MKA\Results\Fattahi 7 | | | | | | | | |
| 3 | **Filename** | **Maximum Generations** | **#Generations for Termination** | **PopSize** | **Sel Ratio** | | **Initialization** | | | **Cmax** |
| 4 | | | | | **Elitism** | **RW** | **GS** | **LS** | **RS** | |
| 5 | **Fattahi7.fjs** | 10000 | 100 | **500** | 20 | 80 | 40 | 10 | 50 | |

**Figure 4.16: A sample input MS Excel sheet**

### 4.3.4 Layer 2: GA

This layer takes the input parameters from the input layer and executes the GA routine. The representation of MSOS proposed by Zhang [157] has been used in GA-IDT. Figure 4.17 presents the Fattahi 12 problem and its conversion into an initial chromosome. The MS part is encoded as already explained in section 4.2.1. For further illustration, the first job ($J_1$) is

comprised of three (03) operations i.e. $O_{11}$, $O_{12}$ and $O_{13}$. If we consider $O_{12}$, the operation can be performed on $M_1$, $M_2$ and $M_4$, out of which it is currently being performed on $M_4$ which is the third machine from the available machines. Hence, 3 is assigned to the respective gene. Similarly, the machine assignment is undertaken until all machines are assigned. The OS part defines the scheduling sequence for the assigned machines. The value of '1' represents that the stated operation belongs to the first job. Since there are three operations of the first job, '1' is repeated three times such that for the first time it represents $O_{11}$, for the second time it represents $O_{12}$ and for the third time it represents $O_{13}$. Similar logic is applicable to other indices of the OS part. The initialization routine continues until all operations are assigned and scheduled according to the defined representation and a joint MS-OS based chromosome is formulated as shown in Figure 4.18.

| | | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ | $M_6$ | $M_7$ | | MS | OS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| J1 | $O_{11}$ | 147 | 123 | 145 | | | | | | 1 | 1 |
| | $O_{12}$ | 123 | 130 | | 140 | | | | | 3 | 1 |
| | $O_{13}$ | | | | 150 | 160 | | 200 | | 3 | 1 |
| J2 | $O_{21}$ | 214 | | 150 | | | | | | 2 | 2 |
| | $O_{22}$ | | 66 | 87 | 99 | | | | | 1 | 2 |
| | $O_{23}$ | | | | | 178 | 95 | 150 | | 2 | 2 |
| J3 | $O_{31}$ | 87 | 62 | | | | | | | 2 | 3 |
| | $O_{32}$ | | | 180 | 105 | | | 145 | | 2 | 3 |
| | $O_{33}$ | | | | 190 | 100 | 153 | | | 2 | 3 |
| J4 | $O_{41}$ | 87 | 65 | | | | | | | 2 | 4 |
| | $O_{42}$ | | | 250 | | 173 | | | | 1 | 4 |
| | $O_{43}$ | | | | 145 | | 136 | | | 2 | 4 |
| J5 | $O_{51}$ | 128 | 123 | 145 | | | | | | 1 | 5 |
| | $O_{52}$ | | | 86 | 65 | 47 | | | | 3 | 5 |
| | $O_{53}$ | | | | | 110 | 85 | | | 2 | 5 |

**Figure 4.17: Conversion of problem into MS and OS parts of chromosome**

| MS | | | | | | | | | | | | | | | OS | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3 | 3 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 3 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 4 | 4 | 4 | 5 | 5 | 5 |

**Figure 4.18: An example chromosome**

Since the population consists of many chromosomes, the initialization mechanism is used to generate required number of individuals. Generally, random procedure is followed (refer Figure 4.4); which has been used for the OS part. However the proposed procedures of LS and GS [157] are also implemented in this algorithm for the MS part.

The LS procedure searches for the available options of the sequences and schedules the process which has minimum processing time. For example, $O_{22}$ can be scheduled on $M_2$ with a

minimum time of 66. The GS procedure takes into consideration the machine busy time ($T_k$) in order to further minimize the processing time. In this regard, $T_k$ is set equal to zero at first as there are no operations scheduled and machines are available / idle at the start of the problem. Once an operation is assigned and scheduled to a said machine, $T_k$ assumes the subsequent value of relevant $P_{ijk}$. The next operation is then scheduled to a machine where the added value of next $P_{ijk}$ becomes lowest when added to $T_k$. In this way, a global minimization is undertaken with reference to the machine busy time. These procedures are used in conjunction with each other so that an amalgamated population with properties of randomness, local and global best are produced for a selected machine selection. These procedure are presented pictorially in Figure 4.19 and Figure 4.20 respectively.

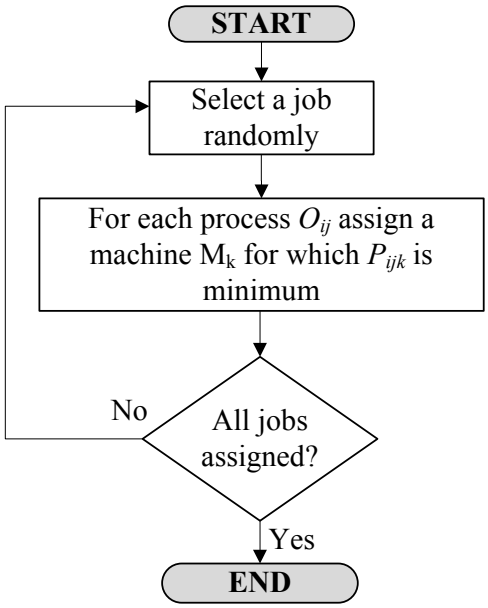


**Figure 4.19: Local search**

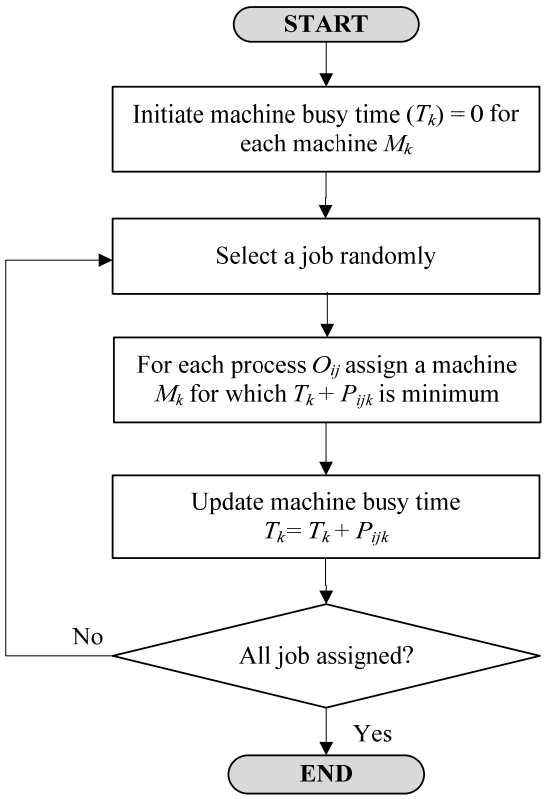


**Figure 4.20: Global search**

The chromosomes are encoded until the required population strength is achieved as presented in Figure 4.21. The initial percentage of LS, RS and GS are pre-set into the input excel sheet.

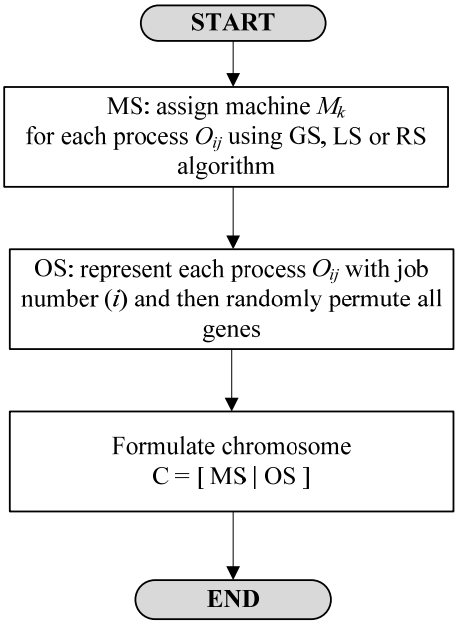


**Figure 4.21: Encoding of chromosome**

Since the algorithm needs to take decision whether the required number of chromosomes from a relevant technique have been generated or not, a decision tree is proposed for generation of population as presented in Figure 4.22. This procedure generates chromosomes until the number becomes equal to the required population size which was previously fed into the input sheet.

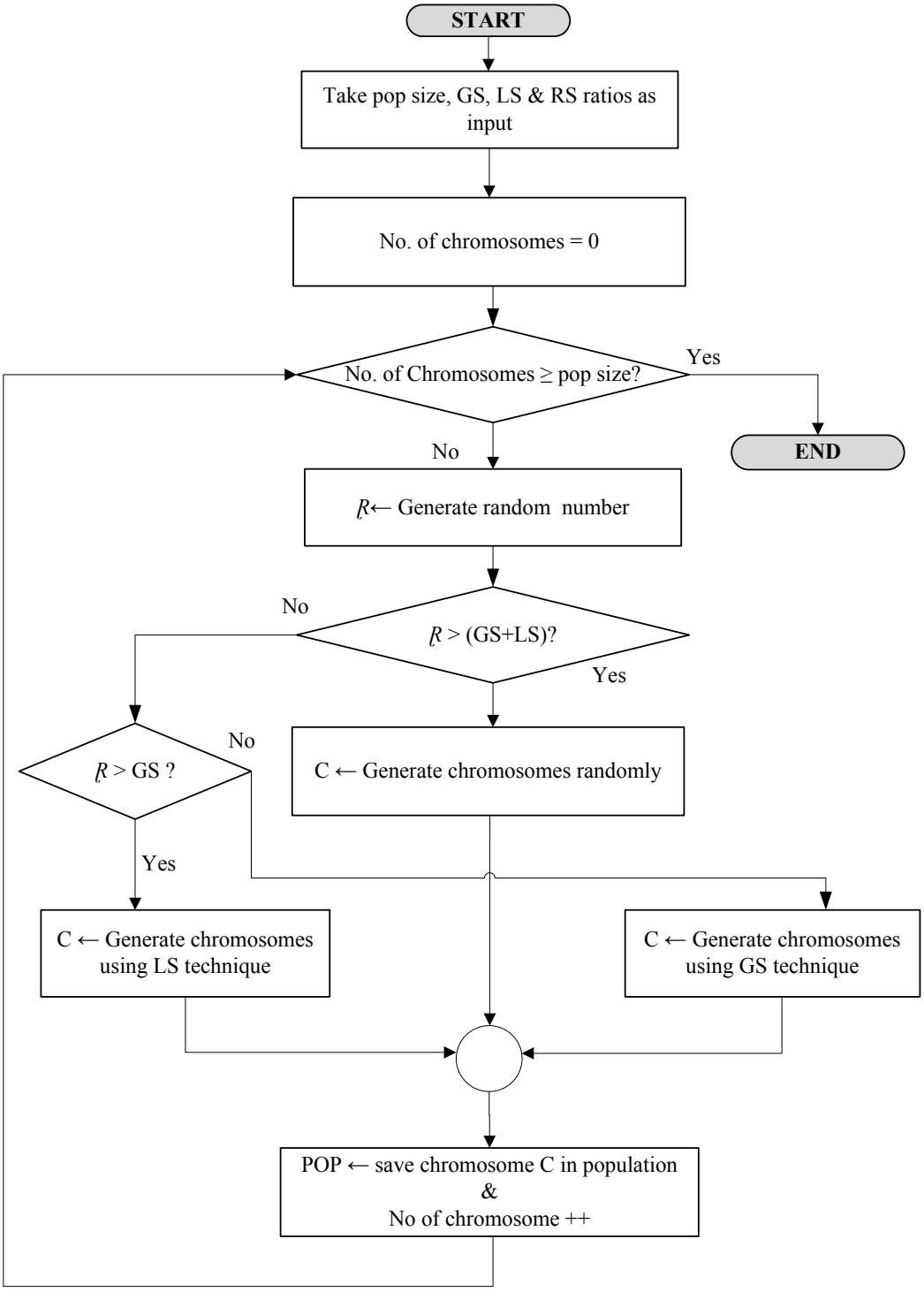


**Figure 4.22: Decision tree for GS, LS and RS**

Once the population has been initialized, the termination counter is set equal to zero and the population is evaluated for fitness. The parent population in then set for recombination as summarized in Figure 4.23 and explained below.

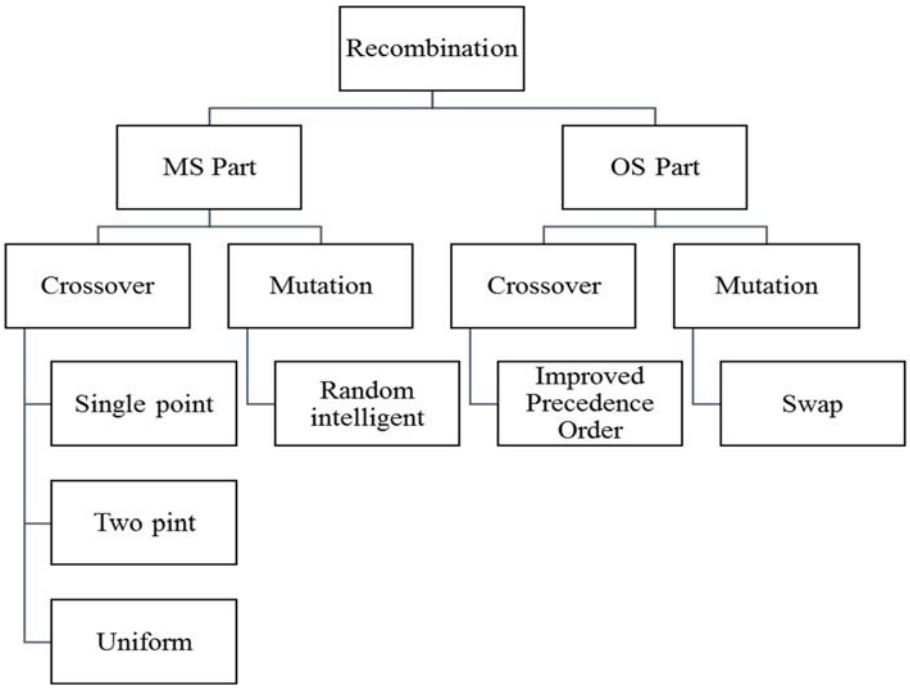


**Figure 4.23: Recombination operators for GA-IDT**

For MS part, SPX [212], TPX [286] and UX [157] are executed. Let us consider two parent chromosomes (MS part only) extracted from Fattahi 12 as shown in Figure 4.24. The SPX generates two offspring from [1 ~ ℝ] and [ℝ ~ $L$] where ℝ is a random number generated on the runtime in [1 ~ $L$]. An example of SPX is shown in

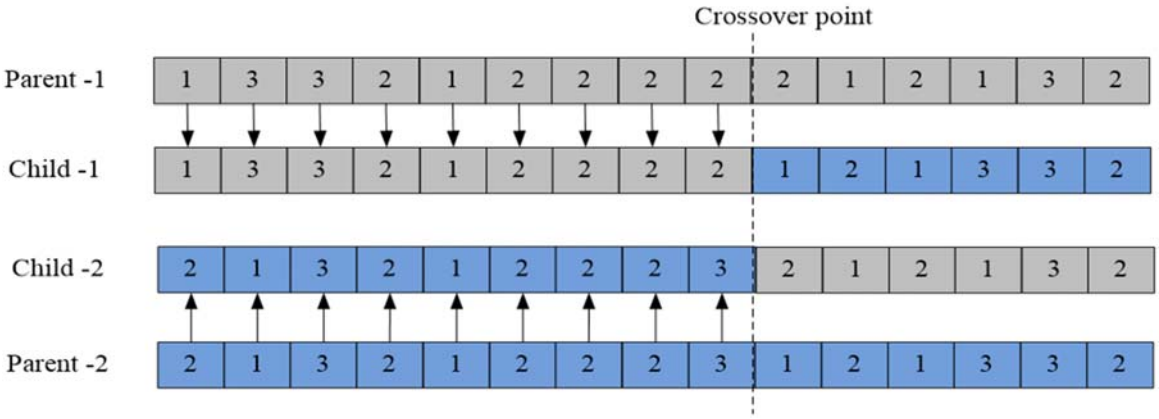


Figure 4.25 and its flowchart is presented in Figure 4.26.

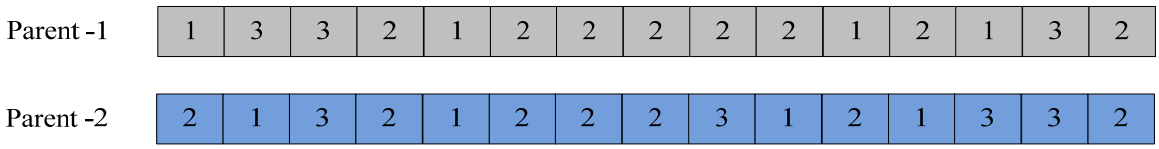


**Figure 4.24: Two parent chromosomes (MS part)**

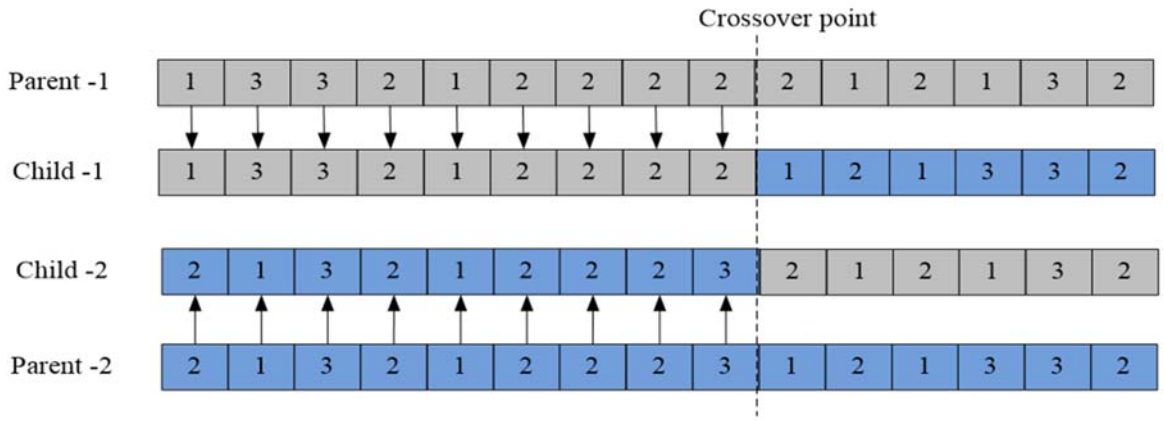


**Figure 4.25: An example of SPX**

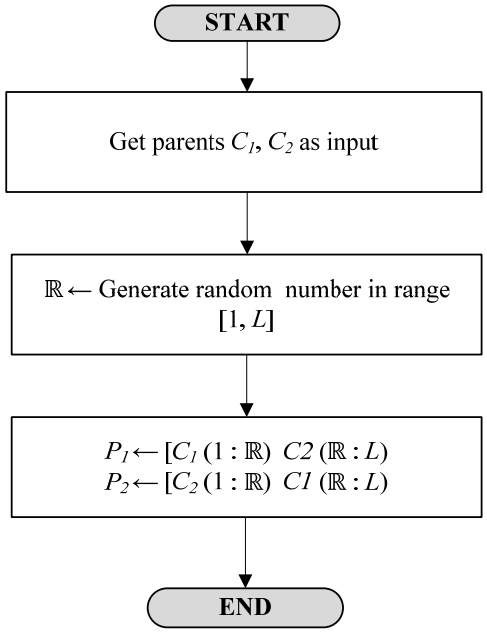


**Figure 4.26: Flowchart of SPX**

The TPX generates two offspring from $[1 \sim \mathbb{R}1]$ and $[\mathbb{R}1 \sim \mathbb{R}2]$ and $[\mathbb{R}2 \sim L]$ where $\mathbb{R}1$ and $\mathbb{R}2$ are random numbers generated on the runtime in the range of $[1, L]$. An example of TPX is presented in Figure 4.27 while its generalized flowchart has already been presented in Figure 4.6.

