NATIONAL UNIVERSITY OF SCIENCES & TECHNOLOGY
COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING,
RAWALPINDI, PAKISTAN

Multi-criteria optimization of process plans for reconfigurable manufacturing systems: An evolutionary approach

By
Ms. Taiba Zahid
MS-68 (Mechanical Engineering)
(2011-NUST-MS-PhD-Mech-25)

Thesis submitted to the faculty of the Department of Mechanical Engineering, National University of Sciences and Technology College of Electrical and Mechanical Engineering, Rawalpindi PAKISTAN in partial fulfillment of the requirements for the degree of Masters

Thesis Supervisor

Associate Prof. Dr. Aamer A. Baqai

Multi-criteria optimization of process plans for reconfigurable manufacturing systems: An evolutionary approach

By
Ms. Taiba Zahid

Submitted to the Faculty of Department of Mechanical Engineering, National University of Sciences and Technology College of Electrical and Mechanical Engineering, Rawalpindi Pakistan, in partial fulfillment of the requirement for the degree of
Master of Science in Mechanical Engineering

Candidate: ____
Ms. Taiba Zahid

Advisor: ____
Associate Professor Dr. Aamer A. Baqai

Committee Members:

1. Assistant Prof. Dr. Imran Akhtar

2. Assistant Prof. Dr. Sajidullah Butt

3. Assistant Prof. Dr. Hasan Aftab Saeed

National University of Sciences and Technology
College of Electrical & Mechanical Engineering
Rawalpindi

29 April, 2013

ABSTRACT

Production systems have developed over the years due to changing environment, external and internal drivers and conditions like new technologies, developed products and customer needs. These needs were the main drivers for integrated and evolved manufacturing systems which can be more responsive and customer focused. Prototypes of manufacturing industries which have been recently introduces as flexible and reconfigurable manufacturing systems are responding to these recent needs in peculiar ways focusing not only on product design level but also on integrated manufacturing systems and process planning level.

Optimization is central to any problem involving decision making whether in engineering or any other field of life. Manufacturing problems are considered as complex combinatorial problems involving non linear constraints and large solution search space. The area of optimization has received enormous attention in recent years, primarily because of the rapid development in computer technology, advance manufacturing methods and products. Many software packages now come with a special optimization tool like Optimization toolbox of MATLAB. When these techniques are used to solve the design problems in engineering then this becomes engineering optimization or design optimization. The aim is to fill the gap between theory of optimization and engineering practices. Since most of the engineering problems are NP hard problems which cannot be solved by conventional techniques that lead to the promotion of developing advance techniques and search methods in optimization.

Evolutionary Algorithms is a class of evolutionary computation which uses mechanisms inspired by evolution. In this work, methodologies are presented to solve NP-Hard problem of process planning through evolutionary optimization using Genetic Algorithms and Cuckoo Search to generate and then find the optimized process plan for a part or part family. Furthermore, an approach has been provided in order to achieve reconfigurability, accommodating new feature in the already generated process plan and thus creating a hybrid between generative and variant process planning approach.

ACKNOWLEDGMENT

I would like to thank Almighty Allah; whose guidance lead this work to be completed in time and whose blessings and benevolent help kept me sheltered all the time.

First and foremost I offer my sincerest gratitude to my supervisor, Dr Aamer Baqai, who has supported me throughout my thesis with his patience and knowledge whilst allowing me the room to work in my own way. I attribute the level of my Masters degree to his effort and encouragement and without him this thesis, too, would not have been completed or written. One simply could not wish for a better or friendlier supervisor. Along with technical insight, he continually conveyed a spirit of adventure in research and an excitement in regard to teaching.

I am indebted to the College of Electrical and Mechanical Engineering, National University of Sciences and Technology resources for this research work.

I would like to thank my beloved parents and family, who always guided me in the right way and whose endless effort and support made it possible for me, to be, what I am today. I would not forget to thank my sister and my friend Ms. Aida Zahid who helped me a lot in my thesis work.

I have been fortunate to met and work with many wonderful people like Prof. Waheed ul Haq, Asst. Prof. Imran Akhtar, Asst. Prof. Hasan Aftab Saeed, Asst. Prof. Sajidullah Butt and Mr. Muhammad Saif ullah Khaild who made my time at college enjoyable and whose continues encouragement made it possible for me to reach all the way to this point.

In the end I would like to thank the entire honorable faculty of Mechanical Department of the college, whose professional approach and vision groomed me.

Taiba Zahid

April 2013

Contents

List of Tables

List of Figures

List of Abbreviations

DML	Dedicated Manufacturing Line
DMS	Dedicated Manufacturing System
FMS	Flexible Manufacturing System
RMS	Reconfigurable Manufacturing System
RMT	Reconfigurable Machine Tool
RAS	Reconfigurable Assembly System
CAPP	Computer Aided Process Planning
KC	Kinematic Configuration
EA	Evolutionary Algorithms
ACO	Ant Colony Optimization
PSO	Particle Swarm Optimization
GA	Genetic Algorithm
CS	Cuckoo Search
SA	Simulated Annealing
TAD	Tool Approach Direction
PGM	Precedence Group Matrix

Chapter 01: Introduction to Reconfigurable Manufacturing Systems

1.1 Introduction

Manufacturing industry has an ability to bring wealth for any organization that is the core purpose of all organizations directly or indirectly. To sustain in the world which has become a global village, industries have to focus on improving quality while maintaining price in a range that gives competitive advantage and is attractable for customers. A pertinent manufacturing system will be able to fulfill that purpose.

Due to the aforesaid cause, production industries are in a continuous race, focusing towards total quality management and trying to improve their products in order to get competitive advantage. Such improvement can be brought from higher quality products, or by lowering cost keeping quality in mind or by timely supply and innovative products. This leads to the need of application of new technologies and concepts not only for product design but also for the manufacturing system and entire setup.

There can be many types of manufacturing systems extending from dedicated and fixed to fully automatic and flexible, from isolated facilities or job shops to computer integrated systems. The level of production volume, demand, type of customer and product are the main factors that determine whether it has to be a job shop, fixed, discrete or continuous production system. Thus multiple factors have to be kept in mind for the selection of manufacturing systems.

Combat is deducted from many factors like responsiveness, good service, customer satisfaction, quality and less waste. The aim of choosing such a manufacturing system that utilizes the optimal resources and gives competitive advantage in lowest capital investment gives designers and managers a big challenge since it is a multi-objective task with many conflicting objectives in a way that increasing one decreases the other. The aforesaid factors are the reason due to which advance manufacturing systems are replacing traditional manufacturing systems in today’s world due to potential benefits associated with them.

1.1.1 Motivation

Production systems have developed over the years due to changing environment and conditions like new technologies, developed products and customer needs. These were the external drivers but along with them there were internal drivers as well such as to keep the waste and scrap minimal, increase productivity, reduce inventory and labor costs etc. These needs were the main drivers for integrated and evolved manufacturing systems which can be more responsive and customer focused.

1.2 Justification for Advance Manufacturing Systems

Prototypes of manufacturing industries which have been recently introduced as flexible and reconfigurable manufacturing systems are responding to these recent needs in peculiar ways. These systems can be responsive and can provide necessary transformations whenever required at different levels.

The main factors or drivers leading towards the shift from traditional manufacturing systems to advance manufacturing can be summarized in the form of variation and uncertainty. These factors would be discussed further in detail.

1.2.1 Changeability/Variations

To compete in global market and production networks, manufacturing systems should be transformed in such a way that processes, products and facilities lay out should be made adaptable so that it can be changed due to variations in external or internal drivers mentioned above. This quality of manufacturing industries to be adaptable in a constantly evolving manufacturing environment can be described as "changeability",

The need for reconfigurability or adaptability is switched by external and internal drivers whose main source is the turbulence in market conditions and customer demands. To keep customers attracted and be center of attention, designers need to bring variety in their products at design level. This would be small level change but a major change can also be adopted such as to enter a new market and to target a whole new type of customers.

Changeability or responsiveness is required due to uncertainty in a manufacturing system. It is required at different levels in a manufacturing system. Uncertainty for factory, assembly, process planning and manufacturing level would be discussed further in detail.

Factory Level

At **factory level** due to the product variations, shop floor has to be changed and layout needs to be changed due to the changing demands. This means a need for a dynamic facility planning. In addition to that it is needed to scale up and down the production volumes according to the demands. Along with that inventory control would also has to be flexible.

Assembly Level

Assembly lines or process lines have to be transformable. With the changing product variety, mobility needs and process lines would be different and one has to reconfigure them.

Process Planning and Control Level

Process planning is a major concern for companies having various part families. Obtaining optimized machine configuration for each part family and then among the part families, determining the one giving the most benefit is an NP hard problem. They have to be adoptable, adjustable and evolvable. One of the major challenges for today's process planners is to design systems in such a way so as to accommodate variations at product design level as well as manufacturing system level for the upcoming suture. Also the availability of resources has to be kept in consideration. This particular need of responsiveness at process planning and control level is a major push for advanced reconfigurable manufacturing systems.

Manufacturing Level

Manufacturing systems undergo incremental changes in modern world of industrialization. To maintain position in such competitiveness, a company should be able to have a manufacturing system that fulfills the present demands and can also be reconfigured for the future demands. It can be considered as a tool for implementing computer integrated manufacturing.

1.2.2 Hierarchy of Product/Process Variations

Market demands, research and development, awareness, science and technology, cost reduction, corporeal changes and legal rules and push the evolution of products. Product designs are evolved with time in response to these requirements. Changes in design of product may lead to change in functions and forms that may produce a whole new set of part family with different design features. This gives rise to product families that contain variants of the products and their parts, components and configurations. Five apparent levels in the hierarchy are described as:

a. Product Features
b. Product/ Part Family
c. Products families
d. Sub assemblies/part components
e. Products Families

1.2.3 Uncertainty

Frequent and unpredictable market changes are challenges facing manufacturing enterprises at present. In the short term, there are many triggers for products changes including evolving over time due to innovation. They also experience significant evolutions in the long term due to products and technological changes.

To increase competitive advantage of company and to have a customized product or features through which company stands out among others research and developers always aim for innovative products.

This innovativeness leads to uncertainty because production teams do not know that what might be their future product. So to ensure that those uncertain features can be developed in the system, it has to be made reconfigurable i.e. flexibility when needed.

1.3 Comparison of Manufacturing Systems

The traditional manufacturing systems do not fit into the present marketing world and the shift was needed. Since dedicated systems were only capable of producing specialized products because of their design. In dedicated manufacturing lines (DML), each line is specifically designed to build a particular/product. Since line is specifically designed for that purpose, it will take a lesser time to be made and as long as the part demand is high making production volume higher, unit cost of the product will be low and cost for purchasing specialized equipment will be compensated. Figure 1 shows a dedicated manufacturing line for a cement plant designed specifically for this purpose.

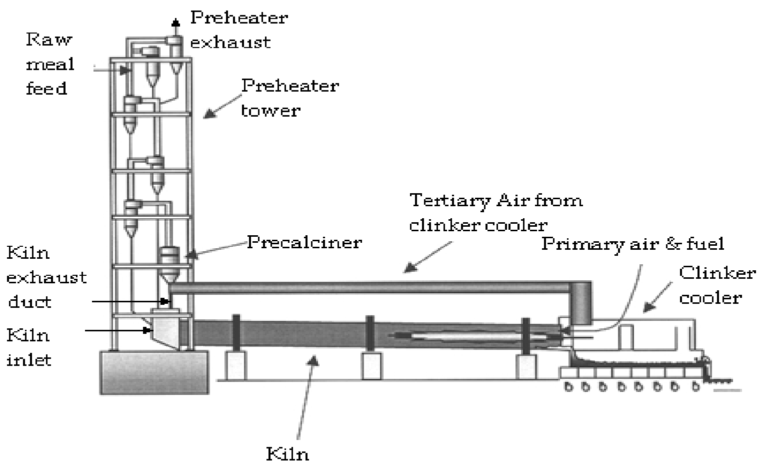


Figure 1: An Example of Dedicated Manufacturing Line [1]

As customer's demands changed, flexible manufacturing systems (FMS) were developed to accommodate various needs. Flexible manufacturing systems aimed for increasing variety of products. So the best application of FMS was found in the production system having larger product variety in small amount. Flexible manufacturing systems consisted of computerized numerically controlled machines and manufacturing systems were designed to be automated. Flexibility was introduced at different levels in a manufacturing system. Figure 2 shows a flexible manufacturing system proposed by S.S Sankar et al [1]. It is composed of different flexible subsystems making a complete FMS. Each circle represents a flexible machine cell (FMC) that consists of two or three machines with automatic tool changer and pallet changer. Each cell has a robot for loading/unloading of the parts. Automated guided vehicles (AGVs) are present for handling the part from the machine cells and loading/unloading them from the automated storage/retrieval system (AS/RS).

[1] http://www.intechopen.com/books/alternative-fuel

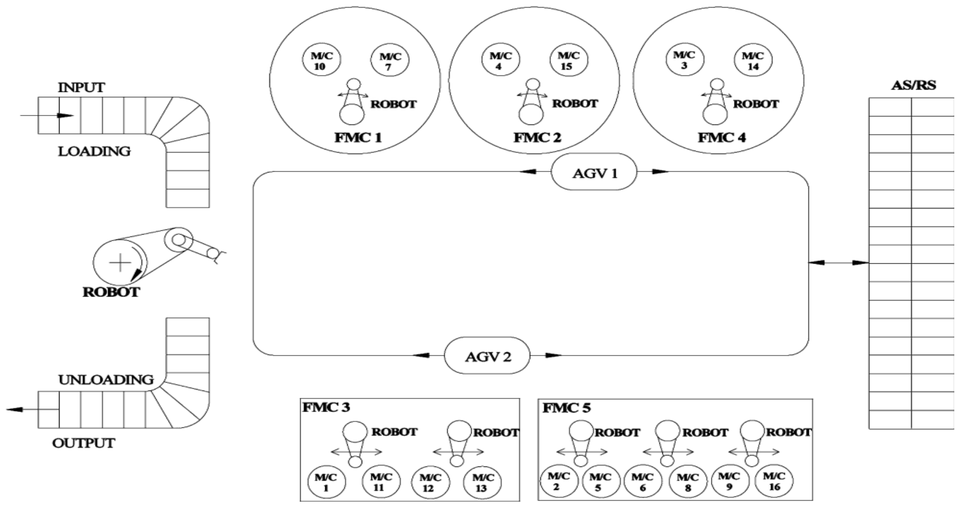


Figure 2: An Example of Flexible Manufacturing System [1]

But these systems were made with flexibilities which in most cases were not needed. The equipment required for these systems was much expensive and the initial capital costs were high. That resulted in the increased cost of manufacturing system and in turn of the product. So these systems were not adopted completely but at some levels of manufacturing.

RMS aims for customized flexibility meaning flexibility when needed. It aims to increase responsiveness of a system. It can be seen as an extension of Just in time (JIT) production system. This customized flexibility leads to a lower setup as well as product cost as compared to FMS and also faster throughput and higher production rates. Figure 3 shows comparison between these systems in terms of their responsiveness. FMS and DMS are static systems but RMS evolves with time [2].

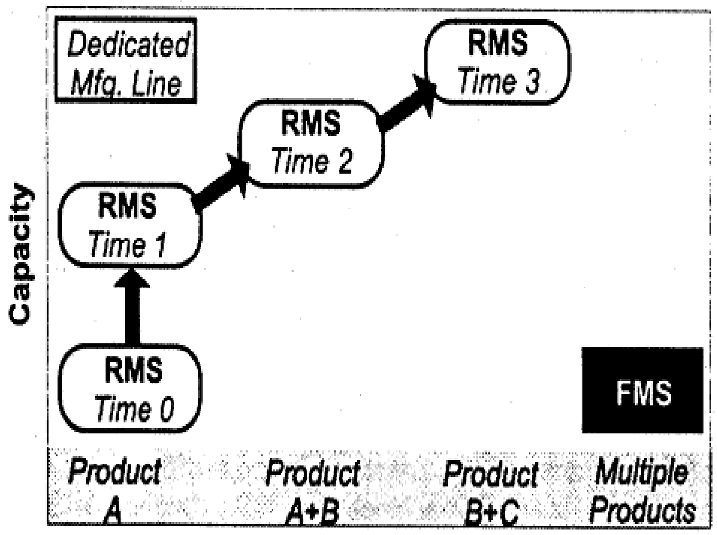


Figure 3: Responsiveness of RMS with the change in product [33]

However integration of machines and selection of machine configuration becomes a tough task for RMS. Table 1 represents general comparison of three basic types of manufacturing systems.

Table 1: Difference between Manufacturing Systems

Features	**DMS**	**FMS**	**RMS**
Aim	Specialized Product	Increased Variety	Increased Responsiveness
Product Variety	Specialized products	High	Customized
Quantity	High	Low	High
Machine Configuration	Dedicated	Flexible	Reconfigurable
Flexibility	No	High	Customized(when needed)
Scalability	No	Yes	Yes
Responsiveness (after market review)	Lowest	Medium	Fast
Market	Stable	Predictable	Uncertain
Process Technology	Fixed	Needs acceptability	Responsive
System Focus	Part	Machine	Part Family
System Structure	Fixed	Adjustable	Adjustable
Manufacturing Policy	Pushing	Pulling	Customizing
Cost	Low	High	Intermediate

Figure 4 shows the comparison of manufacturing systems at the variety-volume scale. As discussed, DMS tends to have the highest production but the lowest variety.

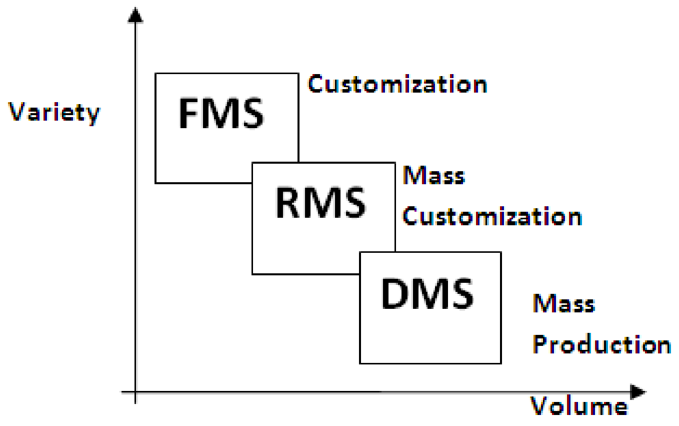


Figure 4: Comparison of DMS, FMS and RMS on Variety-Volume Scale

1.3.1 Reconfigurable Manufacturing Systems

The research work regarding the design and requirements of reconfigurable manufacturing systems in the past ten years resulted in a state-of-the-art definition that Y. Koren [2] summarized as follows:

'*A RMS is a system designed at the outset for rapid changes in structure, as well as in its machines and controls, in order to rapidly adjust production capacity and functionality (within a part family).'*

The basic messages here–in contrast to configurable systems –are 'rapid changes' or 'rapid adjustment' which must happen in relatively short time ranging between minutes and hours and not days or weeks. The reconfigurable manufacturing systems have two important components termed as reconfigurable machine tool and reconfigurable controller.

Reconfigurable Machine Tool (RMT)

The uniqueness of reconfigurable manufacturing system is that the structure of the system as well as of its machines and control can be rapidly changed in response to market changes. A major component of RMS is the reconfigurable machine tools (RMT). RMT are designed for customized range of operations requirement and may be cost effectively reconfigured if the requirements are changed. So they are designed to produce specific range of operations for a specific range of cycle time. The primary aim of RMT is to cope with the changes that a product, part or a system may face.

Reconfigurable Controller

To control a particular machine, specific functions or classes currently must be designed and build into reconfigurable controller. The controller becomes unchangeable at run time for controlling different machines. The controller can be dynamically reconfigured for different machine.

1.3.2 Characteristics of RMS

Ideally a reconfigurable manufacturing system possesses six core characteristics. Yorem Koren [2] introduced these characteristics which would be discussed in described in detail.

Modularity

In RMS components have to be modular such as spindle axis or machine configuration so that the modular components can be changed according to the part design. Modularity is the most important component of RMS that provides feature of customized flexibility.

Diagnosability

It is the characteristic that adds the factor of quality assurance and control by checking the current status of the system and diagnosing the root cause of the problem. It has two main aspects which are:

a. Detecting machine flaw

b. Detecting part flaw

Convertibility

Convertibility is the characteristic that gives responsiveness to the system. System convertibility may have several levels. Conversion may require switching spindles on a milling machines or manual adjustment of passive degrees-of-freedom changes when switching production between two members of the part family within a given day. System conversion at this daily level must be carried out quickly to be effective.