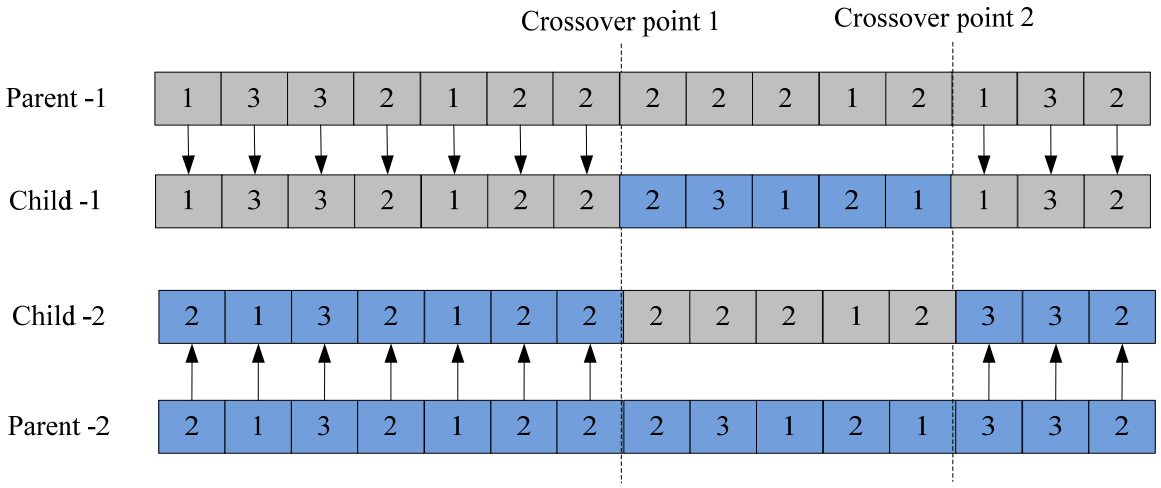


**Figure 4.27: An example of TPX**

The UX generates two offspring from such that offspring contain even and odd genes swapped from both parents turn by turn. An example of UX is presented in Figure 4.28, while its flowchart is shown in Figure 4.29.

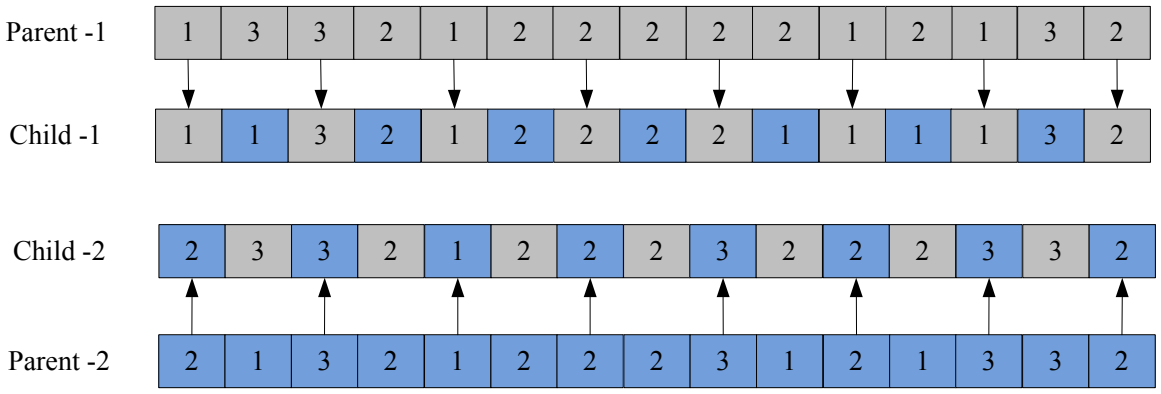


**Figure 4.28: An example of UX**

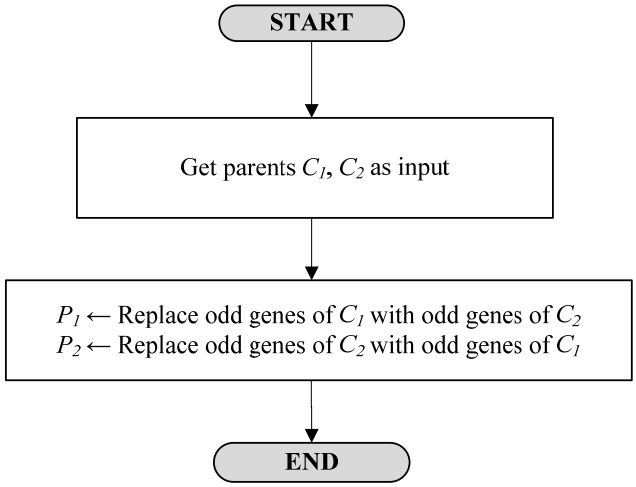


**Figure 4.29: Flowchart of UX**

The OS part is represented such that the sequence of genes represents the order of operations. Thus, it is imperative that this order must be preserved to avoid generation of infeasible / false OS part solutions which will eventually prove meaningless when integrated with the MS part. To ensure that information of order is preserved, iPOX [208] has been implemented. In this procedure, sets of jobs [$J_{s1}$, $J_{s2}$] are generated randomly through a random integer in [1, $n$] and offspring are produced in [1 ~ $x$] and [$x$ ~ $n$], where x is the random break point. An example in this regard is elaborated in Figure 4.30. Here, there are a total of 5 jobs, each having three operations. $J_{s1}$ and $J_{s2}$ are generated as [1, 2] and [3, 4, 5]. Now considering the Child-1, each gene in the chromosome pertaining to Job 1 and 2 will be transferred from Parent-1 and sequence will be copied as per original. Similarly, each gene in Child-1 pertaining to Job 3, 4 and 5 will be copied from Parent-2 and sequence will be kept similar as of the parent. Child-2 is generated vice-versa and two different offspring are generated through a set of job sequences. Obviously, changing the job sequence will change the offspring. A schematic flowchart of iPOX is presented in Figure 4.31.

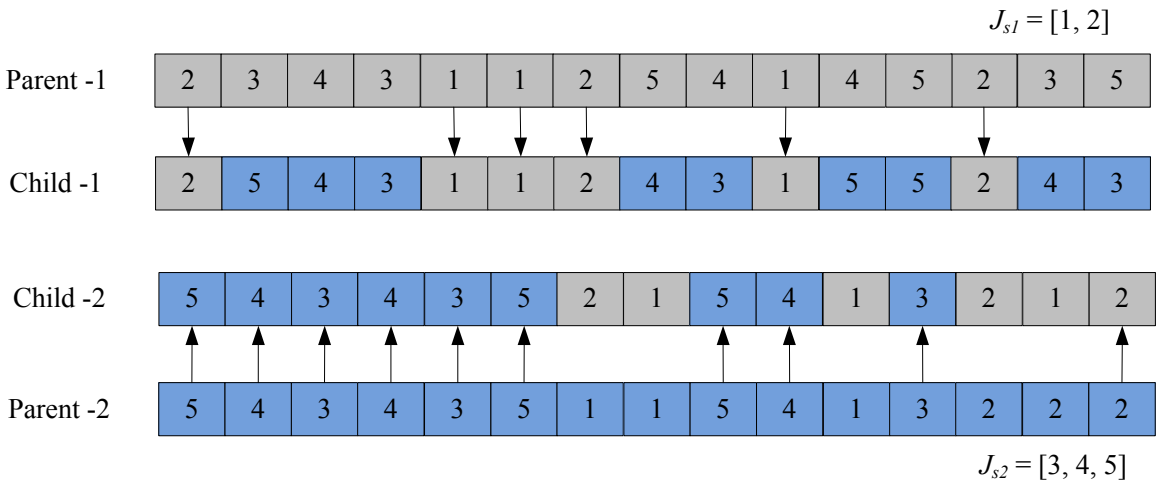


**Figure 4.30: An example of iPOX**

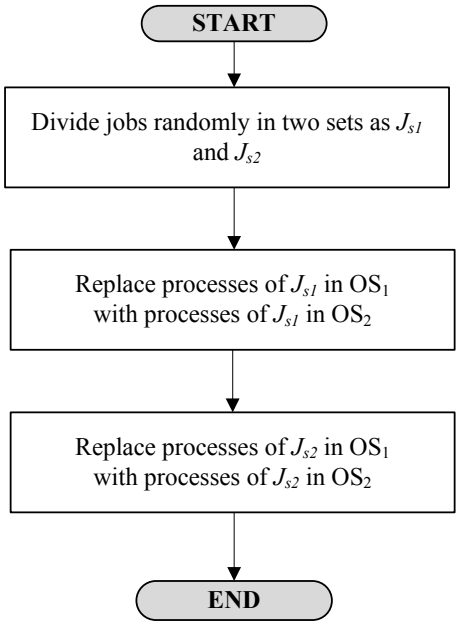


**Figure 4.31: Flowchart of iPOX**

Now coming to the mutation of the chromosome, Random Intelligent Mutation (RIM) has been used in GA-IDT for the MS part. A gene is selected and available options for machine assignment are evaluated and a new machine is assigned to mutate the gene. An example of this procedure is shown in Figure 4.32 and a flowchart is presented in Figure 4.33. Here, $O_{32}$ can be performed on $M_4$, $M_5$ and $M_7$. The operation was assigned to $M_7$ before mutation and after mutation, it has been assigned to $M_4$.

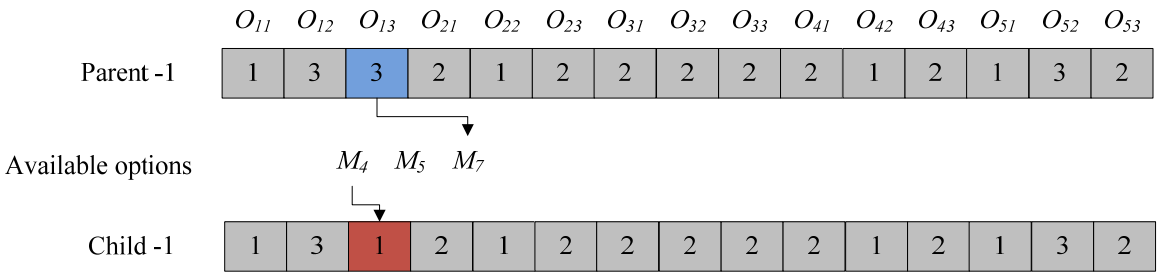


**Figure 4.32: An example of RIM**

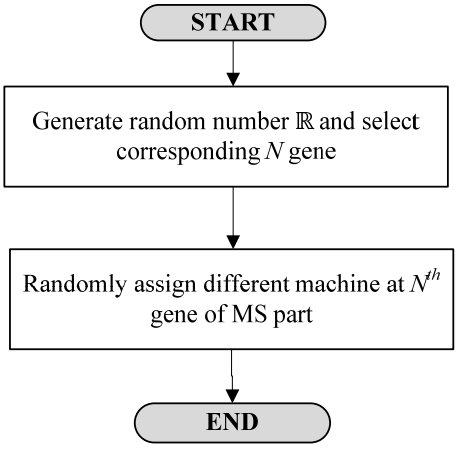


**Figure 4.33: Flowchart of RIM**

A swap type of mutation (SM) has been implemented on the OS part, whereby two genes are randomly selected, and their places are changed. An example of SM is shown in Figure 4.34 and its flowchart is presented in Figure 4.35. Here, gene number 4 and 13 are swapped together.

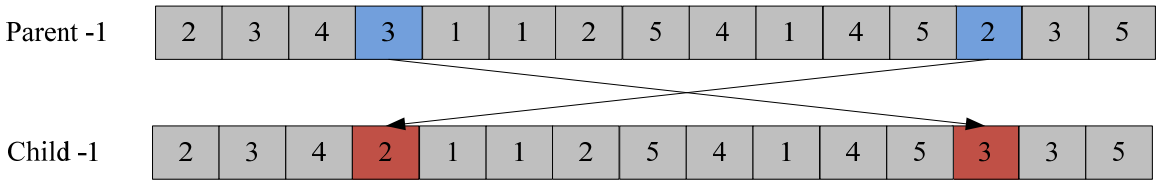


**Figure 4.34: An example of SM**

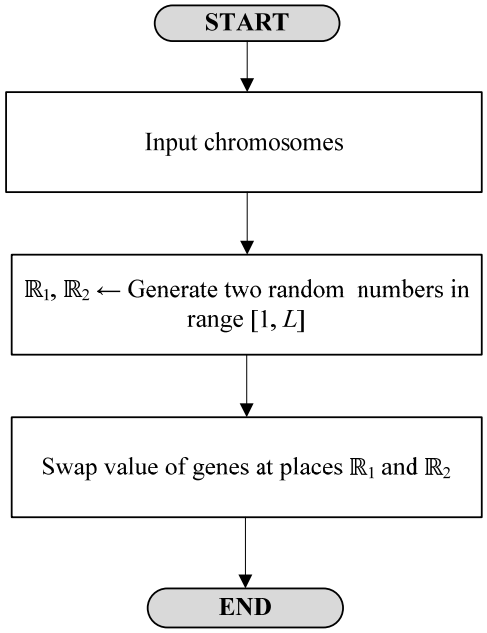


**Figure 4.35: Flowchart of SM**

As explained in GA-PR, the probabilities of recombination operators have also been taken as adaptive in GA-IDT [refer to Eq. (23) and Eq. (25)]. After the recombination process is culminated, hybrid selection mechanism is performed as already explained in Figure 4.9 and Figure 4.10. A set of new population is then achieved, and old population is replaced by it.

The algorithm then performs the check whether the solution fitness has improved. A similar fitness function has been implemented in this algorithm as presented in Figure 4.11. In case the fitness is improved, the termination counter is set to zero again and algorithm continues to run. Otherwise, the termination counter is increased by one and the algorithm satisfies one of the termination conditions. In case the solution is not improved, the algorithm also checks whether the termination limit is hit or not. The algorithm proceeds to repeat the GA procedure in case a false is returned, otherwise it proceeds to the third layer of the algorithm.

### 4.3.5 Layer 3: Re-initialization

When the algorithm reaches this layer, the population has sufficient elite solutions since termination counter is hit. This layer re-initializes the population to induce diversity in the elitist solutions while preserving the best solution. This procedure is meant to further evaluate the possible search space areas those were not studied earlier. To further augment the process, the GS is decreased by 10% and LS is increased by 10%.

Since, the population has been dispersed, the increment in the LS leads to intensify the possibility of local search in the areas of promising solutions and the decrease in GS is compensated by the re-initialization process. Thus, increased computational evaluation is now performed around the best solution, while exploring the overall search space owing to the dispersed population. The overall procedure is further assisted through the adaptability of mutation and crossover probabilities for ensuing the possibility of generating new and better solutions. The newly formulated population is then returned to the GA layer after incorporation of the elite solution. The algorithm continues to cycle through Layer 2 and Layer 3 in a similar way until the re-initialization limit is achieved.

### 4.3.6 Layer 4: Output

This is the final layer that takes the elite chromosome and produces a Gantt chart file and a convergence pattern file. It saves these files in the intended directory as defined in the input layer. Let's consider the example elite chromosome for Fattahi 12 problem as presented in Figure 4.36.

| 1 | 3 | 3 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 3 | 2 | 2 | 3 | 4 | 3 | 1 | 1 | 2 | 5 | 4 | 1 | 4 | 5 | 2 | 3 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**Figure 4.36: Chromosome to be decoded**

Figure 4.37 presents a pictorial representation of chromosome decoding. The MS part is translated into the *y-axis* of the Gantt chart. The operations are decoded to reveal the actual machine number on which the operation is to be performed, e.g. $O_{52}$ is to be performed on third available machine. Since the available machines on which the said operation can be performed are $M_3$, $M_4$ and $M_5$, the index of three points out $M_5$. Consequently, the operation is translated to relevant machine accordingly. All machine assignments are decoded in a similar way and *y-axis* of the Gantt chart is populated completely while solving the assignment problem.

Now coming to the sequencing problem is solved through decoding the OS part and translating relevant operation sequences to the corresponding machines. Again, considering the fifth job, first instance of 5 is obtained at gene number 8 which means that this operation is $O_{51}$. We see that this operation has been assigned to $M_1$. Remaining operations of fifth job, i.e. $O_{52}$ and $O_{53}$ are obtained at gene number 12 and 15. The scheduler refers to the assignment part and finds that these operations are assigned to $M_5$ and $M_6$, respectively. In a similar way, the complete OS part is decoded, and Gantt chart is populated by placing each operation with relevant machine while incorporating its process time. The operation which culminates at the end i.e. $O_{53}$ provides

the $C_{max}$ of the current chromosome which comes out to be 716 as shown in the final Gantt chart in Figure 4.38.

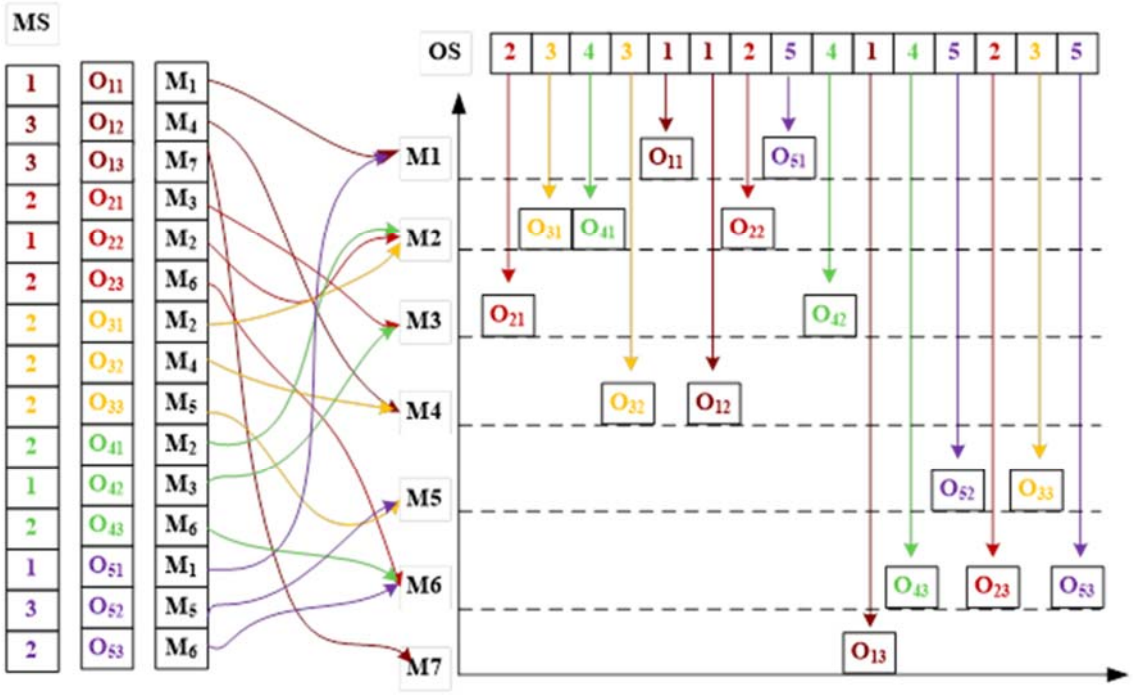


**Figure 4.37: A schematic of chromosome decoding**

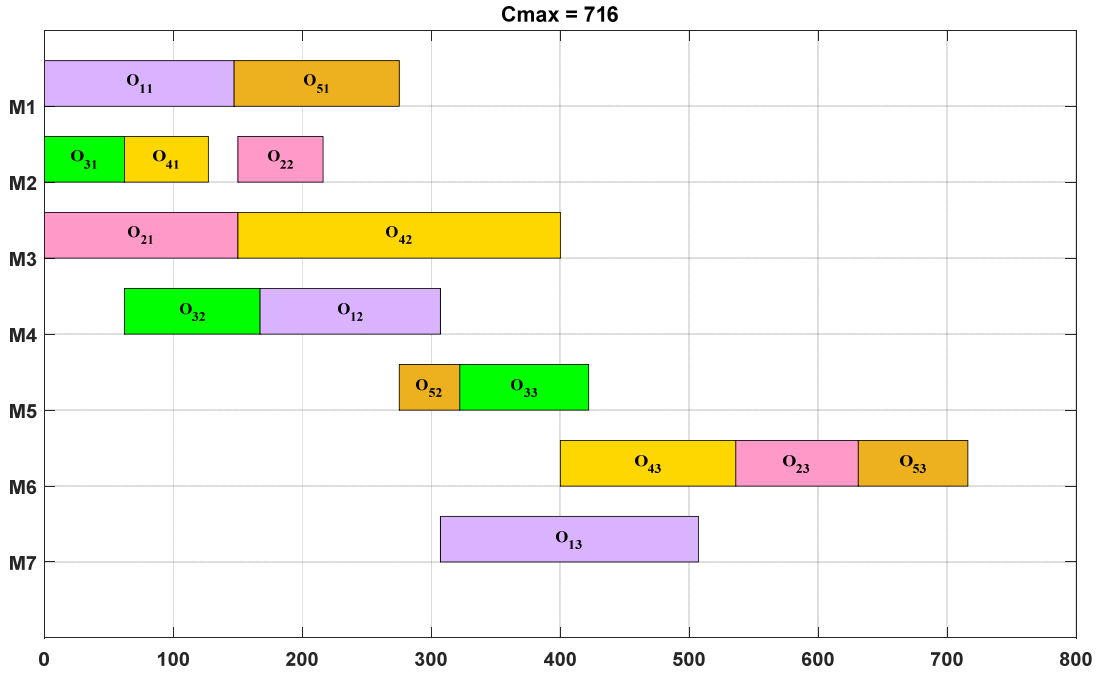


**Figure 4.38: Conversion of decoded chromosome into Gantt chart**

The termination criteria of this algorithm are as follows.

i. Maximum pre-set number of generations is acquired.

ii. No further improvement is achieved until 100 generations & re-initialization counter is hit.