Scalability

The ability to easily change production capacity by rearranging an existing manufacturing system and/or changing the production capacity of reconfigurable stations is known as scalability. Scalability is the counterpart characteristic of convertibility. Scalability may require at the machine level adding spindles to a machine to increase its productivity, and at the system level changing part routing or adding machines to expand the overall system capacity (i.e., maximum possible volume) as the market for the product grows.

Integrability

Integrability is the ability to integrate modules rapidly and precisely by a set of mechanical, informational, and control interfaces that enable integration and communication. At the machine level, axes of motions and spindles can be integrated to form machines. In addition, machine controllers can be designed for integration into a factory control system.

Customized Flexibility

To design system/machine flexibility just around a product family, obtaining thereby customized-flexibility, as opposed to the general flexibility of FMS/CNC is termed as customized flexibility. This characteristic drastically distinguishes RMS from flexible manufacturing systems (FMS), and allows a reduction in investment cost. Customized flexibility for the part family allows the utilization of multiple tools (e.g., spindles in machining or nozzles in injection molding) on the same machine, thereby increasing productivity at reduced cost without compromising flexibility.

1.3.3 Development in RMS

The concept of RMS has rocked the world and promised a global revolution in the 21st century manufacturing landscape [3]. From 20th century, a lot of research has been going on in this area and a lot papers have been archived. The term RMS can be attributed to Dr. Koren. Yoram Koren [2] proposed to the National Science Foundation to establish an Engineering Research Center (ERC) for Reconfigurable Manufacturing Systems (RMS), thereby initiating the RMS paradigm in 1995 in the university of Michigan. The RMS is a manufacturing system that is designed at the outset for rapid changes in its structure to quickly adjust its production capacity and system functionality in response to sudden market changes and customer demands. Delphy study of manufacturing [4], RMS concept was identified as number one priority technology for future manufacturing and one of six key research challenges.

The charter of the ERC-RMS is bringing the RMS science to the factory floor with new scientific methodologies and innovative equipment that enable to begin production faster and with higher productivity and improved part quality. RMS technologies give manufacturers “exactly the production capabilities needed, exactly when needed” to compete in the global marketplace.
Over thirty industrial companies are the founding partners of the ERC-RMS Center. They include machine tool builders and control vendors, as well as end-users such as the auto and aerospace industries and power train builders.
A lot of research is going on to develop reconfigurable machine tool structure. Koren with other researchers have developed an arch type RMT for machining a variety of part family and in particular, low cost RMT that is able to perform a variety of machining operations.

The advantage of this designed RMT is that it permits easy reconfiguration of machine structure and its spindle units to produce a new series of part belonging to the same part family. It includes multiple support units which are able to support multiple desired tools. The base of RMT has several slots positioned in generally circular orientation around the table. Table is at the center of base and can be moved horizontally, vertically or tilted with respect to base. Spindles can be mounted on support units at any desired position and support units can be connected to the base through any slot thus providing easy reconfigurable machine structure and customized flexibility (Koren et al [2]).

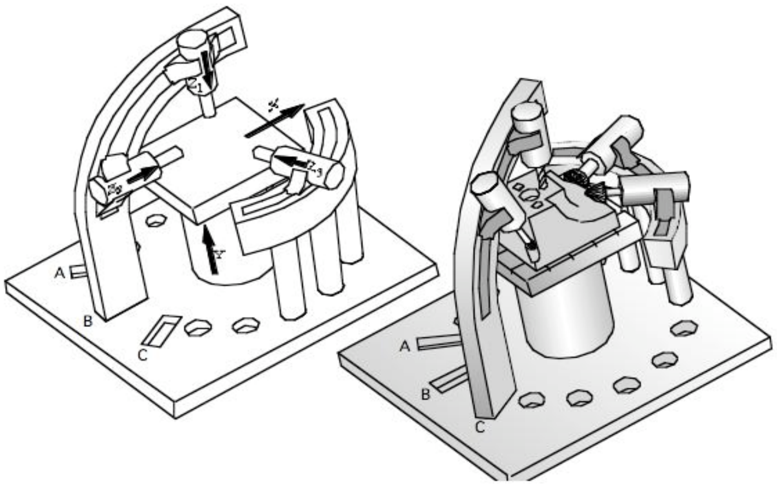


Figure 5: Patented Design of Arch type RMT [2]

Research has been carried out on Reconfigurable Factory Test Bed (RFT). By providing a collaborative mix of hardware/software and real/simulation components distributed across a web-enabled network, the RFT will serve as an excellent environment to consolidate and showcase results of the University of Michigan's Engineering Research Center for RMS. More importantly, it provides an environment to envision, rapidly prototype, and verify in a factory operation environment those solutions that result from the combination of the RFT components. Further it will provide a mechanism for pre-validation of manufacturing software components to standard specifications. Along with RFT, researchers are also working on developing the generic manufacturing automation test bed intended for application to a range of different manufacturing sectors.

Apart from RMT, another element of RMS is reconfigurable assembly system (RAS). Assembly is very important part of manufacturing and it is observed that 20% to 50% of the total cost is assembly cost [5]. DFA is being considered as enabling technology for RAS. The objective of DFA is to optimize a product design with consideration of its production system including supplies, material-handling systems, manufacturing and assembly processes, labor force, distribution systems and customers [3].

Robots are being used for RAS. They can be reprogrammed according to accomplish different assembly tasks. Today, most RAS are developed using industrial robots [5]. So RMS is one of the most promising paradigms that can provide solutions to changes and uncertainties in a competitive manufacturing environment.

1.4 Process Planning - Interconnect between Changing Products & Manufacturing

Changing product designs with different features need efficient sequence planning. It is a phenomenal concern in process planning to make them efficient and responsive in such a way that they are able to accommodate various part features in products. With the addition of any member in the group, this process plan has to be transformed again that justifies the importance of RMS at process planning and control level. Process planning and sequencing/scheduling provide phenomenal links between multiple generations of product families and their respective design features, capacities and layouts of production systems throughout their entire lifecycle.

1.4.1 Process Planning

Process Planning can be defined as a procedure that focuses on selecting resources for use in the execution and completion of a project. In a manufacturing setting, this aspect of planning also includes establishing the general sequence of steps that begin with the acquisition of materials and end with the creation of a finished product.

The practice of process planning in a manufacturing provides precise and clear sequential directions about how the product is to be routed and fabricated in a manufacturing facility. In advance manufacturing systems, this will influence how the facility will be designed and laid out in preparation for the new product.

1.4.2 Factors effecting Process Selection

While determining processes for a product, several factors need to be kept in mind. Selecting a process without considering the influence of these factors on it could adversely affect the cost, quality and ease of manufacturing the parts. Table 2 describes some of the factors that influence process selection.

Table 2: Influence of Factors on Process Selection

Factor	**Potential Influences**
Materials	Some materials require a different process, for example aluminium can be casted while wood cannot.
Quantity	Large lot sizes justify expensive tooling/process
Surface Finish Requirements	Different processes produce different tolerances
Tolerance	Tight tolerances require expensive tooling
Structural Strength	According to the use of product

Apart from the process selection, operation sequence is also a major task. It has to be designed keeping in mind various factors so as to reduce lead time of the product. Each process may has to be performed a number of times for manufacturing. Hence operation sequences must be selected in a way that minimum time should be given to part handling and tool setting activities. Hence these scheduling problems can be differentiated and compiled based upon a number of parameters such as machining time, machine configuration, and precision and so on.

1.4.3 Process Plan Generation

Process planning is an activity that brings designers and manufacturers at one platform. F.Villeneuve et al [6] summarized the approaches which are being used for generation of process plans. Traditionally, in CAPP, designers generated process plan while considering fixed and specific components. Due to this reason only one process plan was generated which was supposed to produce one type of product. So it was related to DMS and there was no product variability and static systems were developed. With the passage of time, researchers focused towards generating multiple process plans that can be compatible with the dynamic nature of manufacturing systems such as FMS and RMS.

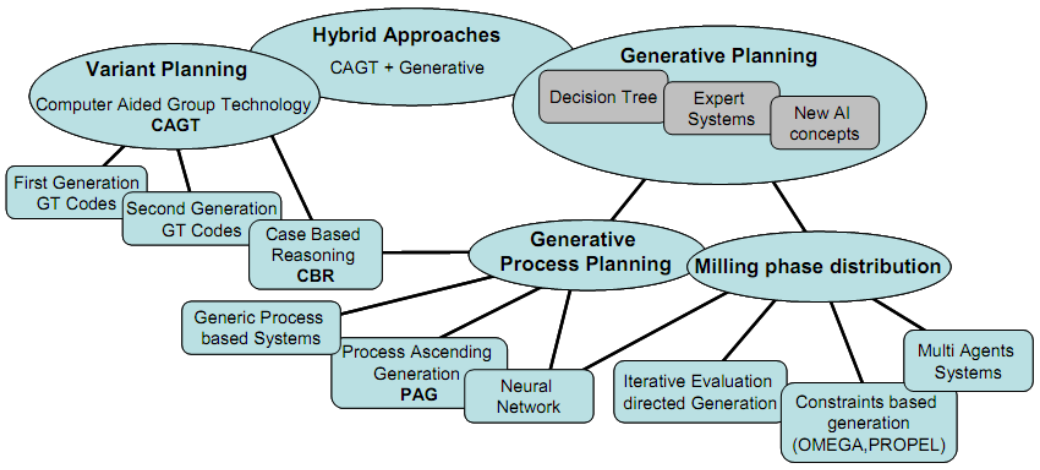

Hybrid Approaches
CAGT + Generative
Variant Planning
Computer Aided Group Technology
CAGT
Generative Planning
Decision Tree
Expert Systems
New AI concepts
First Generation GT Codes
Second Generation GT Codes
Case Based Reasoning CBR
Generative Process Planning
Milling phase distribution
Generic Process based Systems
Process Ascending Generation PAG
Neural Network
Iterative Evaluation directed Generation
Constraints based generation (OMEGA,PROPEL)
Multi Agents Systems

Figure 6: Milling Plans generation concept [6]

FMS focused towards the product variety and hence there was a need of alternative process plan generation. Flexible process plans were developed which were capable of producing alternative sequences and manufacturing resources. The generation of process plans has two main approaches [7] which are described below:

Variant Approach

In variant process planning, alternative cases are used to generate process plans. The variant approach groups and assigns codes to families of components that require similar manufacturing set-ups. All the parts already existing in the company are grouped according to their similarities in features, dimensions and tooling etc. So when the new part arrives, corresponding process plan is selected. This method is based on the previous experience and knowledge of the company. It lacks flexibility and also grouping of the parts (which have been manufactured and designed already in the company) is also required.