## 4.4 Summary

This chapter has explained the proposed algorithms, namely GA-PR and GA-IDT in a detailed manner and the constituent parts of the algorithm are elaborated through figures and flowcharts. Numerous contributions to the existing knowledge body are also highlighted. Results of the proposed algorithms will be presented in the next chapter.

# 5 Chapter 5 – Experimental Investigation and Performance Evaluation

## 5.1 Introduction

This chapter provides results of the developed algorithms. Firstly, a generalized and numerical insight to the search envelope of the FJSSP is suggested to provide a visualization of the complexity and magnitude of the attempted benchmarks. Section-wise results of GA-PR and GA-IDT are then presented and compared with other relevant algorithms for evaluation of performance. The standard datasets of Fattahi and Kacem are selected for this purpose. The reason behind selecting Kacem instances is that their optimal solution has been achieved [282] and the problem size is sufficiently large so there is a room for testing the newly developed algorithm against known best. The known tendency of GA to get trapped in local minima can be easily tested with this dataset. Moreover, Fattahi has been selected keeping in view the fact that initial ten problems are meant for testing purposes i.e., these are small problems and algorithm correctness can be easily assessed by them. However, the later ten problems pose a challenge since their optima is not known.

A percentage deviation from the reported results of literature is calculated as follows.

$$\%\Delta = \frac{Value_{literature} - Value_{achieved}}{Value_{literature}} \tag{40}$$

The results are compared with equated algorithms built on similar patterns, while not excluding the possibility of cross-technique comparison. This is due to fact that hybrid algorithms belong to a different class and their efficiency assessment and quality of result can only be compared with algorithms of similar class. In this regard results produced by HTS-TS / HTS-SA [55], GA [252], AIA [287], MILP [288] and CP [282] have been compared with developed algorithms.

## 5.2 Evaluation of computational complexity and search space

FJSSP belongs to one of the most challenging and computationally complex NP-hard problems [289, 290] and it has been reported that the problem is virtually impossible to solve and may take up to millions of years to find the exact solution [30]. A very straight-forward way to quantitatively assess the enormous complexity of the FJSSP is to evaluate search. Although this can be simply said that the problem is NP-hard, there is an academic interest to find what is

actual amount of the solutions available in the search space. This helps to design the search techniques accordingly.

The search space size depends upon the chromosome length (and consequently the representation) and the flexibility ($U_{ij}$) of the problem. As per the adopted representation in the current research, the search space size is a product of MS part search space and OS part search space i.e. Eq. (31).

$$\aleph = SS(MS) \times SS(OS) \tag{41}$$

The space of search for MS part is evaluated in combinatorial way through product of all possible blends of the operations. Moreover, the search space of the OS part is evaluated again a combinatorial fashion by obtaining ratio of $L!$ and product of summations of $J_{io}!$. Following formulation [Eq. (32)] of the quantitative assessment of search space is proposed.

$$\aleph = \prod_{i=1}^{N} \prod_{j=1}^{J_{io}} U_{ij} \times \frac{L!}{\prod_{i=1}^{N} J_{io}!} \tag{42}$$

This formulation provides an insight to the actual combinations of operations, jobs and machines. The formulation has been applied to complete datasets of Kacem (04 x instances) and Fattahi (20 x instances) and results are presented in Table 5.1. The table also shows the number of jobs and machines for each problem. There is an exponential increase in the search space size with the increase in number of combinations, e.g. for 2 x 2 size, search space is of the order of E+01 (SFJS 01) and for 3 x 3 size it shoots up to the order of E+05 (SFJS 09).

**Table 5.1: Quantitative assessment of search space**

| Instance | *N* | *M* | *L* | א | Instance | *N* | *M* | *L* | א |
|---|---|---|---|---|---|---|---|---|---|
| SFJS1 | 2 | 2 | 4 | 9.60E+01 | MFJS3 | 6 | 7 | 18 | 4.67E+18 |
| SFJS2 | 2 | 2 | 4 | 2.40E+01 | MFJS4 | 7 | 7 | 21 | 1.12E+23 |
| SFJS3 | 3 | 2 | 6 | 1.44E+03 | MFJS5 | 7 | 7 | 21 | 7.45E+22 |
| SFJS4 | 3 | 2 | 6 | 1.44E+03 | MFJS6 | 8 | 7 | 24 | 1.81E+27 |
| SFJS5 | 3 | 2 | 6 | 5.76E+03 | MFJS7 | 8 | 7 | 32 | 3.00E+36 |
| SFJS6 | 3 | 3 | 9 | 1.08E+05 | MFJS8 | 9 | 8 | 36 | 2.82E+42 |
| SFJS7 | 3 | 5 | 9 | 8.60E+05 | MFJS9 | 11 | 8 | 44 | 1.35E+55 |
| SFJS8 | 3 | 4 | 9 | 8.60E+05 | MFJS10 | 12 | 8 | 48 | 6.28E+61 |
| SFJS9 | 3 | 3 | 9 | 8.60E+05 | Kacem 1 | 4 | 5 | 12 | 6.77E+13 |
| SFJS10 | 4 | 5 | 12 | 9.46E+07 | Kacem 2 | 10 | 7 | 29 | 1.41E+48 |
| MFJS1 | 5 | 6 | 15 | 1.39E+13 | Kacem 3 | 10 | 10 | 30 | 4.39E+54 |
| MFJS2 | 5 | 7 | 15 | 2.12E+14 | Kacem 4 | 15 | 10 | 56 | 2.03E+112 |

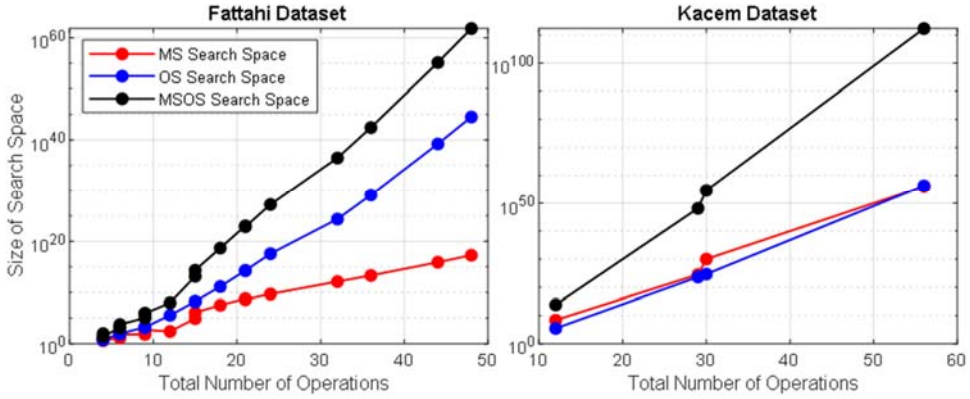


**Figure 5.1: Graphical presentation of search space size**

## 5.3 The results of GA-PR

The algorithms are implemented in MATLAB ® version 2018 and run on the Core-i7 Pentium (RAM of 4 GB). Different types of analysis are conducted to evaluate the overall performance of the algorithm which are explained below.

### 5.3.1 Contribution of priority rules

As explained in section 4.2.2, the priority rules integrated with GA routine solve the scheduling part of the problem turn-by-turn and the rule that provides best solution to the specific instance is selected to generate the schedule. This is because priority rules do not guarantee the

optimality of the solution alone because they are heuristic methods, and the assignment part is being solved by GA. Nevertheless, certain priority rules outsmart other schemes in certain scenario. Therefore, it is imperative that different types of rules may be used such that they may perform in all scenarios e.g., SPT cannot perform best in an environment where all jobs have equal times. Obviously, a limited number of rules can be integrated. Therefore, the rules selected in this study have been identified on the assumption that at least one performs in each scenario.

A procedure for evaluating the performance of rules has been suggested in Figure 5.2. The percentage contribution of each rule is finally calculated as shown in Eq. (33).

$$PC = \frac{E}{1000} \times 100 \tag{43}$$

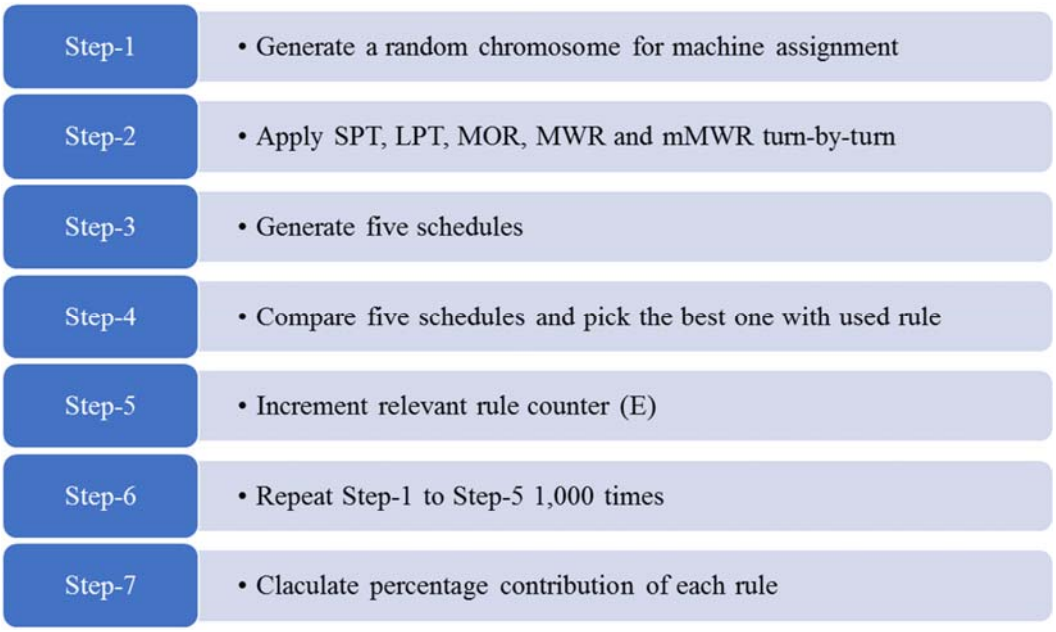


**Figure 5.2: Procedure for evaluating percentage contribution of selected rules**

The procedure presented in Figure 5.2 is applied to complete dataset of Fattahi. Step-6 includes 1,000 iterations of previous steps. Since, the assignment part is being solved by GA, the solutions are randomly oriented. Therefore, 1,000 iterations of priority rules are performed to evaluate the overall effect of rules being used. This procedure has been devised for evaluating the contribution of each rule towards generating the makespan. A visualization is for assessment of percentage contribution is presented in Figure 5.3. The colored areas pertaining to each rule as pointed out in the legend represent the percentage contribution of said rule as compared to the other rules. Following are some observations from the visualizations.

a. Fattahi 2 is completely solved by SPT alone and no other rule has proven to be effective in this case over 1,000 iterations.

b. Similarly, above 80% instances have been solved by SPT for Fattahi 1 and remaining instances were solved using LPT and no other rule proved to be effective.

c. Overall, SPT has shown most contribution.

d. The contribution of SPT falls drastically once the problem size increases and other rules start to play their role.

e. The work / operations remaining rules does not play any part in the initial 4 problems as all jobs are assigned and scheduled in the first instance.

f. MOR has the sleekest contribution, whereas the contribution of LPT is also on the lower side.

g. The MWR and mMWR increase their role in the larger problems because the number of machines become less as compared to the required number of operations to be conducted.

h. The contribution of mMWR increases as the problems become larger.

i. It is a general observation that the rule contribution depends upon the problem size and the specific machine assignment generated by GA. Hence, one rule can become more effective than the other in a specific instance depending upon the nature of prevalent instance, e.g. work remaining rule will dominate when the work remains, and the execution resources are scare etc.

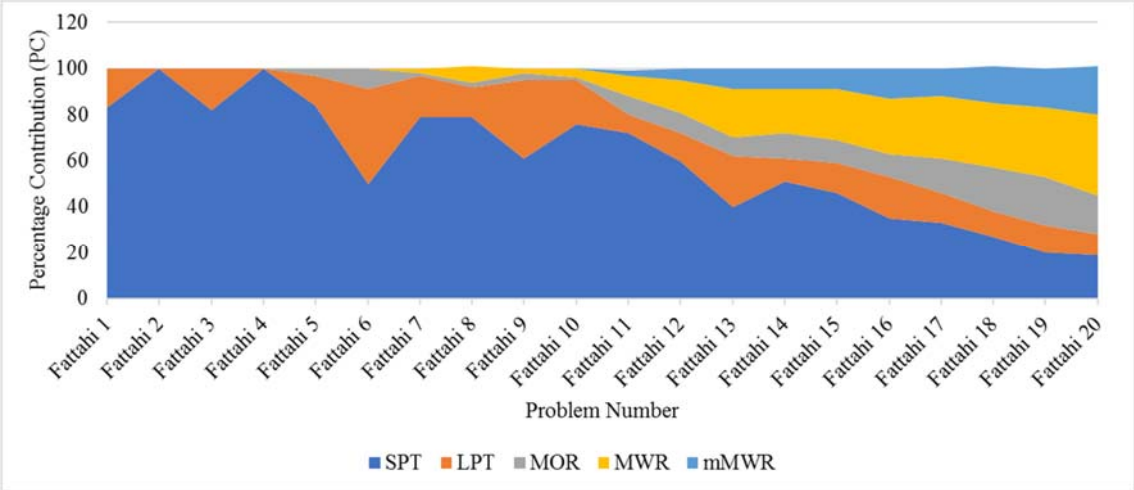


**Figure 5.3: Visualization of percentage contribution of selected rules**

For further illustration of the contribution of each rule in the generation of best $C_{max}$, the instance of Kacem-4 is solved with GA-PR. The upper half of Figure 5.4 shows the minimum,

maximum and average $C_{max}$ over 80 generations of GA-PR. The average and the minimum values continue to reduce over the generations, while the maximum value fluctuates owing to the adaptive parameters of recombination. The lower half of Figure 5.4 presents the contribution of each rule over each generation of algorithm and points out the best rule. The rule that contributes towards finding best solution has been indicated with a blue marker on the graph, e.g., in generation 73 – 75, SPT has contributed to finding the best solution. Similarly, during generations 75 – 80, mMWR has contributed to find the best solution. It is concluded that all rules play their part in providing best solutions over the generations. In addition, since the problem is a large problem, contribution of mMWR is significantly visible.

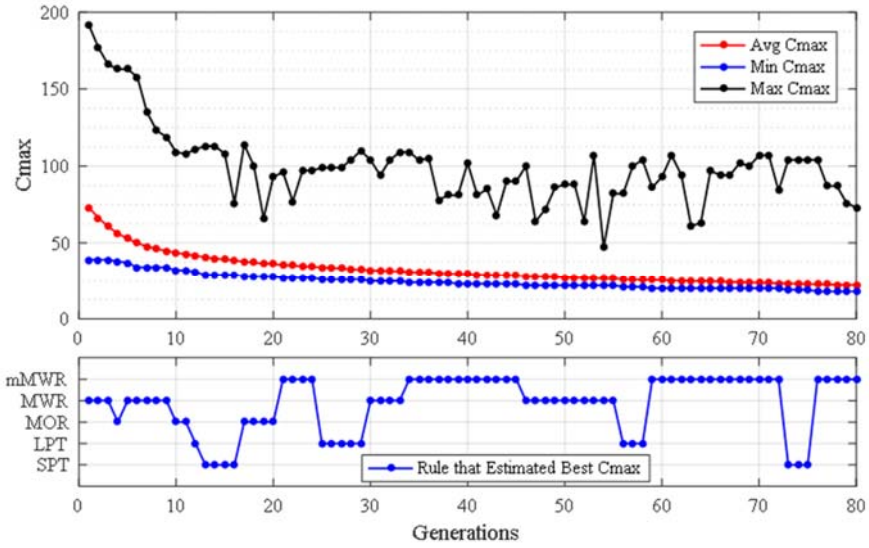


**Figure 5.4: Contribution of each rule in solving Kacem-4**

### 5.3.2 Behavior of adaptive recombination operator probabilities

The initial probability of crossover ($P_c$) is taken as 0.8 which is then subjective to change as per the adaptive implementation. Figure 5.5 presents the behavior of $P_c$ over the number of generations of a sample instance of FJSSP i.e., Kacem-4. The value of $P_c$ tends to increase as shown on the right vertical axis of the figure with the convergence of population which is indicated by the average $C_{max}$ indicator plotted at the left vertical axis.

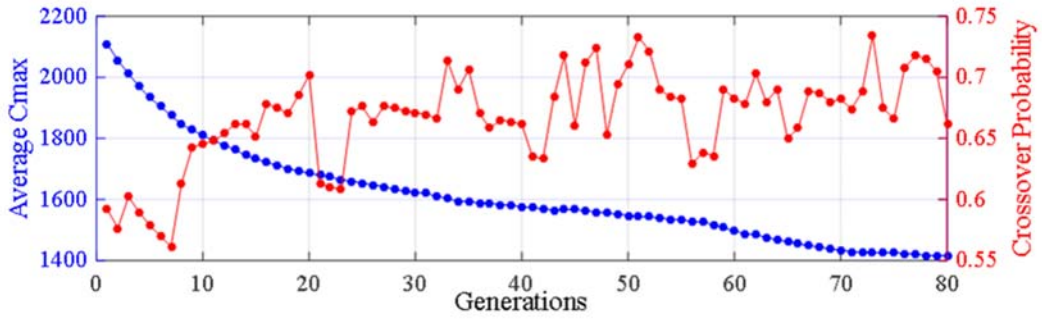


**Figure 5.5: Behavior of $P_c$**

The initial setting of $P_m$ is taken as 0.4 which is then subjected to adaptive increment. Figure 5.6 presents the behavior of Pm over similar generations as of Pc. The value of $P_m$ starts to increase (right vertical axis) as the best $C_{max}$ is lowered (left vertical axis).

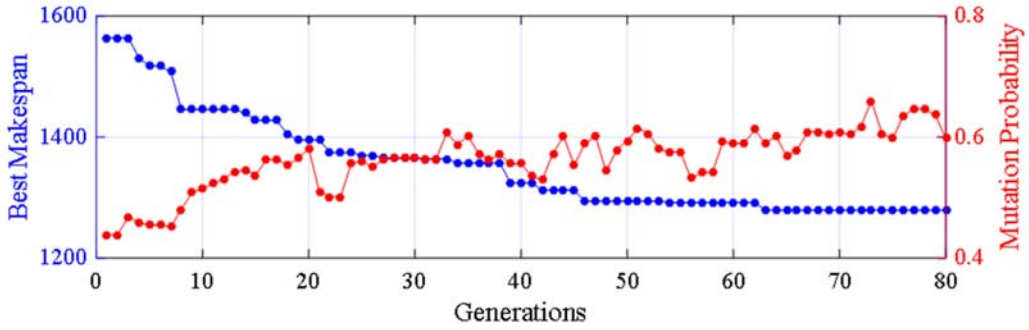


**Figure 5.6: Behavior of $P_m$**

This instrument of adaptive behavior is used to inculcate diversification in the population and to increase the possibility of finding new solutions and to explore extended search areas since the algorithm tends to converge around elite solution.