Generative Approach

Generative process planning as the name suggests, is not based on retrieving process plans from already existing ones but rather generates new process plans. So it uses the part information and design specifications to propose process plan and optimized machine structure. With these diverse approaches, three of the methods are discussed in section 1.11.3 [8] for generating process plans.

Expert System Generation

The first expert system was developed in 1960s. The expert system generally consists of two main elements. Set of rules and a set of tasks. Set of rules consist of “if and then” conditions and require knowledge of user. Set of tasks consist of operations performed by the system. It consists of some variables which are required as input from the user.

Selection by Constraint Satisfaction

Constraints are defined by variables. Objective is achieved by narrowing the domain of solution space by satisfying constraints. As important part of this method is mathematical modeling by which objective function and constraints are defined. This method is useful for combinatorial problems involving more than one objective function.

Reconfigurable Process Plans

As discussed in section 1.4, FMS dealt with the part variety and focused on machine structure. Reconfigurable process plans were developed to deal with change in product and flexibility was introduced when needed. In case of existing machines, RPP propose that part family nearest to the new part is selected and then process plan for the new part is retrieved from it. RPP used two criteria. First, part handling and tool handling activities which do not add any value were minimized. Second, cost of changes was added. A new part may require new machine which in turn would cause change in process plan. RPP supports decision makers in making machine selection and sequencing activities at initial stages of manufacturing, thus promoting concurrent engineering.

There has been extensive research on process plan generation in reconfigurable manufacturing systems. Problems of Multi objective process plans in reconfigurable systems are a challenge for today’s manufacturers and designers. A wide variety of process planning issues exists in literature. Due to the complexity involved in analyzing process planning and related issues, researchers tend to focus their studies on one of the aspect for RPP. For example they consider the process selection problem as in [9], process sequencing [10] or minimum machine structure required [11]. These proposed techniques generated all possible process plans for a certain part family. Certain considerations needed to be taken into account to generate optimized process plan.

1.5 Optimization

The desire of optimality is inherent by humans. The search for extremes has been the goal of human race since the start of the world. Theory of optimization was developed in sixties after the computers became available. The goal of the theory is the creation of reliable methods to catch the extremes (maxima or minima) of a function by an intelligent arrangement of its evaluations (measurements). This theory is vitally important for modern engineering and planning that incorporate optimization at every step of the complicated decision making process.

Optimization is central to any problem involving decision making whether in engineering or any other field of life. The task is to find the best among alternatives. The measure of goodness is described by an objective function developed by extensive mathematical modelling. In general optimization is the selection of best among some alternatives. In mathematics, optimization problem consists of maximizing or minimizing a function from a set of domain.

The area of optimization has received enormous attention in recent years, primarily because of the rapid development in computer technology, advanced manufacturing methods and products. Many software packages now come with a special optimization tool like Optimization toolbox of MATLAB. When these techniques are used to solve the design problems in engineering then this becomes engineering optimization or design optimization. The aim is to fill the gap between theory of optimization and engineering practices. Since most of the engineering problems are NP hard problems which cannot be solved by conventional techniques that lead to the promotion of developing advance techniques and search methods in optimization.

Based upon the objective function and the domain, optimization has been subdivided into many major areas like multi objective optimization, multi modal optimization, convex optimization, combinatorial optimization etc.

Since some of the problems are linear having constrained or unconstrained domains, so linear programming was developed for this class of optimization. Likewise some objective functions have a set of domain from which each input is verified for the criteria of optimization. We can say that there is a limited solution space where optimal point has to be found. But in some cases there is no solution space so bench marking is done according to some specific criteria obtained by mathematical modeling. So a lot of search techniques have been developed for different objectives and engineering optimization is a way to fill the gap between the engineering practice, real life problems and the developed optimization techniques.

1.5.1 Advance Optimization & Manufacturing

Manufacturing problems are considered as complex combinatorial problems involving non linear constraints and large solution search space and are considered to be NP Hard problems. As discussed, conventional approaches cannot provide answer to these problems due to their limitations. Search techniques play an important role in this regards. Oduguwa et al [12] summarized the features of real life industrial problems which make them almost impossible to solve by traditional techniques.

The past decades have seen important advances in the use of search techniques and more importantly evolutionary algorithms to address challenging optimization problems in manufacturing.

Table 3: Features of real-life optimization problems [12]

Classification Schemes	**Features**
Number of parameters	Highly dimensional
Existence of constraints	Constrained
Number of objective functions	Multi-objective
Nature of objective functions	Hybrid
Nature of objective functions	Inseparable
Dependence among variables	Independent and dependent variable
Nature of quantitative search space	Unknown search space, Multi-modal
Nature of equations involved	Linear, non-linear, quadratic
Nature of design variables	Static and dynamic
Permissible values of design variables	Hybrid
Expert knowledge	Imprecise and stochastic
Nature of qualitative search space	Discontinuous and highly multi-modal

1.6 Problem Definition

In current manufacturing systems which require responsiveness, the alternative processing routes are key issues since they provide sequencing and processing flexibility which can be used for reconfiguration issues [13]. When alternative process plans are available, due to the numerous possible combinations of alternative process plans, the goal in computing a solution methodology is to answer the question: which of the process plan will give the optimal results for the given production scenario? Solving such a problem requires a comprehensive analysis of interrelated decision making activities that aim at selecting an optimal manufacturing process plan. This becomes an NP Hard problem [12].

This kind of problem can't be defined by a limited search space involving simple constraints, thus making it difficult to solve by classical techniques. These problems can be solved by using evolutionary approach. By their use an optimized process plan can be found based on criteria which can be set based upon time and tolerance constraints. Methodology has been proposed to add reconfigurability in the proposed process plan so that it can be made around a part family rather than a specific product and to be made modular for future uncertainties. So the problem definition is "to find optimized reconfigurable process plan based on set criteria".

1.6.1 Problem Description

The complete problem description has been divided into three parts by keeping in mind the assumptions that single spindle direction is considered hence parallel machining case option is not considered as well as there is single machine option. The main parts of the problem are as follows:

a. Process plan generation
b. Modeling
c. Application of Evolutionary Approach

1.6.2 Construction of the Thesis

The thesis report has been divided into seven chapters. Chapter 1 provides a comprehensive introduction to this research, description and justification of the research problem and the conceptual approach along with motivation to handle the research problem. Chapter 2 mainly focuses on different techniques of process planning as well as search techniques in optimization and provides a detailed literature review. In chapter 3 and 4, main methodology for generating the process plans and application of evolutionary algorithm have been discussed respectively. Chapter 5 and 6 discuss comparison and analysis of the proposed methodology and also provide directions for future research in this area.

1.7 Summary

In this chapter, we have briefly compared different manufacturing systems and the need of advanced reconfigurable systems. The characteristics of these advanced systems were described in detail along with the recent developments. Along with that, different approaches of process planning along with the need of reconfigurability and optimization in this area were discussed.

CHAPTER 02: Literature Review

2.1 Process Planning

Process planning can be defined as a group of work instructions defined in a sequence to make a product. There can be a single part or assembly of parts in a sequence to synthesize the final product. Due to its importance in industrial as well as manufacturing systems engineering, researchers have proposed many techniques over the years to generate process plans.

ElMaraghy [14] divided the approaches for the generation of process plans into three levels according to their level of granularity. These were described as follows:

1. Multi-domain Process Planning Level: This was based on the initial decisions which have to be taken before machining or assembling the product such as method of assembly, material selection etc.

2. Macro Process Planning Level: It was concerned with selecting the sequence for generating process plan so as to create a setup. In FMS and RMS, this process has to be repeated frequently due to the changes in product design unlike in DMS where the setup is fixed.

3. Micro Process Planning Level: This level is concerned with the exploration of optimal sequence/process plan based upon details and criteria defined by the manufacturer.

Dusan [10] proposed a technique of generating alternative process plans for integrated manufacturing environment in order to get a cost feedback each time a modification was made in the sequence by the designer. The methodology consisted of three main parts; first was the selection of machining processes which can be adopted for generating machining feature specified in the part, second was grouping and sequencing of those processes and third was the generation of a process plan network.

Prabhu et al [15] proposed process plan generation for rotational components. It was presented in the form of a tree where different branches represented alternative process plans for a part. But there was inadequacy of several important issues like generation of process plan for prismatic parts which have complex precedence constraints.

Vamsi et al [16] used feature based modeling for generating automated process plan. The methodology involved feature recognition for rotational components and then using knowledge based system for generating process plan. It consisted of three modules which were feature recognition, feature storage algorithm and process plan generation module. But these models were developed only for simple rotational components and could not handle complex precedence constraints. Similarly Yang et al [17] also proposed a feature recognition technique to generate alternative process plan but it was not limited to rotational components. Rule based knowledge was used to generate process plan taking into account the limitations of the available machine tool.

Ray [18] pointed out that the process plans should be integrated to make process planning useful in a production system. He proposed integrated process planning approach in which two modules were integrated. One module was for the generated process plans while the second managed the flow lines and resource allocation.

The development of process plans along with their kinematic configuration led to the concept of coordinate evolution or co evolution. As discussed, the traditional approaches were focused on developing the process plans on the machine structure which was fixed. But the rapid changes in products design and competition aroused the need of focusing on process plan along with their machine kinematic configuration simultaneously.

In this new line of research and RMS, product and process selection cannot be limited and treated as a simple optimization problem. There is a need for a dynamic model that can accommodate uncertainties where success depends on continuous improvement and quality management and internal environment reflects selection pressures from outside [1].

Tolio [19] presented the concept of co-evolution. The co-evolution paradigm suggests that changes in product, processes and production systems have a direct impact on each other. Therefore, changing one of them causes the change in other two. The author reviewed most of the research work done till 2010 on co evolution and placed it on evolution axis. He discussed in detail that the research which is carried out focus on machine structure or process based upon the objective functions. He used triangular cross section as a representation of the level of integration between the products, processes and production systems for a given configuration. He assumed that transformation (i.e. part production) can only be carried out if and only if all the three parameters (PS3) of co evolution paradigm were designed. He proposed different examples of configuration by which level of integration for a particular configuration can be measured.

The concept of co evolution or generation emphasizes on focusing on machine structure and process sequence at the same time. Few recent approaches have been discussed below in detail which will help us to develop more understanding about the concept of co-generation.

2.1.1 Process Plan Generation Approach

As discussed in section 1.4, due to the need for co evolution, various methodologies have been proposed in the recent literature to sort out this problem. The first approach which is discussed here generated multiple process plans for a given part along with their kinematic machine structure. This approach was presented by A. Baqai [8].

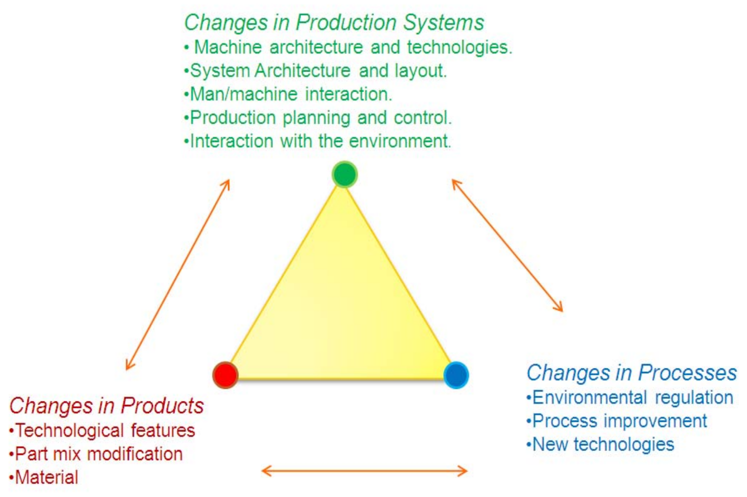

Changes in Production Systems
• Machine architecture and technologies.
•System Architecture and layout.
•Man/machine interaction.
•Production planning and control.
•Interaction with the environment.
Changes in Processes
•Environmental regulation
•Process improvement
•New technologies
Changes in Products
•Technological features
•Part mix modification
•Material

Figure 7:Co-Evolution Paradigm [19]

The main steps of the approach [8] are mentioned below:

Step 0: Initialize the first process plan

Step 1: Generation of process plans along with their associated kinematic configuration for machine structure. Operations having first priority are given zero precedence.

Step 2: To generate ensuing process plans from step 1 by ascribing operations to it which can be performed on the same machine structure.

Step 3: Generation of ensuing process plans after step 1 & 2 by ascribing operations that can be performed on parallel machine structures by subsequent operation of spindles.

Step 4: Generations of ensuing process plans after step 1, 2 & 3 by allocating operations that can be performed after a tool spindle change.

Step 5: Generations of subsequent process plans of step 1, 2, 3 & 4 by allocating operations that can be performed after part rotation.

Step 6: Find any other possible option for process plan.

Description

The **inputs** to the approach are topological interactions which tell about the relation between the operations which have to be performed on the part and precedence matrix which identifies operations that need to be performed before or after any other operation based on topological interaction. After that, precedence ranking of the operations for the part is developed.

Along with these inputs, relationships between the operations are also required. The relationships required are as folloes:

a. Operations having similar axis
b. Operations having similar spindle direction
c. Operations having similar operation type

This knowledge will provide the information that which operations can be performed in parallel, with the same tool and which can be performed after rotating a part. For example if operation 1 and 2 are of same operation type, then they can be performed with the same tool. Due to the primary focus on process plans, the focus shifts from the main concept of co-evolution. On the other hand, because the approach considers the machine configuration as well as the process plans, this approach after improvement can be brought closer to the concept of co-evolution.

2.1.2 Minimum Machine Structure Approach

The second approach under consideration here is the machine structure configuration approach presented by A.I. Shabaka and H.A. ElMaraghy [11]. The flow chart of the approach is shown in figure 8.

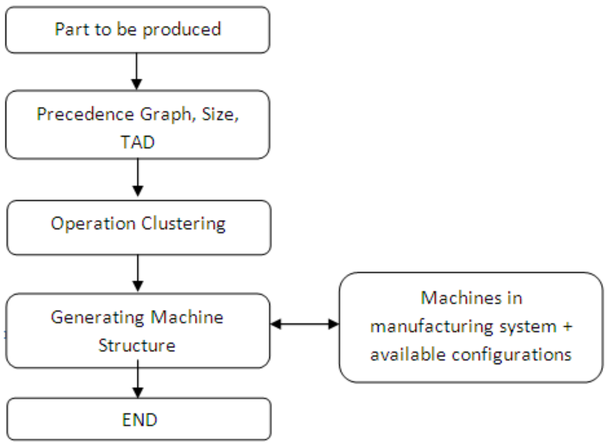

Part to be produced
Precedence Graph, Size, TAD
Operation Clustering
Generating Machine Structure
Machines in manufacturing system + available configurations
END

Figure 8: Machine Structure Configuration Approach [11]

Description

The inputs as shown in the flow chart are

Precedence Graph
Tool Approach Directions
Dimension of Part

Precedence Graph captures the precedence constraints, which define order of succession among operations. In previous approach [8] precedence matrix gave the information about the sequencing of operations. Grouping or clustering of machining features is based on constraints that can be of different types. Elmaraghy in his machine configuration approach [11] mentioned three types of such constraints which were logical, tolerance and datum. Logical constraints are between those operations which are logically related to one another. For example if milling and surface finish has to be performed on same plane of part, then milling must be performed before finishing. Tolerance Datum constraints as the name suggests are between two operations when a group of operations must be performed on the same machine and with the same set-up positions to preserve tolerances with respect to position and the relative positioning of the operated features. Finally other constraints refer to those constraints which exist between operations but are neither tolerance datum nor logical. Graph is used for the clustering of operations.

Tool Approach Directions (TAD) represents all the possible directions by which a tool approaches to build the required feature. Orthogonal dimensions are considered giving the possibility of six directions in which tool can be directed towards the part. Based on the information of TAD and operations clusters, minimum machine configuration is developed for producing the given part.

Comparison of minimum machine structure and kinematic approach

In the first approach, formation of the process plans was mainly discussed and along with that machine configurations are developed for each possible process plans. But in the second approach, focus was towards generating minimum machine structure. Since machine structure is supposed to be simpler, so the possibility of performing parallel operations is ruled out in the second approach. So first approach considers multiple platforms but second does not.

First approach has been developed by considering objective function of minimizing the time needed to manufacture the part but Elmaraghy [11] has considered objective function of minimizing the initial cost. That is why he has emphasized on minimizing the machine structure. Initial cost for the first approach will be high.

2.1.3 Improvement Approach

Maaz et al [20] in his thesis suggested new algorithm by combining the approaches developed by A.Baqai [8] and Elmaraghy [11] on cogeneration. This thesis [20] was focused on improving the previous approaches and to address certain issues regarding co-evolution.

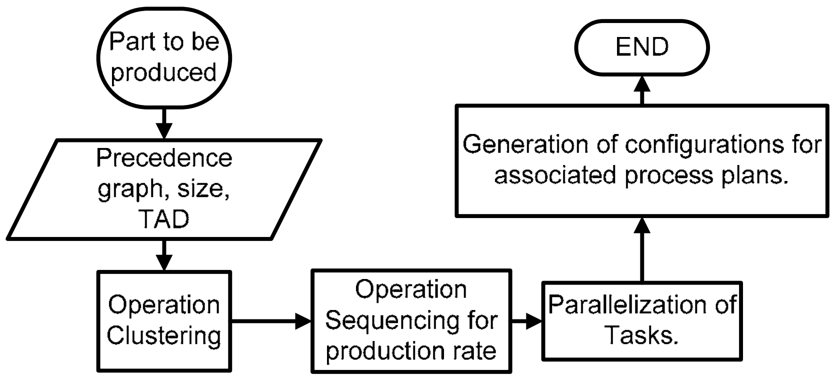

Part to be produced
Precedence graph, size, TAD
Operation Clustering
Operation Sequencing for production rate
Parallelization of Tasks.
Generation of configurations for associated process plans.
END

Figure 9:Co-Generation & Minimum Kinematic Configuration Approach [20]

Description

The algorithm developed for the improved approach [20] is same as that of minimum machine structure approach from the start and utilizes the same inputs, but after the operation clustering and defining minimum TAD for each and every cluster, it move towards the process sequencing and considers possibility of performing operations in parallel. In the end, generation of machine structures with the associated process plans takes place. Hence objective function of minimizing the initial cost and time is combined.

The above discussion helped us in understanding the concept of reconfigurable process plans and cogeneration. From the approaches presented in sections 2.1.1-2.1.3, a number of process plans for the given part along with their machine structures can be obtained. To determine the optimized process plan among them is an NP Hard problem. In later chapters, optimization techniques will be discussed which can be used to address issues that are critical in obtaining optimized process plan in RMS.

In order to get competitive advantage, need of optimality aroused after getting alternative process plans. Further we will discuss the search areas in optimization and techniques which can be used to address issues that are critical in obtaining optimized process plans in RMS.

2.2 Optimization Techniques

Since from the first day, man has always urged for improvement and for optimization. So various search techniques have been developed in order to attain a goal; that is optimization. These techniques can be divided into three main types starting from classical techniques and moving towards modern age optimization techniques. The details of these search areas are further explained in detail. Fig [11] gives an overview of these search areas of optimization.