### 5.3.3 Behavior of hybrid selection

A dedicated experiment was conducted to evaluate the behavior of hybrid elite-RW selection mechanism. An initial population was generated using the random generation technique and its distribution of makespan against the number of individuals are shown in the left side of Figure 5.7.

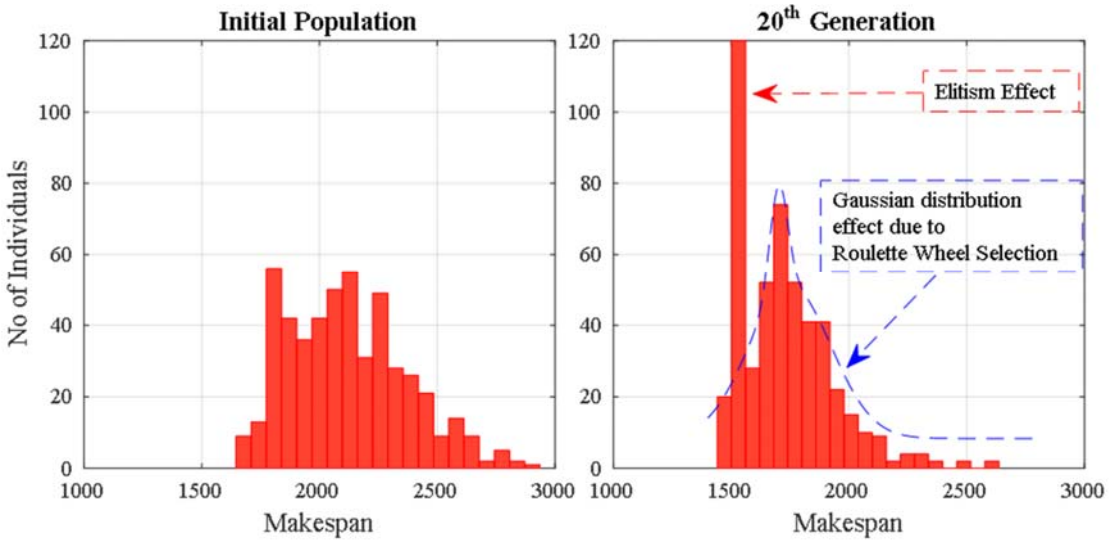


**Figure 5.7: Behavior of hybrid selection**

It is important to ascertain that the random population generation effectively generates a fairly distributed makespan individuals, both on the lower side and the higher side of the makespan bracket. When the 20th generation is achieved, the population has been evolved 20 times and convergence is now being started against the elite solution as indicated by the highest bar in the right side of Figure 5.7. Since the elitism effect is kept on a lower side to prevent hasty convergence, remaining population is undertaken by the RW criteria. It is thought-provoking that RW generates a Gaussian effect on the population distribution pattern.

### 5.3.4 Results of attempted instances

A total of 24 x instances were attempted using GA-PR, which contain 04 x instances of Kacem and 20 x instances of Fattahi. The parameters of GA-PR which were fed to the algorithm before start of execution are listed in Table 5.2. One approach could be to conduct a parametric study. However, the crossover and mutation probabilities are already adaptive, and they change according to the situation of the instance. Rest of the parameters have been taken from the literature review study [10].

**Table 5.2: Input parameters of GA-PR**

| Parameter | Description | Value |
|---|---|---|
| Population size | Total chromosomes in a population | 1500 |
| Generation size | Number of iterations in GA | 500 |
| Crossover probability | Likelihood for chromosome crossover | Adaptive |
| Mutation probability | Likelihood for chromosome mutation | Adaptive |
| Elitism ratio | Elite chromosome selection factor | 20 |
| Roulette wheel ratio | Factor for roulette wheel selection | 80 |

Table 5.3 presents the results of Kacem problems as compared with CP. It has been reported that optimal solutions of these instances have been found [282] and thus comparison is thus only made with one other algorithm in order to assess the efficacy of the algorithm. GA-PR achieved best solutions for the first three problems, however lags for the fourth instance.

**Table 5.3: Results of Kacem dataset for GA-PR**

| Problem | GA-PR | CP | |
|---|---|---|---|
| | | $C_{max}$ | %Δ |
| Kacem1 | 11 | 11 | 0 |
| Kacem2 | 11 | 11 | 0 |
| Kacem3 | 7 | 7 | 0 |
| Kacem4 | 14 | 12 | -16.7 |

The results of Fattahi dataset are presented in Table 5.4 along with the results of other algorithms and relevant %Δ. It is concluded that the algorithm produces comparable solutions. The mean %Δ for each algorithm is shown pictorially in Figure 5.8. The positive value points out that the overall results of the dataset surpass the other algorithm, while in case of CP, the negative value indicates that an overall lag is observed. As an improvement has been indicated in the results of MFJS-7, the Gantt chart is presented in Figure 5.9.

**Table 5.4: Results of Fattahi dataset for GA-PR**

| Problem | GA-PR | HTS/TS [55] | | HTS/SA [55] | | GA [252] | | AIA [287] | | MILP-1 [288] | | CP [282] | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $C_{max}$ | %Δ | $C_{max}$ | %Δ | $C_{max}$ | %Δ | $C_{max}$ | %Δ | $C_{max}$ | %Δ | $C_{max}$ | %Δ |
| SFJS1 | 66 | 66 | 0 | 66 | 0 | 66 | 0 | 66 | 0 | 66 | 0 | 66 | 0 |
| SFJS2 | 107 | 107 | 0 | 107 | 0 | 107 | 0 | 107 | 0 | 107 | 0 | 107 | 0 |
| SFJS3 | 221 | 221 | 0 | 221 | 0 | 221 | 0 | 221 | 0 | 221 | 0 | 221 | 0 |
| SFJS4 | 355 | 355 | 0 | 355 | 0 | 355 | 0 | 355 | 0 | 355 | 0 | 355 | 0 |
| SFJS5 | 119 | 119 | 0 | 119 | 0 | 119 | 0 | 119 | 0 | 119 | 0 | 119 | 0 |
| SFJS6 | 320 | 320 | 0 | 320 | 0 | 320 | 0 | 320 | 0 | 320 | 0 | 320 | 0 |
| SFJS7 | 397 | 397 | 0 | 397 | 0 | 397 | 0 | 397 | 0 | 397 | 0 | 397 | 0 |
| SFJS8 | 253 | 253 | 0 | 256 | 1.2 | 253 | 0 | 253 | 0 | 253 | 0 | 253 | 0 |
| SFJS9 | 210 | 210 | 0 | 210 | 0 | 210 | 0 | 210 | 0 | 210 | 0 | 210 | 0 |
| SFJS10 | 516 | 516 | 0 | 516 | 0 | 516 | 0 | 516 | 0 | 516 | 0 | 516 | 0 |
| MFJS1 | 468 | 469 | 0.2 | 469 | 0.2 | 468 | 0 | 468 | 0 | 468 | 0 | 468 | 0 |
| MFJS2 | 448 | 482 | 7.1 | 468 | 4.3 | 448 | 0 | 448 | 0 | 446 | -0.4 | 446 | -0.4 |
| MFJS3 | 468 | 533 | 12.2 | 538 | 13 | 466 | -0.4 | 468 | 0 | 466 | -0.4 | 466 | -0.4 |
| MFJS4 | 554 | 634 | 12.6 | 618 | 10.4 | 554 | 0 | 554 | 0 | 564 | 1.8 | 554 | 0 |
| MFJS5 | 514 | 625 | 17.8 | 625 | 17.8 | 514 | 0 | 527 | 2.5 | 514 | 0 | 514 | 0 |
| MFJS6 | 634 | 717 | 11.6 | 730 | 13.2 | 634 | 0 | 635 | 0.2 | 635 | 0.2 | 634 | 0 |
| MFJS7 | 881 | 964 | 8.6 | 947 | 7 | 881 | 0 | 879 | -0.2 | 935 | 5.8 | 931 | 5.4 |
| MFJS8 | 884 | 970 | 8.9 | 922 | 4.1 | 891 | 0.8 | 884 | 0 | 905 | 2.3 | 884 | 0 |
| MFJS9 | 1097 | 1105 | 0.7 | 1105 | 0.7 | 1094 | -0.3 | 1088 | -0.8 | 1192 | 8 | 1070 | -2.5 |
| MFJS10 | 1275 | 1404 | 9.2 | 1384 | 7.9 | 1286 | 0.9 | 1267 | -0.6 | 1276 | 0.1 | 1208 | -5.5 |

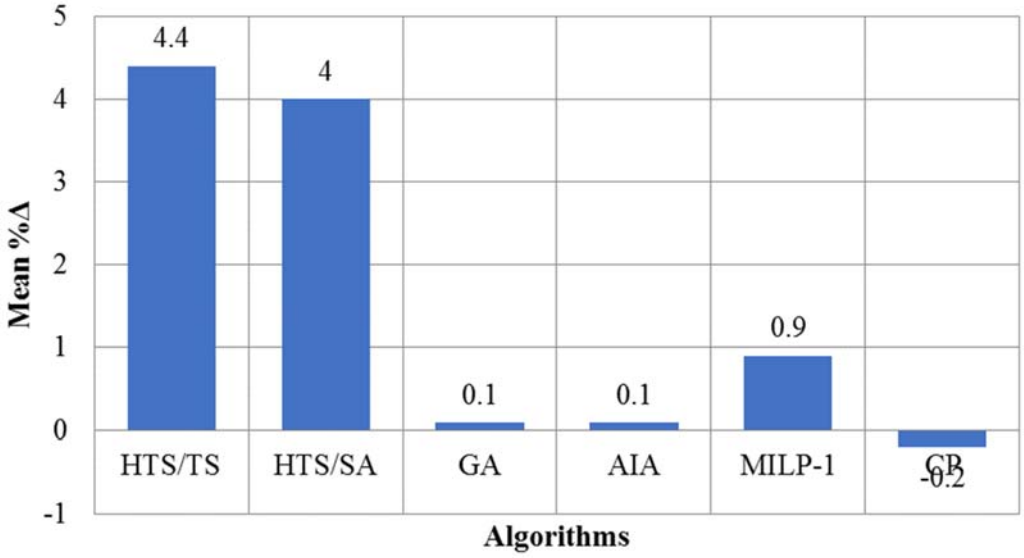


**Figure 5.8: Mean %Δ of GA-PR as compared with other Algorithms**

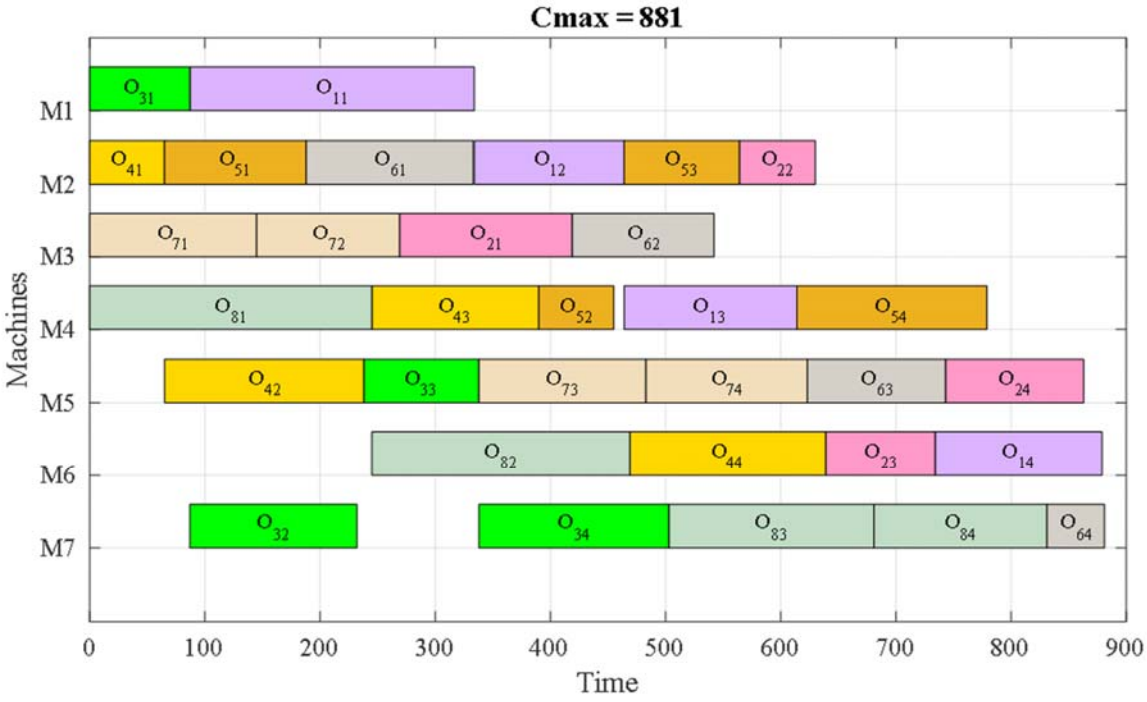


**Figure 5.9: MFJS-7 Gantt chart**

It is deduced from the data presented in Figure 5.8 that GA-PR surpasses other algorithms in general, but is left behind from CP. Since CP is the constraint programming logic and performs on mathematical formulation, it has produced better results as compared to GA-PR. The algorithms outsmart all other algorithms which are designed on the same principles of GA.

## 5.4 The results of GA-IDT

This algorithm is built to manage the aspects of diversification and intensification in an efficient manner. The algorithm is built in similar version of MATLAB ® and experiments are also conducted on similar machine as described in GA-PR. Again, judgement is made with comparable algorithms of similar nature only.

### 5.4.1 Consequence of re-initialization

GA-IDT undergoes re-initialization when termination counter is hit, and no further improvement is found until 100 generations are passed. This invokes changes in the population initialization schemes by altering the pre-set number of operators. To evaluate the effect of re-initialization on makespan quality, let us consider the convergence of MFJS8 as shown in Figure 5.10.

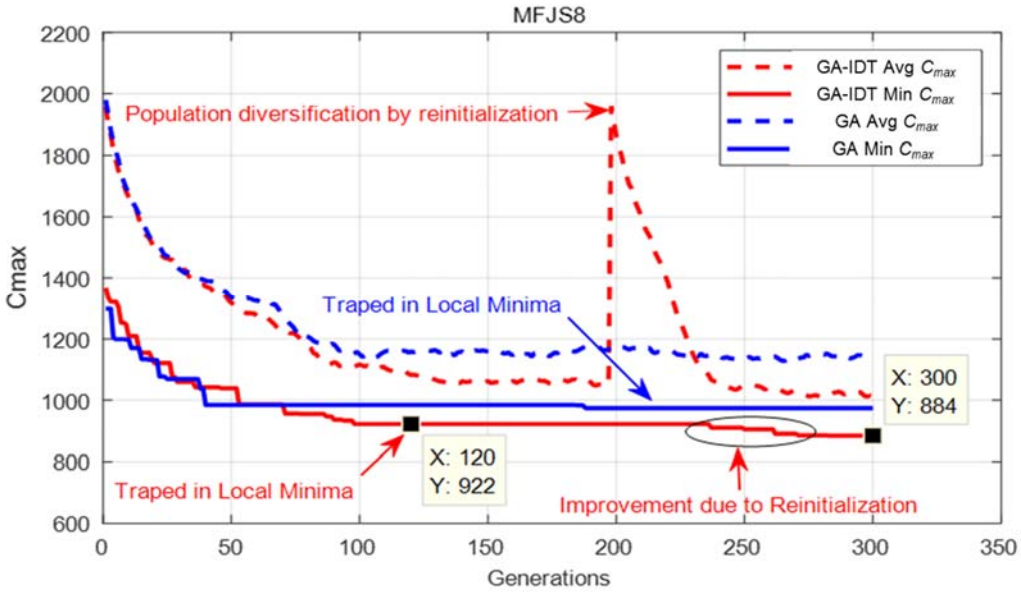


**Figure 5.10: Consequence of re-initialization on MFJS8 makespan**

When the algorithm starts, it undergoes multiple reductions in $C_{max}$ as lower values are continued to be found. The solution of simple GA does not improve after 180$^{th}$ generation and then stays there until another 120 generations have passed. and it seems that optimal point has been achieved as indicated by the straight blue line. However, as better solutions have already been found and reported in literature, it becomes evident that the point is local minima. Nevertheless, the algorithm stays there and terminates on local minima.

Now coming to the GA-IDT, the algorithm converges in a similar manner and attains stagnation after 120$^{th}$ generation. The results do not improve until 100 iterations and the algorithm re-initialize. The population diversity is dispersed after this and the algorithm is forced again to search the solution space while conserving the best solution. In addition to this, the initialization parameters are altered along with recombination probabilities. Just after the re-initialization is complete and algorithm again starts to converge, an improvement of 38 is found in many steps as indicated by the oval shaped marker in Figure 5.10. The algorithm then continues to execute until the termination criteria are met. It is germane to note here that the simple GA does not get out of the local minima once the population has converged against the elite individual; however, the GA-IDT gets out of the local solution and strives to find more better solutions due to the methodology of iterative diversification. In this regard, re-initialization, adaptive crossover, and global search tend to diversify the population, while adaptive mutation, local search and elitist preservation tend to intensify the population. This overall scheme has been termed as "Iterative Diversification Technique" (IDT).

Extended number of runs are conducted for other problems and similar behavior is found with regards to the IDT. Figure 5.11 presents the consequence of IDT on the convergence pattern. Following observations are found with analysis.

a. MFJS-2 gets stuck in local minima at very initial stage (< 50 generations) and remains there until the algorithm is re-initialized thrice and improved solution is found afterwards.
b. MFJS-4 gets caught in local minima after ~ 50 generations and re-initialization is invoked after completion of termination criteria. The algorithm finds better solutions after first attempt; however, no better solutions can be found after three further attempts. As a matter of fact, the solution found after first initialization is optimal.
c. Similar behavior as of MFJS-4 can be found in the convergence patterns of MFJS-5 and MFJS-6 (Figure 5.12). Both in MFJS 5 and MFJS 6, the algorithm gets trapped in local minima and evades from the trap after IDT is invoked.

It is therefore concluded that IDT is effective in escaping from local minima and can divulge the algorithm to better solution, even in the presence of pre-elite solutions. It is also concluded that numerous attempts of re-initialization may also prove to be useful; however, this comes at the cost of computational resources. It is therefore suggested that a total of 04 re-

initialization attempts be made as no improvements could be found after that during our experimentation.

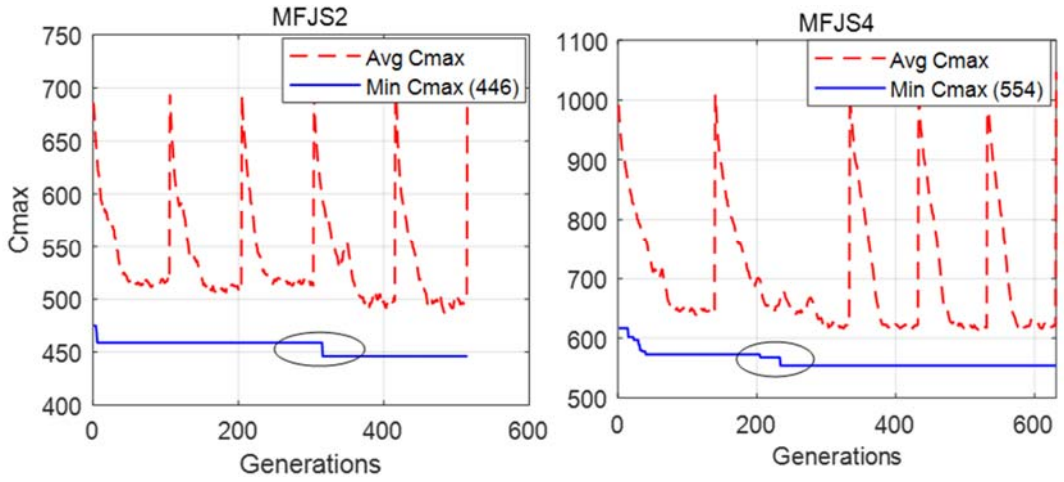


**Figure 5.11: Consequence of re-initialization on convergence of MFJS-2 and MFJS-4**

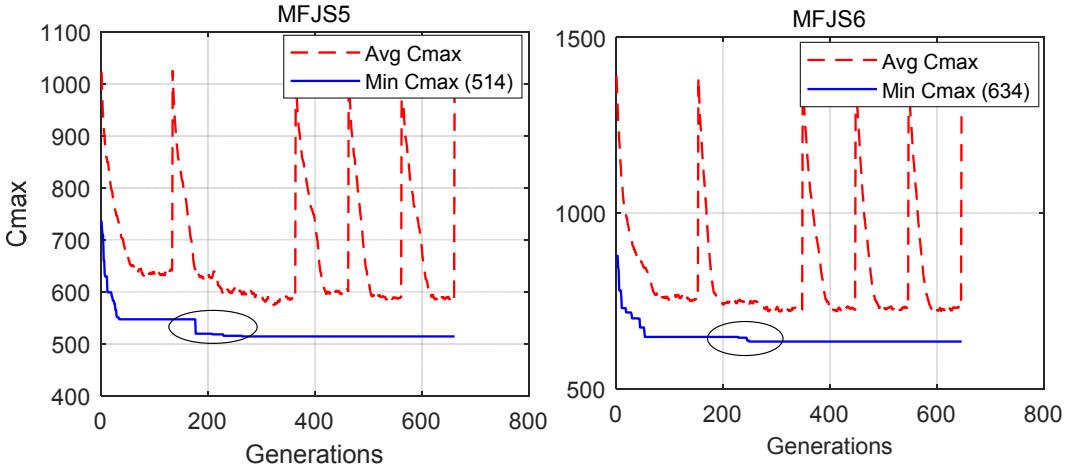


**Figure 5.12: Consequence of re-initialization on convergence of MFJS-5 and MFJS-6**

### 5.4.2 Results of attempted instances

Table 5.6 presents the results of Fatthi and Kacem datasets in comparison with other algorithms. It is notable that GA-IDT outperforms GA-PR by providing best known $C_{max}$ for Kacem-4 instance. Since the global optima for these instances have already been met, no further comparison with other approaches have been undertaken.

The algorithm is fed with predetermined parametric settings as pointed out in Table 5.5. For the purpose of this study, the recombination operator probabilities and selection operator ratios are kept adaptive. The termination counter in incremented after each generation with no improvement and continues to increment until 100 generations. After 100 iterations have been

conducted, the re-initialization counter is set-up and population is re-initialized as per the IDT and termination counter is set to zero again. This procedure continues until a total of 04 re-initialization attempts are completed.

**Table 5.5: Parametric settings of GA-IDT**

| Parameter | Description | Value |
|---|---|---|
| Population size | Total chromosomes in a population | 1500 |
| Generation size | Number of iterations in GA | 250 |
| Crossover probability | Likelihood for chromosome crossover | Adaptive |
| Mutation probability | Likelihood for chromosome mutation | Adaptive |
| Global selection ratio | Population initialization factor for global selection | Adaptive |
| Local selection ratio | Population initialization factor for local selection | Adaptive |
| Random selection ratio | Population initialization factor for random selection | Adaptive |
| Elitism ratio | Elite chromosome selection factor | 20 |
| Roulette wheel ratio | Factor for roulette wheel selection | 80 |
| Termination counter limit | Limit for GA before re-initialization | 100 |
| Re-initialization counter limit | Number of re-initialization attempts | 4 |

The results of Fattahi dataset are also listed in Table 5.6. Percentage difference is calculated from Eq. (31). It is notable that the algorithm at least remains competitive even in the larger problems solved with ILOG programming engine of CP. All entries show positive difference which means that GA-IDT surpasses the obtained solutions accordingly. The algorithm lags in MFJS-10.

**Table 5.6: Results of Fattahi and Kacem datasets for GA-IDT**

| Problem | GA-IDT | HTS/TS [55] | | HTS/SA [55] | | GA [252] | | AIA [287] | | MILP-1 [288] | | CP [282] | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $C_{max}$ | %Δ | $C_{max}$ | %Δ | $C_{max}$ | %Δ | $C_{max}$ | %Δ | $C_{max}$ | %Δ | $C_{max}$ | %Δ |
| SFJS1 | 66 | 66 | 0 | 66 | 0 | 66 | 0 | 66 | 0 | 66 | 0.0 | 66 | 0 |
| SFJS2 | 107 | 107 | 0 | 107 | 0 | 107 | 0 | 107 | 0 | 107 | 0.0 | 107 | 0 |
| SFJS3 | 221 | 221 | 0 | 221 | 0 | 221 | 0 | 221 | 0 | 221 | 0.0 | 221 | 0 |
| SFJS4 | 355 | 355 | 0 | 355 | 0 | 355 | 0 | 355 | 0 | 355 | 0.0 | 355 | 0 |
| SFJS5 | 119 | 119 | 0 | 119 | 0 | 119 | 0 | 119 | 0 | 119 | 0.0 | 119 | 0 |
| SFJS6 | 320 | 320 | 0 | 320 | 0 | 320 | 0 | 320 | 0 | 320 | 0.0 | 320 | 0 |
| SFJS7 | 397 | 397 | 0 | 397 | 0 | 397 | 0 | 397 | 0 | 397 | 0.0 | 397 | 0 |
| SFJS8 | 253 | 253 | 0 | 256 | 1.2 | 253 | 0 | 253 | 0 | 253 | 0.0 | 253 | 0 |
| SFJS9 | 210 | 210 | 0 | 210 | 0 | 210 | 0 | 210 | 0 | 210 | 0.0 | 210 | 0 |
| SFJS10 | 516 | 516 | 0 | 516 | 0 | 516 | 0 | 516 | 0 | 516 | 0.0 | 516 | 0 |
| MFJS1 | 468 | 469 | 0.2 | 469 | 0.2 | 468 | 0 | 468 | 0 | 468 | 0.0 | 468 | 0 |
| MFJS2 | 446 | 482 | 7.5 | 468 | 4.7 | 448 | 0.4 | 448 | 0.4 | 446 | 0.0 | 446 | 0 |
| MFJS3 | 466 | 533 | 12.6 | 538 | 13.4 | 466 | 0 | 468 | 0.4 | 466 | 0.0 | 466 | 0 |
| MFJS4 | 554 | 634 | 12.6 | 618 | 10.4 | 554 | 0 | 554 | 0 | 564 | 1.8 | 554 | 0 |
| MFJS5 | 514 | 625 | 17.8 | 625 | 17.8 | 514 | 0 | 527 | 2.5 | 514 | 0.0 | 514 | 0 |
| MFJS6 | 634 | 717 | 11.6 | 730 | 13.2 | 634 | 0 | 635 | 0.2 | 635 | 0.2 | 634 | 0 |
| MFJS7 | 879 | 964 | 8.8 | 947 | 7.2 | 881 | 0.2 | 879 | 0 | 935 | 6.0 | 931 | 5.6 |
| MFJS8 | 884 | 970 | 8.9 | 922 | 4.1 | 891 | 0.8 | 884 | 0 | 905 | 2.3 | 884 | 0 |
| MFJS9 | 1091 | 1105 | 1.3 | 1105 | 1.3 | 1094 | 0.3 | 1088 | -0.3 | 1192 | 8.5 | 1070 | -2 |
| MFJS10 | 1238 | 1404 | 11.8 | 1384 | 10.5 | 1286 | 3.7 | 1267 | 2.3 | 1276 | 3.0 | 1208 | -2.5 |
| Kacem1 | 11 | - | - | - | - | - | - | - | - | - | - | 11 | 0 |
| Kacem2 | 11 | - | - | - | - | - | - | - | - | - | - | 11 | 0 |
| Kacem3 | 7 | - | - | - | - | - | - | - | - | - | - | 7 | 0 |
| Kacem4 | 12 | - | - | - | - | - | - | - | - | - | - | 12 | 0 |
| | Mean %Δ | | 4.7 | - | 4.2 | - | 0.3 | - | 0.3 | - | 1.1 | - | 0 |

It is also pointed out that no lag is observed in the overall mean deviation despite the fact that GA-IDT lags from CP and the overall value comes out to be zero as improvement from CP has also been obtained. This aspect is pictorially shown in Figure 5.13. Since improvement has been obtained for MFJS-8, its Gantt chart is shown in Figure 5.14.

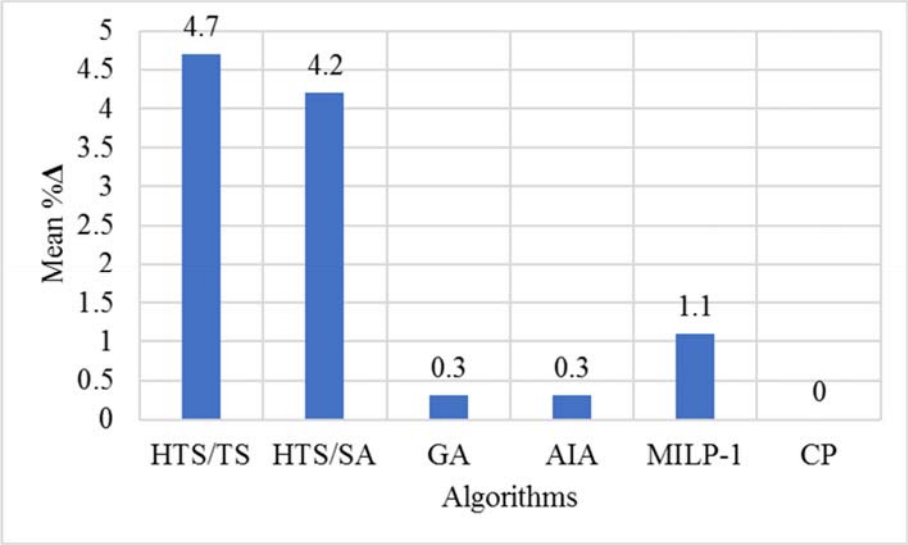


**Figure 5.13: Mean %Δ of GA-IDT as compared with other Algorithms**

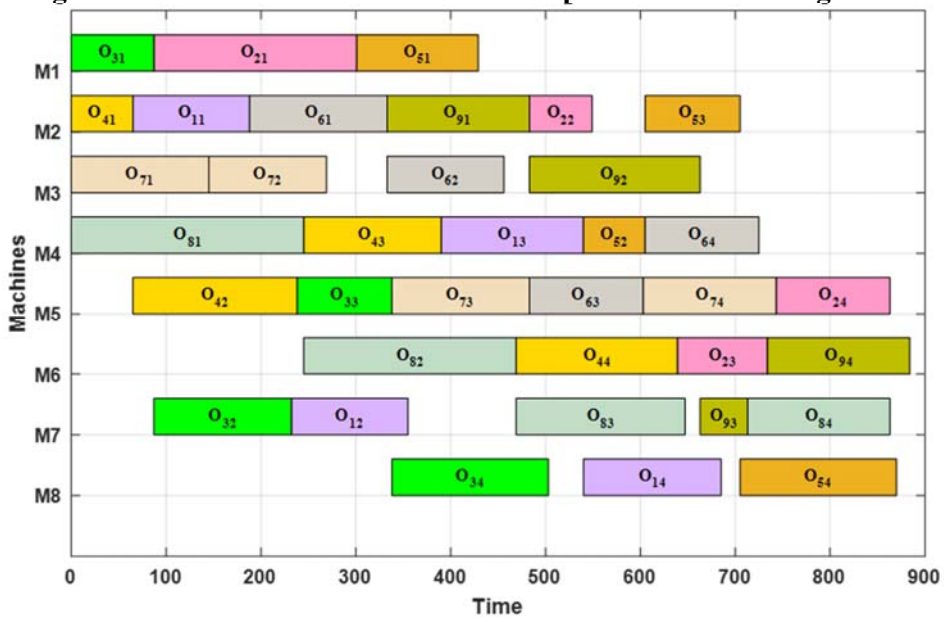


**Figure 5.14: MFJS-8 Gantt chart**

## 5.5 Comparison of GA-PR and GA-IDT

Although GA-PR and GA-IDT have been developed on different schemes and their architecture differs sufficiently, a comparison between the two algorithms is presented in this section. Figure 5.15 presents the comparison of GA-PR and GA-IDT in Fattahi dataset. It is

evident that as the problem size grows, the difference between the two algorithms become significant. The results of both algorithms are plotted as bar charts, while the difference between the two algorithms is shown on secondary axis through line graph. Hence GA-IDT surpasses GA-PR in obtaining $C_{max}$. Similarly, in Kacem dataset, the algorithms perform in the same manner as shown in Figure 5.16 i.e., as the problem size grows, the difference becomes evident.

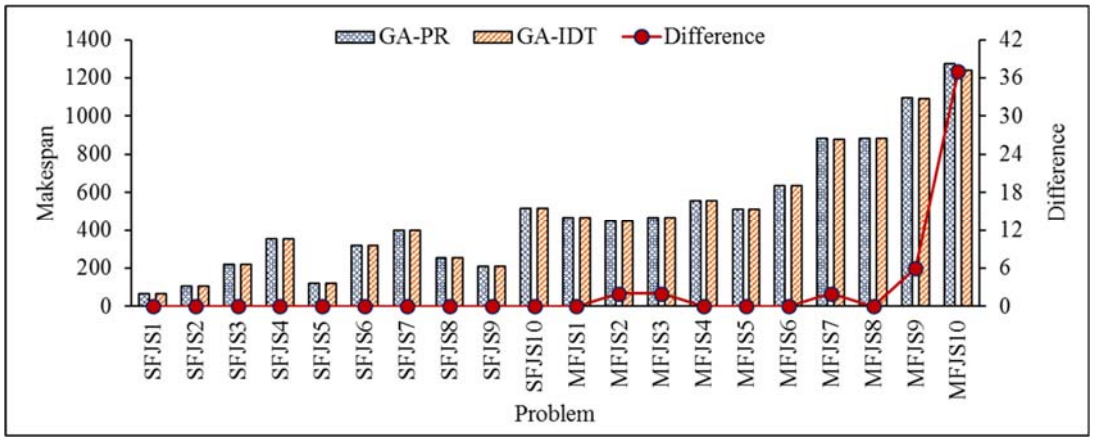


**Figure 5.15: Comparison between GA-and PR GA-IDT for Fattahi instances**

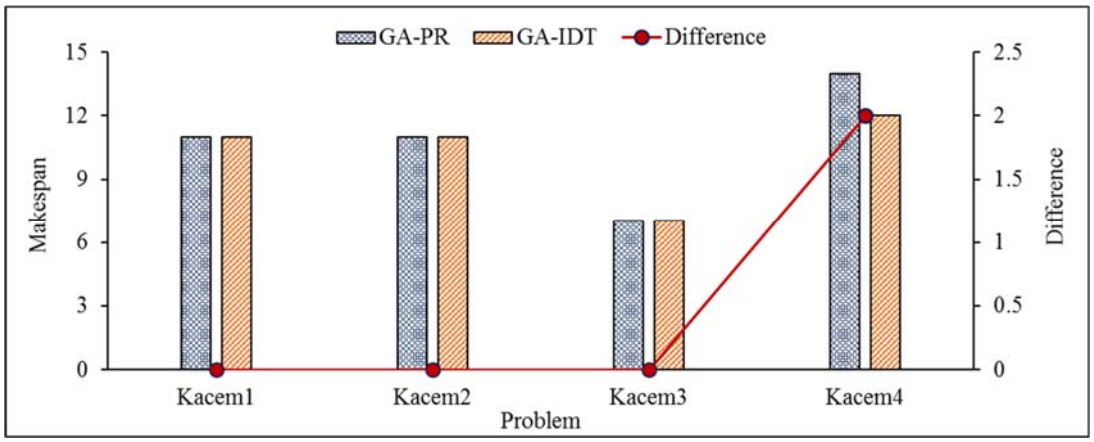


**Figure 5.16: Comparison between GA-and PR GA-IDT for Kacem instances**

The comparison of GA-PR and GA-IDT shows that GA-IDT performs better. Both algorithms have their strengths and weaknesses, however, since the optimization objective is $C_{max}$, GA-IDT provides better results. This is because half of the GA-PR is dependent upon selection and performance of rules. The algorithm cannot produce better results than theoretically possible from the heuristic rule. In addition, the search capability of the algorithm is dependent upon the GA part only. On the other hand, GA-IDT is completely based on GA and this provides advantage in further exploration of the search space and hence produces better results.

## 5.6 Summary

This chapter presented the results of the two proposed algorithms i.e. GA-PR and GA-IDT. The effectiveness and usefulness of the algorithms is then presented through conduct of different sorts of experiments to highlight the effect of proposed changes. The results show that the proposed algorithms perform in an efficient manner with other comparable methods and proposed improvements are useful. This fact is further augmented by the positive mean difference obtained from analysis of experimental results.

# 6 Chapter 6 – Conclusions and Recommendations

## 6.1 Contribution to the existing body of knowledge

Optimization of FJSSP by using artificial intelligence-based techniques is a widely researched area. Following are the contributions of the current work that stand-out from the already published literature.

i. ***Thorough literature review***: This work started with an in-depth literature assessment of the FJSSPs solved using different types of GA approaches. The work encompasses the study of more than 200 papers selected from a superset of more than 400 papers. The literature review outlines major aspects of the solution approaches followed in the literature and classifies these papers in pure GA, advanced GA, and hybrid GA based approaches. Extended statistical analysis of data revealed that GA remain the most popular and attempted technique for solving FJSSP even though initial attempt was made back some four decades ago. Major GA based approaches (whether standalone or hybrid) are also identified, and an analysis of different parameters is also conducted to outline the most useful operators. The study also helps to identify different benchmarks in the field and some advanced formats of the basic FJSSP. The study not only provides a start-up guide for schedulers but also provides future research directions along with an introduction to the advanced concepts in the field. This work has received above 40 citations till date.

ii. ***Insights to scheduling problems***: Manufacturing scheduling is a mathematically complicated area. This work has provided in depth coverage to the solution of scheduling problems through a sample benchmark instance. Different parameters have been solved and procedure has been outlined, in addition to the clear presentation of complex procedures through flowcharts.

iii. ***Quantitative assessment of search space***: The search apace of FJSSP is established as NP-hard and literature has proposed mathematical proofs in this regard. This work has proposed a formulation for evaluation of actual search space size through numerical estimation. Selected benchmarks have been evaluated accordingly and their search space is provided quantitatively to ascertain the genuine complexity. A comparative analysis in this regard the exponential increase in the problem complexity with the minimal increment in problem size.

iv. ***Development of standalone software for scheduling***: After the assessment of complexity, a software has been built in MATLAB ® for automated solution of the

problems. This software can undertake different settings of the problems through a pre-set MS Excel file and continue to evaluate stipulated instances until they all have been solved. The methodology also automatically saves the generated results, both numerically and graphically for later analysis.

v. ***Integration of rules with GA***: A novel method for integration of priority rules with GA has been proposed, whereby 05 x rules are solved turn-by-turn and optimal solution is taken out instead of just relying on a single rule. The rules are selected keeping in view different scenarios developed during the solution of scheduling problem. A modified rule is proposed to further enhance the efficacy of the algorithm.

vi. ***Development of mMWR***: While integrating the priority rules, a modification in the MWR has been proposed which detaches the rule from machine assignment. The rule has been tested against different dataset and found that it solves the larger problems efficiently as compared to other conventional rules. However, the rule may still not be used in a standalone manner and the implementation of complete GA-PR is recommended, since heuristic rules are developed for a targeted environment.

vii. ***Adaptive recombination operators***: The classical approach of fixating the recombination operators for generation of offspring tends to stick the algorithm and promotes convergence in a haste. This work has proposed adaptive operators for crossover and mutation, which acclimate themselves according to the current state of the population fitness to promote further exploration of the search space for a possible better solution.

viii. ***Development of an integrated GA approach for FJSSP***: In addition to the integration of rules with GA, this work also proposes an integrated GA for solving the FJSSP in a parallel fashion. The GA-IDT is equipped with similar adaptive recombination operators as suggested earlier in this work. Moreover, advanced operators have been implemented for more spread in the population initialization instead of random generation only. The algorithm is implemented in four layers which function one by one depending upon the algorithm stage to manage the huge amount of data in a comprehensive manner.

ix. ***Development of IDT***: This work proposes IDT that has been integrated into the GA. The technique is built such that the algorithm explores the search space sufficiently without trapping into local minima. This is done through increment of GS and RS operators, hybrid selection mechanism, adaptive crossover, and re-initialization. Since this may require the algorithm to run unnecessarily and may also restrain the algorithm to exploit the areas of promising solutions, the algorithm is also equipped with techniques which encourage exploitation of the search space. This done through adjustment of LS

operator, adaptive mutation, and elitist preservation. This methodology is tested against benchmarks and it is concluded that it surpasses the other comparable algorithms.