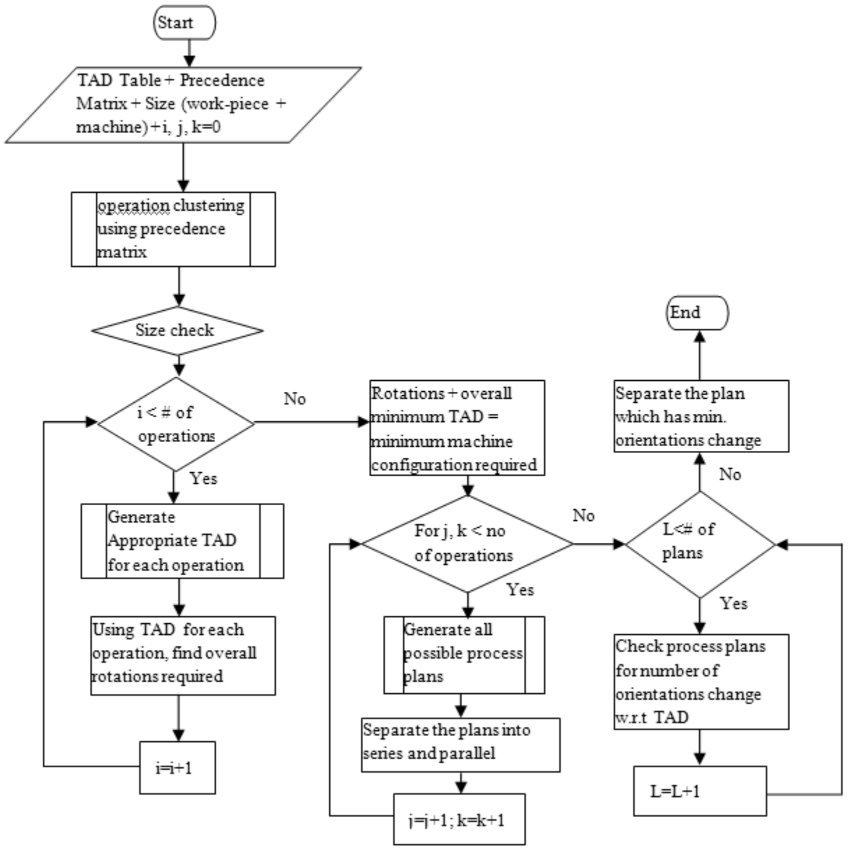

Start
TAD Table + Precedence Matrix + Size (work-piece + machine) + i, j, k=0
operation clustering using precedence matrix
Size check
i < # of operations
No
Yes
Generate Appropriate TAD for each operation
Using TAD for each operation, find overall rotations required
i=i+1
Rotations + overall minimum TAD = minimum machine configuration required
For j, k < no of operations
No
Yes
Generate all possible process plans
Separate the plans into series and parallel
j=j+1; k=k+1
L<# of plans
No
Yes
Check process plans for number of orientations change w.r.t TAD
L=L+1
Separate the plan which has min. orientations change
End

Figure 10: Improved Approach for Co-Generated Process Plan [20]

2.2.1 Calculus Based Techniques

The calculus based methods have been intensely studied and are subdivided in two main classes

a. The direct search methods find a local maximum moving on a function over the relative local gradient directions.

b. The indirect methods usually find the local ends solving a set of non-linear equations, resultant of equaling the gradient from the object function to zero, i.e., by means of multidimensional generalization of the notion of the function's extreme points from elementary calculus give a smooth function without restrictions to find a possible maximum which is to be restricted to those points whose slope is zero in all directions. They involve methods such as gradient based and region elimination methods. Some of these techniques are discussed below in detail.

Gradient based technique and Langragian technique are one of the classical techniques that have been developed. They are easy to understand and apply. But they can be applied to unconstrained functions and unimodal mostly. Also they become difficult when the problem is multidimensional. Since they do not search the solution space efficiently, there were chances that solution might stuck in a local optima.

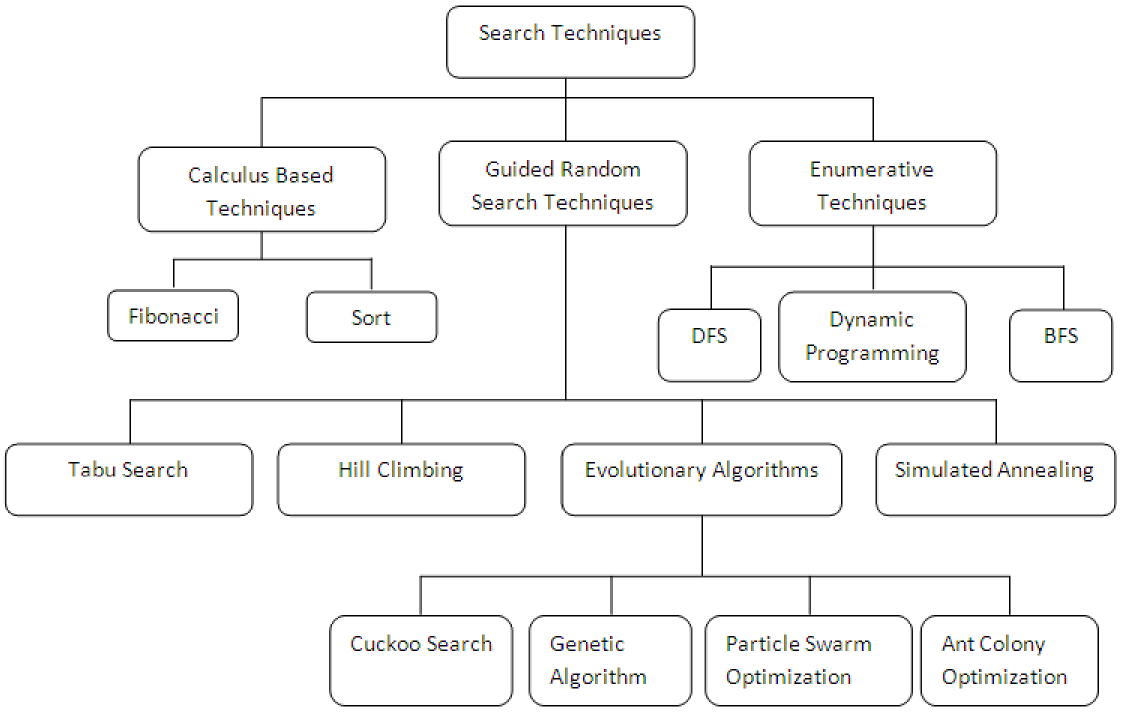


Figure 11: Optimization Search Techniques

For these one directional search methods like fibnocci, golden section search, secant or other region elimination methods, which solve the problem of optima in a closed interval with unimodal condition, developments were done to increase the accuracy of desired optima and to get more accurate answer with fewer calculations but still the conditions could not be changed and these classical gradient based techniques remained for one dimensional unimodal objective functions.

To solve this problem, **convex programming** was suggested in combination with KKT methods that ensure that the solution found is global optima and search space was efficiently searched. But the unconstrained and multidimensional problem was still there.

Gradient base techniques were although used for one directional problems but they played an important part in the development of multi directional problems solving techniques where they were used as line search. To find solution for a multidimensional problem, **conjugate direction** method was developed. In these methods conjugate directions were searched to find optima in the given space. But these types of methods had a restriction on objective function. They cannot be applied if the given objective function is singular or not positive or negative definite. Further development on these methods was conjugate successive search method.

Simplex method was developed for constrained optimization problems. These methods were based on linear programming and were under the class of convex optimization. They were used to solve for the optima under the linear constraints. They had a quality of not getting stuck in local optima. However they came with worst case exponential complexity. This means that by the increase of variables, the computation cost as well as total time was increased thus slowing down the whole procedure. Also the constraints when linear and explicit, were solved easily, but implicit constraints were a lot difficult to handle. Non linear constraints were neglected. In many cases when there were too many constraints, then the solution became infeasible.

So after having a detailed discussion on them we can say that they have two main short comings

a. They depend on the availability of derivative.
b. They have a local focus since they seek the optima in localized neighborhood so they may give local optima.

Limitations on Calculus based Techniques

As discussed, it was necessary to develop new techniques apart from these classical methods so as to deal with the real life practical and NP Hard problems, which can be a solution for industries. Problems associated with the use of conventional methods leading us to development of these random search techniques can be summarized as follows:

a. Provide solutions to unimodal problems
b. Convexity checks
c. Regularity checks
d. Implicit and coupled constraints involved
e. Cannot be used for NP Hard problems
f. In efficient search of solution space
g. Mathematical problems in defining real world problem with complete constraints
h. Unsuitable for unconstrained problems (stuck in local optima)

2.2.2 Enumerative Techniques

As explained in detail short comings of calculus based techniques, further development was based to solve combinatorial optimization problems. Combinatorial problems are the one which were more close to real world problems involving exhaustive search, non linear constraints, complex objective functions and many variables. They were developed for the problems having usually irregular structure.

Enumerative techniques as name indicates are the one in which all the points feasible for optima are searched one by one for checking the optimality criteria. They search every possible point of objective functions domain stepwise. These are simple to implement but may require a lot of computation. They were mainly divided into three main techniques which are described in detail.

Depth First Search (DFS) Technique

DFS is a technique of searching the solution space one by one in form of tree. One starts from the root and goes one by one to the other nodes. It is a uniformed search method. Depth-first search is an example of a directed search use to find a sequence of operations that will move a problem solver from the initial state to a goal state of a problem. It is an approach that makes the search through the problem space more efficient than simpler method. It means that DFS goes from a root to a leaf exploring all the nodes instead of going to another path. First it explores that path to the ultimate node and then starts from the root again for other path. Sometimes any node at the end or any leaf can be chosen but sometimes node that is nearest to the root or having shortest path is selected. This is known as minimum depth.

Breadth First Search (BFS) Technique

BFS method explores the root nearest to current node, before exploring nodes further away. So the roots are not searched in order. They are searched level by level. When a breadth-first search succeeds, it finds a minimum-depth (nearest the root) goal node.When a depth-first search succeeds, the found goal node is not necessarily minimum depth.

Limitations of DFS & BFS

For a large tree, breadth-first search memory requirements may be excessive. For a large tree, a depth-first search may take an excessively long time to find even a very nearby goal node. They are purely exhaustive search methods and will fail to find any solution but for smaller instances that can be found in practical length of time. As discussed, DFS and BFS are exhaustive search techniques so they can be applied to simple problems which have a small solution space but in industrial world, where combinatorial objective functions are involved and benchmark has to be achieved, DFS and BFS fail to provide a solution due to their computational limitations.

Dynamic programming

Dynamic programming is a stage-wise search method suitable for optimization problems whose solutions may be viewed as the result of a sequence of decisions [21]. The most attractive property of this strategy is that during the search for a solution it avoids full enumeration by pruning early partial decision solutions that cannot possibly lead to optimal solution. In many practical situations, this strategy hits the optimal solution in a polynomial number of decision steps. However, in the worst case, such a strategy may end up performing full enumeration.