## 6.2 Future research directions

This research has opened further avenues for future endeavors which may be pursued as follows.

i. This research has evaluated the search spaces of basic FJSSPs. Efforts may be undertaken in future for numerical search space assessment of advanced forms of FJSSP, e.g. distributed FJSSP, just-in-time etc.

ii. Five rules have been integrated with GA for performance assessment. Other advanced rules may be integrated in the algorithm for further analysis. In addition, the objective function change may also be given a preference in future.

iii. The IDT may be further enhanced by integration of advanced concepts like Chaos maps for further enhanced of search performance.

iv. The algorithm may be applied to advanced cases of scheduling problems like resource constrained, machine breakdown, setup dependent times etc.

v. Lastly, other objective functions may be solved with this algorithm. In this regard fitness function for simultaneous optimization of multiple objectives may also be considered.

# References


1. Stearns, P.N. and W.L. Langer, *The encyclopedia of world history*, W.L. Langer, Editor. 2001, Houghton Mifflin Harcourt.
2. Beausmeister III, T.A.E.A., *Mark's Standard Handbook for Mechanical Engineers*. 11 ed, ed. A.M. Sadegh. 2007, New York: McGraw-Hill, New York. 2304.
3. Groover, M.P., *Automation, production systems, and computer-integrated manufacturing*. 2016: Pearson Education India.
4. Timings, R., *Basic manufacturing*. 2006, UK: Newnes, Elsevier.
5. Swamidass, P.M., *Encyclopedia of production and manufacturing management*. 2000, Springer Science & Business Media: United States of America.
6. WorldBank. *Manufacturing, value added (% of GDP)*. 2016; Available from: http://data.worldbank.org/indicator/NV.IND.MANF.ZS/countries?display=graph.
7. WorldBank. *World Bank Open Data*. 2020 02-02-2021]; Available from: https://data.worldbank.org/.
8. Rehman, F.U., A. Duffy, and X. Yan, *Design management and prediction*. 2008: Northwestern Polytechnical University Press.
9. Andreasen, M.M. and L. Hein, *Integrated product development*. 2000, Denmark: IFS.
10. Amjad, M.K., et al., *Recent Research Trends in Genetic Algorithm Based Flexible Job Shop Scheduling Problems.* Mathematical Problems in Engineering, 2018. **2018**: p. 32.
11. Koren, Y., *The global manufacturing revolution: product-process-business integration and reconfigurable systems*. Vol. 80. 2010: John Wiley & Sons.
12. Kusiak, A., *The generalized group technology concept.* International journal of production research, 1987. **25**(4): p. 561-569.
13. Ham, I., K. Hitomi, and T. Yoshida, *Group technology: applications to production management*. 2012: Springer Science & Business Media.
14. Tempelmeier, H. and H. Kuhn, *Flexible manufacturing systems: decision support for design and operation*. Vol. 12. 1993: John Wiley & Sons.
15. Sethi, A.K. and S.P. Sethi, *Flexibility in manufacturing: a survey.* International journal of flexible manufacturing systems, 1990. **2**(4): p. 289-328.
16. Slack, N., *The flexibility of manufacturing systems.* International Journal of Operations & Production Management, 1987. **7**(4): p. 35-45.
17. Hyer, N. and U. Wemmerlov, *Reorganizing the factory: Competing through cellular manufacturing*. 2001: CRC Press.

18. Bessant, J. and B. Haywood, *Flexibility in manufacturing systems.* Omega, 1986. **14**(6): p. 465-473.
19. Jain, A., et al., *A review on manufacturing flexibility.* International Journal of Production Research, 2013. **51**(19): p. 5946-5970.
20. Browne, J., et al., *Classification of flexible manufacturing systems.* The FMS magazine, 1984. **2**(2): p. 114-117.
21. Ploydanai, K. and A. Mungwattana, *Algorithm for solving job shop scheduling problem based on machine availability constraint.* International Journal on Computer Science and Engineering, 1919. **2**(5): p. 2010.
22. Mellor, P., *A review of job shop scheduling.* OR, 1966: p. 161-171.
23. Kiran, A.S. and M.L. Smith, *Simulation studies in job shop sheduling—I a survey.* Computers & Industrial Engineering, 1984. **8**(2): p. 87-93.
24. Adams, J., E. Balas, and D. Zawack, *The Shifting Bottleneck Procedure for Job Shop Scheduling.* Management Science, 1988. **34**(3): p. 391-401.
25. Krishna, K., K. Ganeshan, and D.J. Ram, *Distributed simulated annealing algorithms for job shop scheduling.* Systems, Man and Cybernetics, IEEE Transactions on, 1995. **25**(7): p. 1102-1109.
26. Brucker, P. and R. Schlie, *Job-shop scheduling with multi-purpose machines.* Computing, 1990. **45**(4): p. 369-375.
27. Chambers, J.B. and J.W. Barnes, *Tabu Search for the Flexible-Routing Job Shop Problem*. 1996, University of Texas at Austin: Austin, Texas.
28. Najid, N.M., S. Dauzere-Peres, and A. Zaidat, *A modified simulated annealing method for flexible job shop scheduling problem*, in *IEEE International Conference on Systems, Man and Cybernetics*, A.E. Kame, K. Mellouli, and P. Borne, Editors. 2002: Tunisia. p. 6 pp.
29. Pinedo, M.L., *Scheduling: theory, algorithms, and systems*. 2012, New York: Springer Science & Business Media.
30. Framinan, J.M., R. Leisten, and R.R. García, *Manufacturing scheduling systems*. London: Springer-Verlag. Vol. 10. 2014. 978-1.
31. Baker, K.R. and D. Trietsch, *Principles of sequencing and scheduling*. 2013: John Wiley & Sons.
32. T'kindt, V. and J.-C. Billaut, *Multicriteria scheduling: theory, models and algorithms*. 2006: Springer Science & Business Media.

33. Mokhtari, H. and M. Dadgar, *Scheduling optimization of a stochastic flexible job-shop system with time-varying machine failure rate.* Computers & Operations Research, 2015. **61**: p. 31-45.
34. Pinedo, M. and L. Schrage, *Stochastic shop scheduling: A survey*. Deterministic and Stochastic Scheduling, ed. M.A.H. Dempster. 1982: D. Reidel Publishing Company.
35. Kamburowski, J., *Stochastically minimizing the makespan in two-machine flow shops without blocking.* European Journal of Operational Research, 1999. **112**(2): p. 304-309.
36. Kundakcı, N. and O. Kulak, *Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem.* Computers & Industrial Engineering, 2016. **96**: p. 31-51.
37. Elgendy, A.E., M. Hussein, and A. Elhakeem, *Optimizing Dynamic Flexible Job Shop Scheduling Problem Based on Genetic Algorithm.* International Journal of Current Engineering and Technology, 2017.
38. Xiong, H., et al., *A simulation-based study of dispatching rules in a dynamic job shop scheduling problem with batch release and extended technical precedence constraints.* European Journal of Operational Research, 2017. **257**(1): p. 13-24.
39. Conway, R., W. Maxwell, and L. Miller, *Theory of scheduling. 1967.* Addison-Wesley, Reading, Mass.[: 5] M. Eisenberg, Two queues with changeover times, Oper Res.(2), 1971. **19**: p. 386-401.
40. Graham, R.L., et al., *Optimization and approximation in deterministic sequencing and scheduling: a survey*, in *Annals of discrete mathematics*. 1979, Elsevier. p. 287-326.
41. Lawler, E.L., et al., *Sequencing and scheduling: Algorithms and complexity.* Handbooks in operations research and management science, 1993. **4**: p. 445-522.
42. Cuiyu, W., L. Yang, and L. Xinyu, *Solving flexible job shop scheduling problem by a multi-swarm collaborative genetic algorithm.* Journal of Systems Engineering and Electronics, 2021. **32**(2): p. 261-271.
43. Brucker, P., Y.N. Sotskov, and F. Werner, *Complexity of shop-scheduling problems with fixed number of jobs: a survey.* Mathematical Methods of Operations Research, 2007. **65**(3): p. 461-481.
44. Gepp, M., et al., *A structured review of complexity in engineering-projects: State-of-research and solution concepts for the plant manufacturing business.* International Journal of Business and Management Studies, 2013. **5**(1): p. 318-327.
45. Van Dyke Parunak, H., *Characterizing the manufacturing scheduling problem.* Journal of Manufacturing Systems, 1991. **10**(3): p. 241-259.

46. Saidat, S., et al., *Modified job shop scheduling via Taguchi method and genetic algorithm.* Neural Computing and Applications, 2021: p. 1-18.
47. Yang, R. *An improved genetic algorithm for solving flexible job shop*. in *Journal of Physics: Conference Series*. 2021. IOP Publishing.
48. Brandimarte, P., *Routing and scheduling in a flexible job shop by tabu search.* Annals of Operations Research, 1993. **41**(3): p. 157-183.
49. Hurink, J., B. Jurisch, and M. Thole, *Tabu search for the job-shop scheduling problem with multi-purpose machines.* Aerospace Science and Technology Operations Research Spektrum, 1994. **15**(4): p. 205-215.
50. Fisher, H. and G.L. Thompson, *Probabilistic learning combinations of local job-shop scheduling rules.* Industrial scheduling, 1963. **3**(2): p. 225-251.
51. Dauzère-Pérès, S. and J. Paulli, *An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search.* Annals of Operations Research, 1997. **70**: p. 281-306.
52. Barnes, J.W.C., J. B, *Flexible Job Shop Scheduling by Tabu Search*, T.R.S. Graduate Program in Operations Research and Industrial Engineering, ORP96-09, Editor. 1996, University of Texas at Austin.
53. Lawrence, S., *Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (supplement)*. Graduate School of Industrial Administration, 1984.
54. Kacem, I., S. Hammadi, and P. Borne. *Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems*. in *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*. 2002.
55. Fattahi, P., M. Saidi-Mehrabad, and F. Jolai, *Mathematical modeling and heuristic approaches to flexible job shop scheduling problems.* Journal of Intelligent Manufacturing, 2007. **18**(3): p. 331-342.
56. Li, L.H., Jia-zhen, *Multi-Objective Flexible Job-Shop Scheduling Problem in Steel Tubes Production.* Systems Engineering - Theory & Practice, 2009. **29**(8): p. 117-126.
57. Mati, Y.L., Chams; Dauzère-Pérès, Stéphane, *Modelling and solving a practical flexible job-shop scheduling problem with blocking constraints.* International Journal of Production Research, 2011. **49**(8): p. 2169-2182.
58. Wong, S.V.K.Y., *A Genetic Algorithm Approach for Solving a Flexible Job Shop Scheduling Problem.* International Journal of Computer Science Issues, 2012. **9**(3): p. 6 pages.

59. Zhang, T.R., Oliver, *Scheduling in a Flexible Job Shop with Continuous Operations at the Last Stage*, in *15th ASIM 'Simulation in Production and Logistics' 2013*. 2013: Paderborn, Germany. p. 611-620.
60. Li, Y., G. Luo, and B. Wu. *Flexible Job Shop Scheduling Based on Genetically Modified Neighborhood Hybrid Algorithm*. in *2019 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*. 2019. IEEE.
61. Bhatti, M.A., *Practical Optimization Methods: With Mathematica® Applications*. 2012: Springer Science & Business Media.
62. Balci, S.A., *Solution Approaches For Flexible Job Shop Scheduling Problems*, in *Department of Industrial Engineering*. 2013, Middle East Technical University, Turkey: Middle East Technical University, Turkey.
63. Johnson, S.M., *Optimal two- and three-stage production schedules with setup times included.* Naval Research Logistics Quarterly, 1954. **1**(1): p. 61-68.
64. Moore, J.M., *An n job, one machine sequencing algorithm for minimizing the number of late jobs.* Management science, 1968. **15**(1): p. 102-109.
65. Lawler, E.L., *Optimal sequencing of a single machine subject to precedence constraints.* Management science, 1973. **19**(5): p. 544-546.
66. Chen, Y.J., M. Zhang, and M.M. Tseng, *An Integrated Process Planning and Production Scheduling Framework for Mass Customization.* Journal for Manufacturing Science and Production, 2004. **6**(1-2): p. 89–101.
67. Brucker, P., E.K. Burke, and S. Groenemeyer, *A mixed integer programming model for the cyclic job-shop problem with transportation.* Discrete Applied Mathematics, 2012. **160**(13): p. 1924-1935.
68. Mehrabad, M.S. and S. Zarghami, *Exact Mixed Integer Programming for Integrated Scheduling and Process Planning in Flexible Environment.* Journal of Optimization in Industrial Engineering, 2014. **7**(15): p. 47-53.
69. Shim, S.-O. and Y.-D. Kim, *A branch and bound algorithm for an identical parallel machine scheduling problem with a job splitting property.* Computers & Operations Research, 2008. **35**(3): p. 863-875.
70. Brucker, P., E.K. Burke, and S. Groenemeyer, *A branch and bound algorithm for the cyclic job-shop problem with transportation.* Computers & Operations Research, 2012. **39**(12): p. 3200-3214.
71. Rinnooy Kan, A., B. Lageweg, and J.K. Lenstra, *Minimizing total costs in one-machine scheduling.* Operations research, 1975. **23**(5): p. 908-927.

72. Taillard, E., *Benchmarks for basic scheduling problems.* European Journal of Operational Research, 1993. **64**(2): p. 278-285.
73. Liu, S.Q. and E. Kozan, *A hybrid shifting bottleneck procedure algorithm for the parallel-machine job-shop scheduling problem.* Journal of the Operational Research Society, 2012. **63**(2): p. 168-182.
74. Logendran, R., P. Ramakrjshna, and C. Sriskandarajah, *Tabu search-based heuristics for cellular manufacturing systems in the presence of alternative process plans.* International Journal of Production Research, 1994. **32**(2): p. 273-297.
75. Armentano, V.A. and D.S. Yamashita, *Tabu search for scheduling on identical parallel machines to minimize mean tardiness.* Journal of Intelligent Manufacturing, 2000. **11**(5): p. 453-460.
76. Mati, Y., N. Rezg, and X. Xiaolan, *An integrated greedy heuristic for a flexible job shop scheduling problem*, in *IEEE International Conference on Systems, Man, and Cybernetics*. 2001: Arizona, USA. p. 2534-2539.
77. Resende, M.G.C. and C.C. Ribeiro, *Greedy Randomized Adaptive Search Procedures: Advances, Hybridizations, and Applications*, in *Handbook of Metaheuristics*, M. Gendreau and J.-Y. Potvin, Editors. 2010, Springer US. p. 283-319.
78. Baker, J.E., *Adaptive Selection Methods for Genetic Algorithms*, in *Proceedings of the 1st International Conference on Genetic Algorithms*. 1985, L. Erlbaum Associates Inc. p. 101-111.
79. Biegel, J.E. and J.J. Davern, *Genetic algorithms and job shop scheduling.* Computers & Industrial Engineering, 1990. **19**(1): p. 81-91.
80. Kumar, R., M.K. Tiwari, and R. Shankar, *Scheduling of flexible manufacturing systems: an ant colony optimization approach.* Proceedings of the I MECH E Part B Journal of Engineering Manufacture, 2003. **217**(11): p. 1443-1453.
81. Dorigo, M. and C. Blum, *Ant colony optimization theory: A survey.* Theoretical computer science, 2005. **344**(2): p. 243-278.
82. Hart, E., P. Ross, and J. Nelson. *Producing robust schedules via an artificial immune system*. in *IEEE International Conference on Evolutionary Computation Proceedings and IEEE World Congress on Computational Intelligence*. 1998.
83. Hong, L., *A new artificial immune algorithm for flexible job-shop scheduling.* Advanced Materials Research, 2010. **121-122**: p. 266-270.
84. Davarzani, Z., M.-R. Akbarzadeh-T, and N. Khairdoost, *Multiobjective Artificial Immune Algorithm for Flexible Job Shop Scheduling Problem.* International Journal of Hybrid Information Technology, 2012. **5**(3): p. 75-88.

85. Sipahioglu, A. and A. Aladag, *An Artificial Immune System Approach for Flexible Job Shop Scheduling Problem*, in *2012 International Conference on Industrial Engineering and Operations Management*. 2012: Istanbul, Turkey.

86. Tasgetiren, M.F., et al. *A Discrete Differential Evolution Algorithm for the No-Wait Flowshop Scheduling Problem with Total Flowtime Criterion*. in *Computational Intelligence in Scheduling, 2007. SCIS '07. IEEE Symposium on*. 2007.

87. Wang, L., et al., *A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems.* Computers & Operations Research, 2010. **37**(3): p. 509-520.

88. Gao, K.-z., et al. *A novel grouping harmony search algorithm for the no-wait flow shop scheduling problems with total flow time criteria*. in *Computer Communication Control and Automation (3CA), 2010 International Symposium on*. 2010.

89. Wang, L., Q.-K. Pan, and M.F. Tasgetiren, *Minimizing the total flow time in a flow shop with blocking by using hybrid harmony search algorithms.* Expert Systems with Applications, 2010. **37**(12): p. 7929-7936.

90. Gao, K.-Z., Q.-K. Pan, and J.-Q. Li, *Discrete harmony search algorithm for the no-wait flow shop scheduling problem with total flow time criterion.* International Journal of Advanced Manufacturing Technology, 2011. **56**(5-8): p. 683-692.

91. Mingyue, F., et al., *A Grouping Particle Swarm Optimization Algorithm for Flexible Job Shop Scheduling Problem*, in *IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application (PACIIA '08)*, Y. Xianqing, et al., Editors. 2008: Wuhan, China. p. 332-336.

92. Pan, Q.-K., L. Wang, and Q. B, *A novel multi-objective particle swarm optimization algorithm for no-wait flow shop scheduling problems.* Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 2008. **222**(4): p. 519-539.

93. Pongchairerks, P. and V. Kachitvichyanukul, *A particle swarm optimization algorithm on job-shop scheduling problems with multi-purpose machines.* Asia-Pacific Journal of Operational Research, 2009. **26**(02): p. 161-184.

94. Pansuwan, P., N. Rukwong, and P. Pongcharoen. *Identifying Optimum Artificial Bee Colony (ABC) Algorithm's Parameters for Scheduling the Manufacture and Assembly of Complex Products*. in *Second International Conference on Computer and Network Technology (ICCNT)*. 2010.

95. Junqing, L., P. Quanke, and X. Shengxian, *Flexible job shop scheduling problems by a hybrid artificial bee colony algorithm*, in *IEEE Congress on Evolutionary Computation (CEC)*. 2011. p. 78-83.