Dynamic programming takes advantage of the duplication and it arranges to solve each subproblem only once, saving the solution (in table or in a globally accessible place) for later use. The underlying idea of dynamic programming is: avoid calculating the same stuff twice, usually by keeping a table of known results of subproblems.

Limitations of Dynamic Programming

Dynamic programming has been applied to various manufacturing problems which involve production scheduling. Approximate dynamic programming promises to provide results within 1% to 2% of global optimum [21]. Dynamic programming has some limitations since it involves enumeration. For complex combinatorial problems, in which one decision effects the objective function of other such as labor cost and completion time of batch of product. If labor cost is reduced by reducing labor, then it will have an effect on completion time for a batch to be manufactured. Since in dynamic programming, the decisions are considered to be independent, so it can’t be employed there.

All these enumerative techniques involve a lot of computation and become more and more difficult to apply practically in a situation involving more variables and a large search space.

Guided Random Search Techniques

As discussed optimization problems are ubiquitous in our everyday life and may come in a variety of forms. The real world has many discontinuities and noises which is not surprising due to the innovations and complex constraints involved. Guided random search techniques can be sub divided in further categories which are further discussed below in detail.

Tabu Search

Tabu search is a metaheuristic local search algorithm which can be used for combinatorial optimization problems. Local search start from considering a solution to be optimum and checking its immediate neighborhood for improvement. They perform it by having a memory structure (which can be short, intermediate or long term) that describes the visited solution but they have a tendency to stuck into local optima.

Tabu search uses the exclusive memory function to reach to final solution. The basic algorithm of Tabu search by Pham et al [22] is described in figure 12.

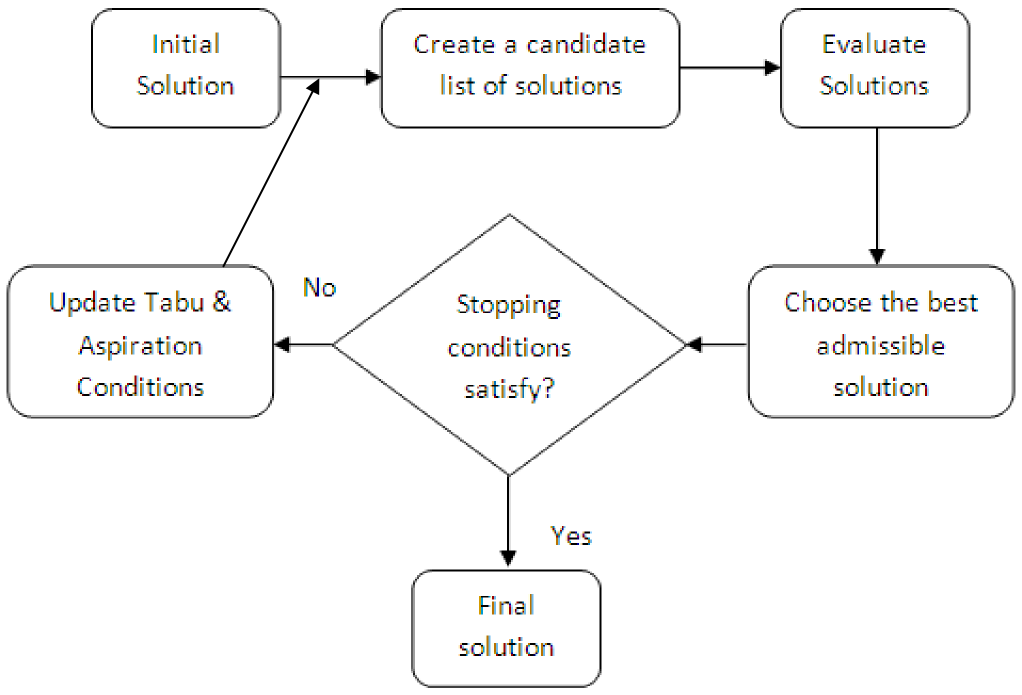

Initial Solution
Create a candidate list of solutions
Evaluate Solutions
Choose the best admissible solution
Stopping conditions satisfy?
No
Update Tabu & Aspiration Conditions
Yes
Final solution

Figure 12: Flow Chart for Tabu Search [22]

Limitations of Tabu Search

It allows non-improving solution to be accepted in order to escape from a local optimum and uses Tabu list for memory function to avoid cycling. It can be applied to both discrete and continuous solution spaces.But there are too many parameters to be determined and number of iterations could be very large.

By the use of systematic memory function, the tabu search avoids cycling. It can be used in manufacturing where complex optimization problems involving non linear constraints are not involved since they tend to make solution space bigger which makes the tabu list and in turn memory function too large to handle.

Simulated Annealing (SA)

Simulated annealing is a method based on the heat treatment process of annealing. It is a method used of finding a good approximation of global optima in a large search space. It is usually more effective than enumerative methods provided if the objective is to find a good approximation of optima rather than to have a exact solution.

F. Busetti [13] gave an overview on simulated annealing. He explained SA with the help of a bouncing ball which can jump over higher valleys depending upon temperature. It can be proved that by controlling temperature, SA can provide global optima.

Limitations of SA

SA can deal with high non linear models. It's flexible and can reach to global optima. It is robust and has general technique. It is said to be very versatile. However there is a clear tradeoff between the quality of the solutions and the time required to compute them. The precision of the numbers used in implementation is of SA can have a significant effect upon the quality of the outcome. Both SA and GAs, by comparison, start with an initial random population, and allocate increasing trials to regions of the search space found to have high fitness. This is a disadvantage if the maximum is in a small region, surrounded on all sides by regions of low fitness.

Hill Climbing

Hill climbing is again a local search method that moves towards the solution incrementally by having one iteration at a time. If the new solution is better than before, then an increment is made to the solution until no further improvement is possible or a stopping criterion is met. It is again very good to find local optima in short time but does not guarantees global optima.

Limitations of Hill Climbing

Tabu search and simulated annealing give better results than hill climbing and are more advanced but hill climbing has to be applied for those applications where the time is limited such as real time problems. This makes necessary to develop new techniques that does not get stuck to local optima and can be applied to real time problems as well.

Random search and gradient search may be combined to give an iterated hill climbing search. Once one peak has been located, the hill climb is started again, but with another, randomly chosen, starting point. This technique has the advantage of simplicity, and can perform well if the function does not have too many local maxima.

2.2.3 Evolutionary Algorithms (EA)

EA is a class of evolutionary computation which uses mechanisms inspired by evolution: reproduction, mutation, combination, selection. The common underlying idea behind all these algorithms is the same, given a population of individuals, environmental pressure causes natural selection and this leads to rise in fitness of population among which survival of the fittest occurs. Based upon a fitness function selected that can be any according to objective function, best individuals are selected and a new generation is created after assigning some random variations which differ for different evolutionary algorithms. So they use random variations to get new generation of solutions. Evolutionary algorithms have two main distinguish features named as intensification and diversification.

Intensification enables them to find the best while **diversification** compels them to find new search spaces so that it can be done efficiently and the solution does not limits itself in local optima. So the power of all evolutionary algorithms is that they intimate the best in nature especially the biological systems evolved in millions of years. So the forces that form the basis of these algorithms can be summarized into two main categories as:

a. Variation operators that create the necessary diversity
b. Selection acts as a force pushing quality

There have been many developments on evolutionary algorithms. Some of them are discussed further in detail.

Due to the limitations discussed above, researchers have explored evolutionary algorithms in a past few years for engineering problems. Venter [23] reviewed optimization techniques for non-linear constrained optimization problems. The designer should be aware that no such algorithm exists that will solve all optimization problems. He classified the optimization algorithms as classical gradient based methods and evolutionary algorithms and proposed that for problems where local minima is not an issue and analyses is computationally expensive, local search techniques or classical gradient should be used for optimization. Problems where gradient is not present and global optimum has to be found, one should use global techniques such as evolutionary algorithms.

Odugva et al [12] summarized the application of evolutionary computing in manufacturing industry. He concluded that unlike classical techniques, which usually gave sub-optimal solution, evolutionary algorithms have a better approach in solving industrial problems which are more complex and random. They gave detailed overview showing application of these algorithms in paper, chemical and metal forming industry.

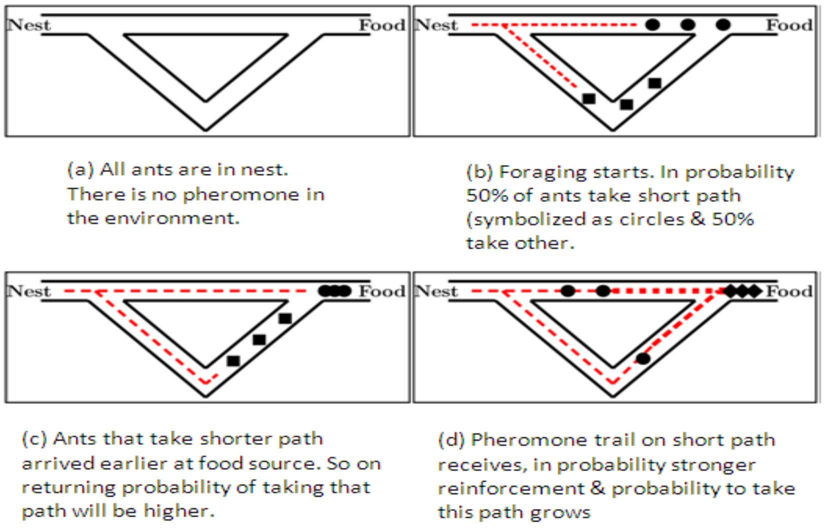
Since 1980s, trend of using evolutionary techniques in practical applications has been increased. For routing and scheduling problems, ACO has been applied widely. Shahram and Iraj [24] proposed a method to solve cell formation problem in cellular manufacturing by ACO. They solved it with the help of building a pherone matrix consisting of cells, machines and parts. This pherone matrix was used to develop next random solutions. The proposed algorithm resulted in 5.73% improvement of cellular movements.

Yasuhiro Yamada et al [25] used particle swarm optimization for solving layout problem and for optimized resource allocation for a manufacturing system. 3D graphic simulator was used for clear demonstration and the effectiveness was validated through several manufacturing examples. Since this research focuses on evolutionary techniques of optimization, brief detail of these techniques has been discussed in this section below.