96. Mladenović, N. and P. Hansen, *Variable neighborhood search.* Computers & Operations Research, 1997. **24**(11): p. 1097-1100.
97. Liu, H., et al., *A Novel Variable Neighborhood Particle Swarm Optimization for Multi-objective Flexible Job-shop Scheduling Problems*, in *2nd International Conference on Digital Information Management ( ICDIM '07)*. 2007: Lyon, France. p. 138-145.
98. Lei, D., Y. Zheng, and X. Guo, *A shuffled frog-leaping algorithm for flexible job shop scheduling with the consideration of energy consumption.* International Journal of Production Research, 2017. **55**(11): p. 3126-3140.
99. Lu, K., et al., *An Improved Shuffled Frog-Leaping Algorithm for Flexible Job Shop Scheduling Problem.* Algorithms, 2015. **8**(1): p. 19-31.
100. Ye, X., W. Ling, and W. Shengyao, *An effective shuffled frog-leaping algorithm for the flexible job-shop scheduling problem*, in *IEEE Symposium on Computational Intelligence in Control and Automation (CICA)*. 2013. p. 128-134.
101. Lin, J., *A hybrid biogeography-based optimization for the fuzzy flexible job-shop scheduling problem.* Knowledge-Based Systems, 2015. **78**: p. 59-74.
102. Karthikeyan, S., et al., *A hybrid discrete firefly algorithm for solving multi-objective flexible job shop scheduling problems.* International Journal of Bio-Inspired Computation, 2015. **7**(6): p. 386-401.
103. Karthikeyan, S., P. Asokan, and S. Nickolas, *A hybrid discrete firefly algorithm for multi-objective flexible job shop scheduling problem with limited resource constraints.* The International Journal of Advanced Manufacturing Technology, 2014. **72**(9-12): p. 1567-1579.
104. Mekni, S. and B.C. Fayech, *A modified invasive weed optimization algorithm for multiobjective flexible job shop scheduling problems.* International Journal of Computer Science & Information Technology, 2014. **6**(6): p. 51-60.
105. Talbi, E.-G., *Metaheuristics: from design to implementation*. Vol. 74. 2009: John Wiley & Sons.
106. Gogna, A. and A. Tayal, *Metaheuristics: review and application.* Journal of Experimental & Theoretical Artificial Intelligence, 2013. **25**(4): p. 503-526.
107. Sundaram, R.K., *A first course in optimization theory*. 1996: Cambridge university press.
108. Onar, S.Ç., et al., *A Literature Survey on Metaheuristics in Production Systems*, in *Metaheuristics for Production Systems*. 2016, Springer. p. 1-24.
109. Li, M. and D. Lei, *An imperialist competitive algorithm with feedback for energy-efficient flexible job shop scheduling with transportation and sequence-dependent setup times.* Engineering Applications of Artificial Intelligence, 2021. **103**: p. 104307.

110. Wolpert, D.H. and W.G. Macready, *No free lunch theorems for optimization.* IEEE transactions on evolutionary computation, 1997. **1**(1): p. 67-82.
111. Vilcot, G. and J.-C. Billaut, *A tabu search algorithm for solving a multicriteria flexible job shop scheduling problem.* International Journal of Production Research, 2011. **49**(23): p. 6963-6980.
112. Yazdani, M., et al., *A simulated annealing algorithm for flexible job shop scheduling problem.* Journal of Applied Sciences, 2009. **9**(4): p. 662-670.
113. Xing, L.-N., et al., *A Knowledge-Based Ant Colony Optimization for Flexible Job Shop Scheduling Problems.* Applied Soft Computing, 2010. **10**(3): p. 888-896.
114. Bihari, M. and P. Kane, *Evaluation and Improvement of Makespan Time of Flexible Job Shop Problem Using Various Dispatching Rules—A Case Study*, in *Advances in Mechanical Engineering*. 2021, Springer. p. 611-617.
115. Fan, H.-L., et al. *Survey of the selection and evaluation for dispatching rules in dynamic job shop scheduling problem*. in *Chinese Automation Congress (CAC), 2015*. 2015. IEEE.
116. Holland, J.H., *Adaptation in natural and artificial systems*. 1975, Ann Arbor, MI: University of Michigan Press.
117. Goldberg, D.E., *Genetic algorithms in search, optimization and machine learning*. 1989: Addison-Wesley Longman Publishing Co., Inc.
118. Yamamoto, K. and O. Inoue. *Applications of genetic algorithm to aerodynamic shape optimization*. in *12th Computational Fluid Dynamics Conference*. 1995.
119. Kasat, R.B., A.K. Ray, and S.K. Gupta, *Applications of genetic algorithm in polymer science and engineering.* Materials and Manufacturing Processes, 2003. **18**(3): p. 523-532.
120. Verma, G. and V. Verma, *Role and applications of genetic algorithm in data mining.* International journal of computer applications, 2012. **48**(17): p. 5-8.
121. Canyurt, O.E. and H.K. Öztürk, *Three different applications of genetic algorithm (GA) search techniques on oil demand estimation.* Energy conversion and management, 2006. **47**(18-19): p. 3138-3148.
122. Ghaheri, A., et al., *The applications of genetic algorithms in medicine.* Oman medical journal, 2015. **30**(6): p. 406.
123. Chardaire, P., et al. *Applications of Genetic Algorithms*. in *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications*. 1995. Psychology Press.
124. Karr, C. and L.M. Freeman, *Industrial applications of genetic algorithms*. Vol. 5. 1998: CRC press.

125. Roeva, O., *Real-world applications of genetic algorithms*. 2012: BoD–Books on Demand.
126. Grefenstette, J.J., *Genetic algorithms and their applications: proceedings of the second international conference on genetic algorithms*. 2013: Psychology Press.
127. Darwin, C., *On the origin of species by means of natural selection. 1859.* Ed. Joseph Carroll. Toronto: Broadview, 2003.
128. Tamaki, H. *Maintenance of Diversity in a Genetic Algorithm and an Application to the Jobshop Scheduling*. in *IMACS/SICE International Symposium on MR2*. 1992.
129. Park, J.-S., et al., *Unified Genetic Algorithm Approach for Solving Flexible Job-Shop Scheduling Problem.* Applied Sciences, 2021. **11**(14): p. 6454.
130. Chung, H. and K.-s. Shin, *Genetic algorithm-optimized long short-term memory network for stock market prediction.* Sustainability, 2018. **10**(10): p. 3765.
131. Sun, W. and Y. Xu, *Financial security evaluation of the electric power industry in China based on a back propagation neural network optimized by genetic algorithm.* Energy, 2016. **101**: p. 366-379.
132. Mirjalili, S., et al., *Genetic algorithm: Theory, literature review, and application in image reconstruction*, in *Nature-Inspired Optimizers*. 2020, Springer. p. 69-85.
133. Yan, X., et al., *Hybrid genetic algorithm for engineering design problems.* Cluster Computing, 2017. **20**(1): p. 263-275.
134. Tian, L. and C. Collins, *An effective robot trajectory planning method using a genetic algorithm.* Mechatronics, 2004. **14**(5): p. 455-470.
135. Gulsen, M., A. Smith, and D. Tate, *A genetic algorithm approach to curve fitting.* International Journal of Production Research, 1995. **33**(7): p. 1911-1923.
136. Lee, L.H., C.U. Lee, and Y.P. Tan, *A multi-objective genetic algorithm for robust flight scheduling using simulation.* European Journal of Operational Research, 2007. **177**(3): p. 1948-1968.
137. Lin, Y.-K. and Y.-Y. Chou, *A hybrid genetic algorithm for operating room scheduling.* Health Care Management Science, 2019: p. 1-15.
138. Potvin, J.-Y., *Genetic algorithms for the traveling salesman problem.* Annals of Operations Research, 1996. **63**(3): p. 337-370.
139. Caputo, A.C., et al., *Safety-based process plant layout using genetic algorithm.* Journal of Loss Prevention in the Process Industries, 2015. **34**: p. 139-150.
140. Davis, L. *Job Shop Scheduling with Genetic Algorithms*. in *Proceedings of the 1st International Conference on Genetic Algorithms*. 1985. L. Erlbaum Associates Inc.

141. Chaudhry, I.A. and A.A. Khan, *A research survey: review of flexible job shop scheduling techniques.* International Transactions in Operational Research, 2015: p. 41.
142. Xie, J., et al., *Review on flexible job shop scheduling.* IET Collaborative Intelligent Manufacturing, 2019. **1**(3): p. 67-77.
143. Gen, M. and L. Lin, *Multiobjective evolutionary algorithm for manufacturing scheduling problems: state-of-the-art survey.* Journal of Intelligent Manufacturing, 2013: p. 1-18.
144. Lei, D., *Multi-objective production scheduling: a survey.* International Journal of Advanced Manufacturing Technology, 2009. **43**(9-10): p. 926-938.
145. Lal, V. and C.A.D. Durai, *A Survey on Various Optimization Techniques with Respect to Flexible Job Shop Scheduling.* International Journal of Scientific and Research Publications, 2014. **4**(3): p. 7 pages.
146. Genova, K., L. Kirilov, and V. Guliashki, *A Survey of Solving Approaches for Multiple Objective Flexible Job Shop Scheduling Problems.* Cybernetics and Information Technologies, 2015. **15**(2): p. 3-22.
147. Zhang, J., et al., *Review of job shop scheduling research and its new perspectives under Industry 4.0.* Journal of Intelligent Manufacturing, 2019. **30**(4): p. 1809-1830.
148. Çaliş, B. and S. Bulkan, *A research survey: Review of AI solution strategies of job shop scheduling problem.* Journal of Intelligent Manufacturing, 2015. **26**(5): p. 961-973.
149. Allahverdi, A., *The third comprehensive survey on scheduling problems with setup times/costs.* European Journal of Operational Research, 2015. **246**(2): p. 345-378.
150. Chen, J.W., Wei; Rong, Gang; Fujimura, Shigeru, *Integrating Genetic Algorithm with Time Control for Just-In-Time Scheduling Problems.* IFAC-PapersOnLine, 2015. **48**(3): p. 893-897.
151. Buddala, R. and S.S. Mahapatra, *An integrated approach for scheduling flexible job-shop using teaching–learning-based optimization method.* Journal of Industrial Engineering International, 2018: p. 1-12.
152. KołOdziej, J. and S.U. Khan, *Multi-level hierarchic genetic-based scheduling of independent jobs in dynamic heterogeneous grid environment.* Information Sciences, 2012. **214**: p. 1-19.
153. Ishikawa, S.K., Ryosuke; Horio, Keiichi, *Effective hierarchical optimization by a hierarchical multi-space competitive genetic algorithm for the flexible job-shop scheduling problem.* Expert Systems with Applications, 2015. **42**(24): p. 9434-9440.
154. Mesghouni, K., et al., *Hybrid approach to decision-making for job-shop scheduling.* Production Planning & Control, 1999. **10**(7): p. 690-706.

155. Chen, H.I., Jiirgen; Lehmann, Carsten. *A genetic algorithm for flexible job-shop scheduling*. in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*. 1999. IEEE.

156. Tay, J.C.W., Djoko, *An Effective Chromosome Representation for Evolving Flexible Job Shop Schedules*, in *Genetic and Evolutionary Computation – GECCO 2004*, K. Deb, Editor. 2004, Springer Berlin Heidelberg. p. 210-221.

157. Zhang, G.G., Liang; Shi, Yang, *An effective genetic algorithm for the flexible job-shop scheduling problem.* Expert Systems with Applications, 2011. **38**(4): p. 3563-3573.

158. Saidi-Mehrabad, M. and P. Fattahi, *Flexible job shop scheduling with tabu search algorithms.* The International Journal of Advanced Manufacturing Technology, 2007. **32**(5-6): p. 563-570.

159. Pezzella, F.M., G.; Ciaschetti, G., *A genetic algorithm for the Flexible Job-shop Scheduling Problem.* Computers & Operations Research, 2008. **35**(10): p. 3202-3212.

160. Shi, Y.Z., Guohui; Gao, Liang; Yuan, Kun, *A novel initialization method for solving Flexible Job-shop Scheduling Problem*, in *International Conference on Computers & Industrial Engineering (CIE 2009)*. 2009. p. 68-73.

161. Zeb, A., et al., *Hybridization of simulated annealing with genetic algorithm for cell formation problem.* The International Journal of Advanced Manufacturing Technology, 2016. **86**(5): p. 2243-2254.

162. Zhang, H. and M. Gen, *Multistage-based genetic algorithm for flexible job-shop scheduling problem.* Journal of Complexity International, 2005. **11**: p. 223-232.

163. Asma, F.O., J.; Atidel, B. H. A.; Ibtissem, B. N., *The performance of the AGAIS (II) algorithm for the resolution of the Flexible Job-shop scheduling problem*, in *Second International Conference on Engineering Systems Management and Its Applications (ICESMA)*. 2010: Sharjah. p. 1-6.

164. Agrawal, R.P., L.N.; Kumar, S., *Scheduling of a flexible job-shop using a multi-objective genetic algorithm.* Journal of Advances in Management Research, 2012. **9**(2): p. 178-188.

165. Pandian, P.P.S., S. Saravana; Ponnambalam, S. G.; Raj, Victor, *Scheduling of Automated Guided Vehicle and Flexible Jobshop using Jumping Genes Genetic Algorithm.* American Journal of Applied Sciences, 2012. **9**(10): p. 1706-1720.

166. Ren, H.X., Han; Sun, Shenshen. *Immune genetic algorithm for multi-objective flexible job-shop scheduling problem*. in *Control and Decision Conference (CCDC), 2016 Chinese*. 2016. IEEE.

167. Butt, S.I. and S. Hou-Fang, *Application of Genetic Algorithms & Rules in Scheduling of Flexible Job Shops.* Journal of Applied Sciences, 2006. **6**(7): p. 1586-1590.

168. Xu, X.-h.Z., Ling-Li; Fu, Yue-wen, *Hybrid particle swarm optimization for flexible job-shop scheduling problem and its implementation*, in *IEEE International Conference on Information and Automation (ICIA)*. 2010. p. 1155-1159.

169. Manier, Q.Z.H.M.M.-A., *A hybrid metaheuristic algorithm for flexible job-shop scheduling problems with transportation constraints*, in *Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference*. 2012, ACM: Philadelphia, Pennsylvania, USA. p. 441-448.

170. Zhang, Q., H. Manier, and M.A. Manier, *A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times.* Computers & Operations Research, 2012. **39**(7): p. 1713-1723.

171. Kaweegitbundit, P. and T. Eguchi, *Flexible job shop scheduling using genetic algorithm and heuristic rules.* Journal of Advanced Mechanical Design, Systems, and Manufacturing, 2016. **10**(1).

172. Jang, Y.-J.K., Ki-Dong; Jang, Seong-Yong; Park, Jinwoo, *Flexible Job Shop Scheduling with Multi-level Job Structures.* JSME International Journal Series C Mechanical Systems, Machine Elements and Manufacturing, 2003. **46**(1): p. 33-38.

173. Ho, N.B. and J.C. Tay, *GENACE: an efficient cultural algorithm for solving the flexible job-shop problem*, in *Congress on Evolutionary Computation (CEC2004)*. 2004. p. 1759-1766 Vol.2.

174. Ming, W.X., Fan; Fengming, Zhang; Chaohui, Bai, *An Integrated Genetic Algorithm for Flexible Job-Shop Scheduling Problem*, in *International Conference on Computational Intelligence and Software Engineering (CiSE)*. 2010: Wuhan, China. p. 1-4.

175. Sheng-Ta, H.S.-Y., Chiu; Shi-Jim, Yen, *An improved multi-objective genetic algorithm for solving flexible job shop problem*, in *IET International Conference on Frontier Computing : Theory, Technologies and Applications*. 2010. p. 427-431.

176. Sun, W.P., Ying; Lu, Xiaohong; Ma, Qinyi, *Research on flexible job-shop scheduling problem based on a modified genetic algorithm.* Journal of Mechanical Science and Technology, 2010. **24**(10): p. 2119-2125.

177. Phanden, R.K.J., Ajai; Verma, Rajiv, *A Genetic Algorithm-Based Approach for Flexible Job Shop Scheduling*, in *Applied Mechanics and Materials*. 2011. p. 3930-3937.

178. Fan, S.C.W., J. F., *Scheduling for the flexible job-shop problem based on genetic algorithm (GA)*, in *Advanced Materials Research*. 2012. p. 616-619.

179. Wang, Y.M.Y., Hong Li; Qin, Kai Da, *A novel genetic algorithm for flexible job shop scheduling problems with machine disruptions.* International Journal of Advanced Manufacturing Technology, 2013. **68**(5-8): p. 1317-1326.

180. Hao-Chin, C.H.-T., Tsai; Tung-Kuan, Liu. *Application of genetic algorithm to optimize unrelated parallel machines of flexible job-shop scheduling problem*. in *11th IEEE International Conference on Control & Automation (ICCA)*. 2014.

181. Zambrano Rey, G.B., Abdelghani; Prabhu, Vittaldas; Trentesaux, Damien, *Coupling a genetic algorithm with the distributed arrival-time control for the JIT dynamic scheduling of flexible job-shops.* International Journal of Production Research, 2014. **52**(12): p. 3688-3709.

182. Parjapati, S.K.J., Ajai, *Optimization of Flexible Job Shop Scheduling Problem with Sequence Dependent Setup Times Using Genetic Algorithm Approach.* International Journal of Mathematical, Computational, Natural and Physical Engineering, 2015. **9**: p. 41-47.

183. Phanden, R.K.J., Ajai, *Assessment of makespan performance for flexible process plans in job shop scheduling.* IFAC-PapersOnLine, 2015. **48**(3): p. 1948-1953.

184. Wu, M.-C., et al., *Effects of different chromosome representations in developing genetic algorithms to solve DFJS scheduling problems.* Computers & Operations Research, 2017. **80**: p. 101-112.

185. Qiao, L., Z. Zhang, and M.K. Nawaz. *Genetic Algorithm Based Novel Methodology of Multi-Constraint Job Scheduling*. in *Enterprise Systems (ES), 2017 5th International Conference on*. 2017. IEEE.

186. Zhou, E., J. Zhu, and L. Deng. *Flexible job-shop scheduling based on genetic algorithm and simulation validation*. in *MATEC Web of Conferences*. 2017. EDP Sciences.

187. Jamrus, T.C., Chen-Fu; Gen, Mitsuo; Sethanan, Kanchana, *Hybrid Particle Swarm Optimization combined with Genetic Operators and Cauchy Distribution for Flexible Job-shop Scheduling Problem*, in *Asia Pacific Industrial Engineering and Management System Conference* 2013: Cebu, The Philippines. p. 12 pages.

188. Rohaninejad, M.K., Amirsaman; Fattahi, Parviz, *Simultaneous lot-sizing and scheduling in flexible job shop problems.* The International Journal of Advanced Manufacturing Technology, 2015. **78**(1-4): p. 1-18.

189. Azzouz, A., M. Ennigrou, and L.B. Said. *A self-adaptive evolutionary algorithm for solving flexible job-shop problem with sequence dependent setup time and learning effects*. in *Evolutionary Computation (CEC), 2017 IEEE Congress on*. 2017. IEEE.

190. Na, H.P., Jinwoo, *Multi-level job scheduling in a flexible job shop environment.* International Journal of Production Research, 2014. **52**(13): p. 3877-3887.

191. Frutos, M.O., Ana Carolina; Tohmé, Fernando, *A memetic algorithm based on a NSGAII scheme for the flexible job-shop scheduling problem.* Annals of Operations Research, 2010. **181**(1): p. 745-765.

192. Ida, K.O., Kensaku, *Flexible job-shop scheduling problem by genetic algorithm.* Electrical Engineering in Japan, 2011. **177**(3): p. 28-35.

193. Tavakkoli-Moghaddam, R.S., N.; Mohammadi-Andargoli, H.; Abolhasani-Ashkezari, M.H., *Duplicate Genetic Algorithm for Scheduling a Bi-Objective Flexible Job Shop Problem.* International Journal of Research in Industrial Engineering, 2012. **1**(2): p. 10-26.

194. Tung-Kuan, L.Y.-P., Chen; Jyh-Horng, Chou, *Solving Distributed and Flexible Job-Shop Scheduling Problems for a Real-World Fastener Manufacturer.* IEEE Access, 2014. **2**: p. 1598-1606.

195. ZHANG, L. and T.N. Wong, *Solving Integrated Process Planning and Scheduling Problem with Constraint Programming*, in *The 13th Asia Pacific Industrial Engineering and Management Systems Conference (APIEMS 2012) and 15th Asia Pacific Regional Meeting of the International Foundation for Production Research*. 2012: Phuket, Thailand. p. 1525-1532.

196. Guimaraes, K.F.F., M. A., *An Approach for Flexible Job-Shop Scheduling with Separable Sequence-Dependent Setup Time*, in *IEEE International Conference on Systems, Man and Cybernetics (SMC '06)*. 2006: Taiwan. p. 3727-3731.

197. Hongze, Q.W., Zhou; Hailong, Wang, *A Genetic Algorithm-Based Approach to Flexible Job-Shop Scheduling Problem*, in *Fifth International Conference on Natural Computation (ICNC '09)*, H.W.K.S.L.K.W.J. Sun, Editor. 2009: China. p. 81-85.

198. Liu, J., et al. *Research on flexible job-shop scheduling problem under uncertainty based on genetic algorithm*. in *Natural Computation (ICNC), 2010 Sixth International Conference on*. 2010. IEEE.

199. Al-Hinai, N.E., T. Y., *Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm.* International Journal of Production Economics, 2011. **132**(2): p. 279-291.

200. Ahmadi, E.Z., Mostafa; Farrokh, Mojtaba; Emami, Seyed Mohammad, *A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms.* Computers & Operations Research, 2016. **73**: p. 56-66.

201. Wang, L., C. Luo, and J. Cai, *A Variable Interval Rescheduling Strategy for Dynamic Flexible Job Shop Scheduling Problem by Improved Genetic Algorithm.* Journal of Advanced Transportation, 2017. **2017**.

202. He, Y., W. Weng, and S. Fujimura. *Improvements to genetic algorithm for flexible job shop scheduling with overlapping in operations*. in *Computer and Information Science (ICIS), 2017 IEEE/ACIS 16th International Conference on*. 2017. IEEE.

203. Zribi, N.K., I.; El-Kamel, A.; Borne, P., *Optimization by phases for the flexible job-shop scheduling problem*, in *5th Asian Control Conference*. 2004: Melbourne, Victoria, Australia. p. 1889-1895.

204. Al-Hinai, N.E., T. Y., *An efficient hybridized genetic algorithm architecture for the flexible job shop scheduling problem.* Flexible Services and Manufacturing Journal, 2011. **23**(1): p. 64-85.

205. Fard, A.R.Y., B.Y.; Khanlarzade, N., *Hybrid Genetic Algorithm for Flexible Job Shop Scheduling with Overlapping in Operations*, in *Advanced Materials Research*. 2012. p. 1499-1505.

206. Yokoyama, S.I., H.; Yamamoto, M. *Hybrid genetic algorithm with priority rule-based reconstruction for flexible job-shop scheduling*. in *Soft Computing and Intelligent Systems (SCIS), 2014 Joint 7th International Conference on and Advanced Intelligent Systems (ISIS), 15th International Symposium on*. 2014.

207. Azzouz, A., M. Ennigrou, and L.B. Said, *A hybrid algorithm for flexible job-shop scheduling problem with setup times.* International Journal of Production Management and Engineering, 2017. **5**(1): p. 23-30.

208. Nouri, H.E., O.B. Driss, and K. Ghédira, *Solving the flexible job shop problem by hybrid metaheuristics-based multiagent model.* Journal of Industrial Engineering International, 2017: p. 1-14.

209. Li, M.Y., M., *Research on improved genetic algorithm solving flexible job-shop problem.* Advanced Materials Research, 2012. **479-481**: p. 1918-1921.

210. Demir, Y. and S.K. İşleyen, *An effective genetic algorithm for flexible job-shop scheduling with overlapping in operations.* International Journal of Production Research, 2014. **52**(13): p. 3905-3921.

211. Moghadam, A.M.K.Y., Wong; Piroozfard, H. *An efficient genetic algorithm for flexible job-shop scheduling problem*. in *IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 2014* 2014.

212. Song, W.J., et al., *Flexible Job-Shop Scheduling Problem with Maintenance Activities Considering Energy Consumption.* Applied Mechanics and Materials, 2014. **521**: p. 707-713.

213. Chang, H.-C.C., Yeh-Peng; Liu, Tung-Kuan; Chou, Jyh-Horng, *Solving the Flexible Job Shop Scheduling Problem With Makespan Optimization by Using a Hybrid Taguchi-Genetic Algorithm.* IEEE Access, 2015. **3**: p. 1740-1754.

214. Driss, I.M., KinzaNadia; Laggoun, Assia, *A new genetic algorithm for flexible job-shop scheduling problems.* Journal of Mechanical Science and Technology, 2015. **29**(3): p. 1273-1281.

215. Zhang, J. and J. Yang, *Flexible job-shop scheduling with flexible workdays, preemption, overlapping in operations and satisfaction criteria: an industrial application.* International Journal of Production Research, 2016. **54**(16): p. 4894-4918.

216. Zhang, G.G., L.; Shi, Y., *A novel variable neighborhood genetic algorithm for multi-objective flexible job-shop scheduling problems*, in *Advanced Materials Research*. 2010. p. 369-373.

217. Zhang, G.G., Liang; Shi, Yang, *A Genetic Algorithm and Tabu Search for Multi Objective Flexible Job Shop Scheduling Problems*, in *International Conference on Computing, Control and Industrial Engineering (CCIE)*. 2010. p. 251-254.

218. Javadi, R.H., M., *A new method for hybridizing metaheuristics for multi-objective flexible job shop scheduling*, in *2nd International eConference on Computer and Knowledge Engineering (ICCKE)*. 2012. p. 105-110.

219. Araghi, M.E.T.J., F.; Rabiee, M., *Incorporating learning effect and deterioration for solving a SDST flexible job-shop scheduling problem with a hybrid meta-heuristic approach.* International Journal of Computer Integrated Manufacturing, 2013. **27**(8): p. 733-746.

220. Wang, S.X.Z., Chao Yong ; Jin, Liang Liang *A Hybrid Genetic Algorithm for Flexible Job-Shop Scheduling Problem.* Advanced Materials Research, 2014. **889 - 890**: p. 1179-1184.

221. Yanguang, L.G., Zhou. *The flexible job shop scheduling based on ATC and GATS hybrid algorithm*. in *Information and Automation (ICIA), 2014 IEEE International Conference on*. 2014.

222. Wang, C., et al. *A hybrid evolutionary algorithm for flexible job shop scheduling problems*. in *35th Chinese Control Conference (CCC), 2016*. 2016. IEEE.

223. Chang, H.-C. and T.-K. Liu, *Optimisation of distributed manufacturing flexible job shop scheduling by using hybrid genetic algorithms.* Journal of Intelligent Manufacturing, 2017. **28**(8): p. 1973-1986.

224. Girish, B.S.J., N., *Scheduling job shop associated with multiple routings with genetic and ant colony heuristics.* International Journal of Production Research, 2009. **47**(14): p. 3891-3917.

225. Liu, X.L., C.; Tao, Z., *Study on scheduling optimization for flexible job shop.* Applied Mechanics and Materials, 2010. **26-28**: p. 821-825.

226. Liu, X.X.L., C. B.; Tao, Z., *Research on bi-objective scheduling of dual-resource constrained flexible job shop*, in *Advanced Materials Research*. 2011. p. 1091-1095.

227. Candan, G.Y., Harun Resit, *Genetic algorithm parameter optimisation using Taguchi method for a flexible manufacturing system scheduling problem.* International Journal of Production Research, 2015. **53**(3): p. 897-915.

228. Lu, P.-H.W., Muh-Cherng; Tan, Hao; Peng, Yong-Han; Chen, Chen-Fu, *A genetic algorithm embedded with a concise chromosome representation for distributed and flexible job-shop scheduling problems.* Journal of Intelligent Manufacturing, 2015: p. 1-16.

229. Lei, D., *Co-evolutionary genetic algorithm for fuzzy flexible job shop scheduling.* Applied Soft Computing, 2012. **12**(8): p. 2237-2245.

230. Tamaki, H.O., T.; Murao, H.; Kitamura, S., *Modeling and genetic solution of a class of flexible job shop scheduling problems*, in *8th IEEE International Conference on Emerging Technologies and Factory Automation*. 2001. p. 343-350.

231. Cao, X.Y., Zhenhe, *An Improved Genetic Algorithm for Dual-Resource Constrained Flexible Job Shop Scheduling*, in *International Conference on Intelligent Computation Technology and Automation (ICICTA)*. 2011. p. 42-45.

232. Geyik, F.D., AyşeTuğba, *Process plan and part routing optimization in a dynamic flexible job shop scheduling environment: an optimization via simulation approach.* Neural Computing and Applications, 2013. **23**(6): p. 1631-1641.

233. Selvaraj, N., *An Effective Hybrid Heuristic to Solve Multiobjective Flexible Job Shop Scheduling Problems.* Academic Journal of Science, 2014. **3**(1): p. 61-73.

234. Ho, N.B.T., Joe Cing, *Solving Multiple-Objective Flexible Job Shop Problems by Evolution and Local Search.* IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 2008. **38**(5): p. 674-685.

235. Moon, I.L., Sanghyup; Bae, Hyerim, *Genetic algorithms for job shop scheduling problems with alternative routings.* International Journal of Production Research, 2008. **46**(10): p. 2695-2705.

236. Zhang, G.S., Yang; Gao, Liang, *A Genetic Algorithm and Tabu Search for Solving Flexible Job Shop Schedules*, in *International Symposium on Computational Intelligence and Design (ISCID '08)*. 2008: Wuhan, China. p. 369-372.

237. Azardoost, E.B., *A Hybrid Algorithm for Multi Objective Flexible Job Shop Scheduling Problem*, in *2nd International Conference on Industrial Engineering and Operations Management (IEOM 2011)*. 2011: Kuala Lumpur, Malaysia. p. 795-801.

238. Defersha, F.M.M., Chen, *A Coarse-Grain Parallel Genetic Algorithm for Flexible Job-Shop Scheduling with Lot Streaming*, in *International Conference on Computational Science and Engineering (CSE '09)*. 2009: Vancouver, Canada. p. 201-208.

239. Guohui, Z., *Using Matrix-Coded Genetic Algorithm for Solving the Flexible Job-Shop Scheduling*, in *International Conference on Computational Intelligence and Software Engineering (CiSE)*. 2010: China. p. 1-4.

240. Moradi, E.F.G., S. M. T.; Zandieh, M., *An efficient architecture for scheduling flexible job-shop with machine availability constraints.* International Journal of Advanced Manufacturing Technology, 2010. **51**(1-4): p. 325-339.

241. Tanev, I.T.U., Takashi; Morotome, Yoshiharu, *Hybrid evolutionary algorithm-based real-world flexible job shop scheduling problem: application service provider approach.* Applied Soft Computing, 2004. **5**(1): p. 87-100.

242. Zribi, N.E.-K., A.; Borne, P., *Total Tardiness in a Flexible Job-shop*, in *IMACS Multiconference on Computational Engineering in Systems Applications*. 2006: Beijing, China. p. 1543-1549.

243. Xing, L.-N.C., Ying-Wu; Yang, Ke-Wei, *Multi-population interactive coevolutionary algorithm for flexible job shop scheduling problems.* Computational Optimization and Applications, 2011. **48**(1): p. 139-155.

244. Jalilvand-Nejad, A.F., Parviz, *A mathematical model and genetic algorithm to cyclic flexible job shop scheduling problem.* Journal of Intelligent Manufacturing, 2013: p. 1-14.

245. Li, X. and L. Gao, *An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem.* International Journal of Production Economics, 2016. **174**: p. 93-110.

246. Mokhtari, H. and A. Hasani, *An energy-efficient multi-objective optimization for flexible job-shop scheduling problem.* Computers & Chemical Engineering, 2017. **104**: p. 339-352.

247. Pan, Y.Z., W. X.; Gao, T. Y.; Ma, Q. Y.; Xue, D. J., *An adaptive Genetic Algorithm for the Flexible Job-shop Scheduling Problem*, in *IEEE International Conference on Computer Science and Automation Engineering (CSAE)*. 2011. p. 405-409.

248. Liang, D.T., Ze, *A genetic algorithm with Tabu Search for multi-objective scheduling constrained flexible job shop*, in *Cross Strait Quad-Regional Radio Science and Wireless Technology Conference (CSQRWC)*. 2011: Harbin, CHINA p. 1662-1665.

249. Hui, H., *Approach for multi-objective flexible job shop scheduling.* Advanced Materials Research, 2012. **542-543**: p. 407-410.

250. Shahsavari-Pour, N.G., Behrooz, *A novel hybrid meta-heuristic algorithm for solving multi objective flexible job shop scheduling.* Journal of Manufacturing Systems, 2013. **32**(4): p. 771-780.

251. Azzouz, A.E., Meriem; Jlifi, Boutheina. *Diversifying TS using GA in multi-agent system for solving Flexible Job Shop Problem*. in *Informatics in Control, Automation and Robotics (ICINCO), 2015 12th International Conference on*. 2015. IEEE.

252. Zandieh, M., I. Mahdavi, and A. Bagheri, *Solving the Flexible Job-Shop Scheduling Problem by a Genetic Algorithm.* Journal of Applied Sciences, 2008. **8**(24): p. 4650-4655.

253. Xiu-li, W.S.-J., Li *Mass Variety and Small Batch Scheduling in the Flexible Job Shop*, in *2nd International Conference on Biomedical Engineering and Informatics (BMEI '09)*, R.F. Shi, Wenjiang; Wang, Yuanquan; Wang, Huaibin, Editor. 2009. p. 1-7.

254. Saravanan, M.N.E.C.D., *Single Objective Evolutionary Algorithm for Flexible Job-shop Scheduling Problem.* International Journal of Mathematics Trends and Technology, 2012. **3**(2): p. 78-81.

255. Yang, X.W., Jianming; Hou, Minglei; Fan, Xiaoliang. *Job shop scheduling based on genetic algorithm using Matlab*. in *2015 IEEE Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. 2015. IEEE.

256. Chan, F.T.S.W., T. C.; Chan, L. Y., *Flexible job-shop scheduling problem under resource constraints.* International Journal of Production Research, 2006. **44**(11): p. 2071-2089.

257. Goodman, P.U.E., *Solving multiobjective flexible job-shop scheduling using an adaptive representation*, in *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. 2010, ACM: Portland, Oregon, USA. p. 737-742.

258. Li, X., et al. *An effective multi-swarm collaborative evolutionary algorithm for flexible job shop scheduling problem*. in *Computer Supported Cooperative Work in Design (CSCWD), 2011 15th International Conference on*. 2011. IEEE.

259. Zribi;, N., et al., *Minimizing the total tardiness in a flexible job-shop*, in *International Symposium on Intelligent Automation and Control, World Automation Congress*. 2006: Budapest, Hungary.

260. Vilcot, G.B., J.; Esswein, C., *A Genetic Algorithm For A Bicriteria Flexible Job Shop Scheduling Problem*, in *International Conference on Service Systems and Service Management*. 2006. p. 1240-1244.

261. Lee, S.M., Ilkyeong; Bae, Hyerim; Kim, Jion, *Flexible job-shop scheduling problems with 'AND'/'OR' precedence constraints.* International Journal of Production Research, 2012. **50**(7): p. 1979-2001.

262. Palacios, J.J.G., Miguel A.; Vela, Camino R.; González-Rodríguez, Inés; Puente, Jorge, *Genetic tabu search for the fuzzy flexible job shop problem.* Computers & Operations Research, 2015. **54**: p. 74-89.

263. Morinaga, Y., M. Nagao, and M. Sano, *Balancing setup workers' load of flexible job shop scheduling using hybrid genetic algorithm with tabu search strategy.* International Journal of Decision Support Systems, 2016. **2**(1-3): p. 71-90.

264. Gao, J.G., Mitsuo; Sun, Linyan, *Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm.* Journal of Intelligent Manufacturing, 2006. **17**(4): p. 493-507.

265. Zribi, N.K., I.; El Kamel, A.; Borne, P., *Assignment and Scheduling in Flexible Job-Shops by Hierarchical Optimization.* IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 2007. **37**(4): p. 652-661.

266. Yi, W., X. Li, and B. Pan, *Solving flexible job shop scheduling using an effective memetic algorithm.* International Journal of Computer Applications in Technology, 2016. **53**(2): p. 157-163.

267. Li, J.J.M.W.K.M.X.L.J., *Hybrid Genetic Algorithm for Flexible Job-shop Scheduling with Multi-objective.* Journal of Information and Computational Science, 2011. **8**(11): p. 2197- 2205.

268. Zhou, D.L.Z., *A Flexible Job-shop Scheduling Method Based on Hybrid Genetic Annealing Algorithm.* Journal of Information and Computational Science, 2013. **10**(17): p. 5541-5549.

269. Li, J.P., Q.; Xie, S, *A Hybrid Variable Neighborhood Search Algorithm for Solving Multi-Objective Flexible Job Shop Problems*. Computer Science and Information Systems, 2010. **7**(4): p. 907-930.

270. Tayebi Araghi, M.E.J., F.; Rabiee, M., *Incorporating learning effect and deterioration for solving a SDST flexible job-shop scheduling problem with a hybrid meta-heuristic approach.* International Journal of Computer Integrated Manufacturing, 2013. **27**(8): p. 733-746.

271. Türkyılmaz, A.B., Serol, *A hybrid algorithm for total tardiness minimisation in flexible job shop: genetic algorithm with parallel VNS execution.* International Journal of Production Research, 2015. **53**(6): p. 1832-1848.

272. Doh, H.-H., et al., *A priority scheduling approach for flexible job shops with multiple process plans.* International Journal of Production Research, 2013. **51**(12): p. 3748-3764.

273. Xia, W. and Z. Wu, *An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems.* Computers & Industrial Engineering, 2005. **48**(2): p. 409-425.

274. Kang, Y.W., Z. M.; Lin, Y.; Zhang, Y. F., *An improved genetic algorithm for flexible job-shop scheduling problems*, in *Advanced Materials Research*. 2013. p. 345-348.

275. Xiong, J.T., Xu; Yang, Ke-wei; Xing, Li-ning; Chen, Ying-wu, *A Hybrid Multiobjective Evolutionary Approach for Flexible Job-Shop Scheduling Problems.* Mathematical Problems in Engineering, 2012. **2012**: p. 1-27.

276. Teekeng, W.T., Arit. *Modified genetic algorithm for flexible job-shop scheduling problems*. in *Procedia Computer Science*. 2012.

277. Qiao, W.L., Qiaoyun, *Solving the Flexible Job Shop Scheduling Problems Based on the Adaptive Genetic Algorithm*, in *International Forum on Computer Science-Technology and Applications IFCSTA '09)*, Z. Qihai, Editor. 2009. p. 97-100.

278. Yang, J.J.J., L. Y.; Liu, B. Y., *The improved genetic algorithm for multi-objective flexible job shop scheduling problem*, in *Applied Mechanics and Materials*. 2011. p. 870-875.

279. Mastrolilli, M. *Flexible Job Shop Problem*. Available from: https://people.idsia.ch//~monaldo/fjsp.html.

280. Amjad, M.K., et al., *A layered genetic algorithm with iterative diversification for optimization of flexible job shop scheduling problems.* Advances in Production Engineering & Management, 2020. **15**(4): p. 377-389.

281. Amjad, M.K., S.I. Butt, and N. Anjum. *Improved Genetic Algorithm Integrated with Scheduling Rules for Flexible Job Shop Scheduling Problems*. in *5th International Conference on Power, Energy and Mechanical Engineering*. 2021. Shanghai, China.

282. Behnke, D. and M.J. Geiger, *Test Instances for the Flexible Job Shop Scheduling Problem with Work Centers*. 2012, Universitätsbibliothek der Helmut-Schmidt-Universität: Hamburg.

283. Kennedy, J. and R. Eberhart. *Particle swarm optimization*. in *IEEE International Conference on Neural Networks*. 1995.

284. Eberhart, R. and J. Kennedy. *A new optimizer using particle swarm theory*. in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science (MHS '95)*. 1995.

285. Wen, C.D., Lei; Tao, Wang; Qiongfang, Zhang, *GA and SA based Evolutionary algorithm for fuzzy flexible job shop scheduling*, in *8th World Congress on Intelligent Control and Automation (WCICA)*. 2010. p. 688-693.

286. De Giovanni, L.P., Ferdinando, *An improved genetic algorithm for the distributed and flexible job-shop scheduling problem.* European journal of operational research, 2010. **200**(2): p. 395-408.

287. Bagheri, A., et al., *An artificial immune algorithm for the flexible job-shop scheduling problem.* Future Generation Computer Systems, 2010. **26**(4): p. 533-541.

288. Özgüven, C.Ö., Lale; Yavuz, Yasemin, *Mathematical models for job-shop scheduling problems with routing and process plan flexibility.* Applied Mathematical Modelling, 2010. **34**(6): p. 1539-1548.

289. Zhou, W.B., Yan-ping; Zhou, Ye-qing, *An improved genetic algorithm for solving flexible job shop scheduling problem*, in *25th Chinese Control and Decision Conference (CCDC)*. 2013: Guiyang, China. p. 4553-4558.

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