Ant Colony Optimization (ACO)

Ant Colony Optimization is a probabilistic technique based upon the nature of ant travel. Ants wonder in search of food and upon finding food they return to their colony laying pheromone trail behind them so that other ants can follow their path. More of the ants travel through a path, that means that more the path is suitable. This feature lay basis of optimization.

ACO was first established in 1990s. It was first checked for travelling sales man problem. With the advent of time, it was used by manufacturers for routing and scheduling problem. Christian Blum [26] reviewed ACO and mentioned the recent trends in which ACO is used along with GA and constraint programming to produce hybrid and more effective ACO. The main idea of hybridization is to reduce the solution search space.



(a) All ants are in nest. There is no pheromone in the environment.

(b) Foraging starts. In probability 50% of ants take short path (symbolized as circles & 50% take other.

(c) Ants that take shorter path arrived earlier at food source. So on returning probability of taking that path will be higher.

(d) Pheromone trail on short path receives, in probability stronger reinforcement & probability to take this path grows

Figure 13: Description of ACO exhibiting behavior of ants [26]

Particle swarm optimization (PSO)

Particle swarm optimization (PSO) is based on the natural process in which the flock of bird travels along a path in search of food. There has been a lot of research in PSO and many further techniques like post particle swarm optimization have been developed to further refine PSO. Most of the changes have been made which adjust the velocity of the bird.

In PSO, each particle is considered as a bird which is placed in a solution search space and each calculates objective function at its current location [27]. Each bird then determines its next location based on its current location, its previous best position and the best position of the bird from the whole flock or set of particles. Each individual is composed of three dimensions, the current position, previous best positions and the velocity. Eventually, the whole flock is likely to move to reach the location of food i.e. global optimum position.

In an article of Dr. Umarani [27], he has discussed applications of PSO. PSO finds its applications in data mining, controls and combinatorial optimization problems. NASA is working on planetary mapping through swarm intelligence. Research has been proposed for its use in medicine by controlling nanobots in the body for killing cancer tumors. Mostly it is being applied to solve unconstrained problems.

PSO can be applied to multi objective problems but there are still many stochastic parameters involved in it. It is fact converging which may lead one to local minima. For this as previously stated, that velocity factor is being controlled through various evolved PSO algorithms.

Genetic algorithms (GA)

Genetic algorithms (GA) are the most popular class of EA which are based on Darwin's theory of survival of the fittest and were first developed in 1970's by Fraser and Bernall. A generation of solution is produced from the previous generation of solution known as parent generation and based on objective function; the fitter ones survive thus providing better solution. A brief description of evolutionary process would help in better understanding the concept of GA.

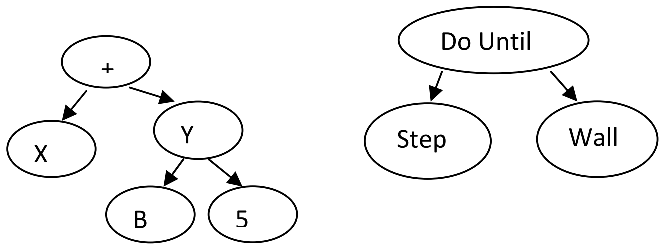
In nature all living organisms basically consist of cells. Every cell consists of a set of chromosomes. Each chromosome in turn is a string of DNA and serves as a model for the whole organism. A chromosome is basically a collection of genes, each one representing a trait. Each string has a specific position in a chromosome. Parents combine by crossover and mutation operators to produce new chromosomes. In crossover, recombination takes place. During mutation basically a small change is incorporated in the elements of DNA. Research indicates that errors in copying genes from parents result in these changes.

John Holland in 1970 first proposed the methodology of GA in detail and gave the basic framework for it. Initially a population that consisted of a set of solution was generated and then evaluated on the basis of some fitness value. Fitness value was assigned according to the objective function. After selecting the fitter solutions from the initial population, they were termed as parents who were used to generate new population.

The first step in GA was generating random solutions and encoding i.e. the representation of problem solution/chromosome. Encoding technique varies with the problem. Sankar and Ranjendran [1] applied GA on same scheduling problem by using both phenotype and binary encoding and concluded that there isn't any difference in both in terms of computational time and ease of programming but binary encoding gives better results as compared to phenotype encoding. Some of the different encoding techniques are given in table below.

After encoding and generation of initial population, next step is the selection of fitter chromosomes for the generation of next population. Selection operator deserves a special position in Genetic algorithm since it is the one which mainly determines the evolutionary search spaces. It is used to improve the chances of the survival of the fittest individuals. There are many traditional selection mechanisms used and many user specified selection mechanisms specific to the problem definition. Various methods for example roulette wheel selection, rank selection, tournament selection and elitism etc have been proposed for this purpose. A detail on selection methods can be found in [28] by Sivaraj in which the author discussed different parameters involved in choosing the selection method.

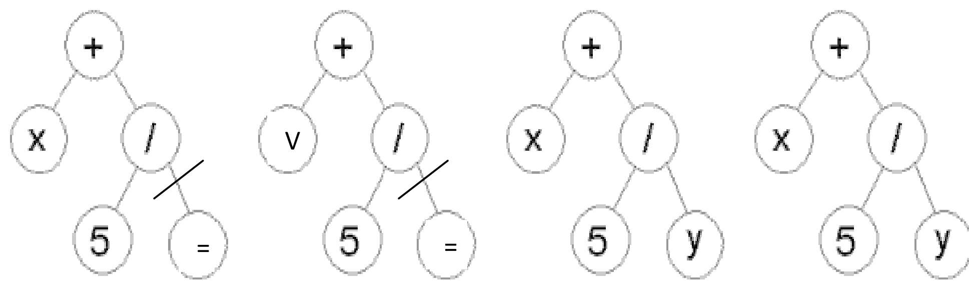
Table 4: Encoding Schemes used in GA [29]

S/No	Encoding Technique	Example
1	Binary Encoding	Chromosome 1: 1100101 Chromosome 2: 0111001
2	Tree Encoding	Chromosome A Chromosome B 
3	Permutation Encoding	Chromosome 1: 143652 Chromosome 2: 645132
4	Value Encoding	Chromosome 1: 5.345 1.342 4.564 3.786 Chromosome 2: DHEFIGTSR Chromosome 3: (right), (back), (left), (up), (forward)

After selection, recombination occurs between parents known as crossover. Crossover is a genetic operator used to vary the programming of a chromosome or chromosomes from one generation to the next. It is analogous to reproduction and biological crossover, upon which genetic algorithms are based. Cross over is a process of taking more than one parent solutions and producing a child solution from them. There are methods for selection of the chromosomes. Crossover can be of many types such as single point crossover, two point crossover or uniform crossover etc. It depends upon the encoding techniques used for representing solution.

Table 5: Different techniques for crossover [29]

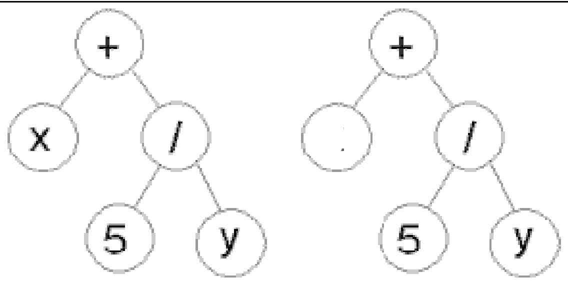
S/No	Encoding Techniques	Crossover Techniques	Example
1	Binary	Single point crossover	111/100 + 001/110 = 111/110 + 001/100
		Uniform crossover	111111 + 000000 = 101010 + 010101
		Double point crossover	11/01/00 + 10/11/11 = 11/11/00 + 10/01/11
		Arithmetic crossover	001010 + 100101 = 110010 + 111101 (First three digits are added for 1st child and the last three digits are from the parent on Similarly same method is adopted for 2nd child.
2	Permutation encoding	Single point swap	1234/5678 + 3142/7586 = 1234/7586 + 3142/5678 (it can also be double point crossover like in binary encoding)

3	Tree Encoding	Swapping	 + x / 5 = + v / 5 = + x / 5 y + x / 5 y

Once the crossover takes place, next step is of mutation. Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of algorithm chromosomes to the next. It is analogous to biological mutation. Mutation alters one or more gene values in a chromosome from its initial state. In mutation, the solution may change entirely from the previous solution. Hence GA can come to better solution by using mutation. Mutation occurs during evolution according to a user-defined mutation probability. This probability should be set low. If it is set too high, the search will turn into a primitive random search.

Table 6: Mutation Techniques used in GA [29]

S/No	**Encoding Type**	**Mutation Technique**	**Example**
1	Permutation	Change of order	2341567 => 2**1**4**3**567 (single point swap) 23456718 => **32**4**8**671**5** (double point swap)
2	Binary	Bit Inversion	110011 => 1100**0**1 (single point) 110001 => **0**1**1**001 (double point)
3	Real Value	Addition of small number	1.33 1.45 1.11 => 1.33 **1.56** 1.11
4	Tree	Change of operator	

The decision of crossover and mutation rate is a very important one. Crossover rate actually determines the probability or chances of recombination between the parents. Higher will be the crossover rate, higher will be the possibility of recombination between the selected parents for next generation. Crossover is done in a hope of getting better children having better fitness value and in turn higher chances of survival for the next generation. Crossover performs the intensification function and thus searches the solution space for better chromosome/solution.

Likewise mutation rate determines chances of a chromosome being mutated or not. If we assign zero percentage of mutation then it means that there is going to be no mutation in any chromosome of a generation. Mutation performs the diversification function and tends to generate new solutions by exploring search space more efficiently. If we set mutation arte too high, then we are actually increasing the chances in a generation for chromosomes of being getting mutate and it may cause our solution to cross the boundaries of search space. M.K.A.Ariffin et al [30] applied fuzzy logics to determine the optimum rate of crossover and mutation for the routing of automated guided vehicles. The flowchart for basic genetic algorithm can be represented in figure 15. To understand concept of GA a simple example given by Eiben and Smith [31] is described in figure 14.

String no.	Initial population	x Value	Fitness $f(x) = x^2$	$Prob_i$	Expected count	Actual count
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4
Average			293	0.25	1.00	1
Max			576	0.49	1.97	2

Figure 14: An algebraic example exhibiting implementation of GA

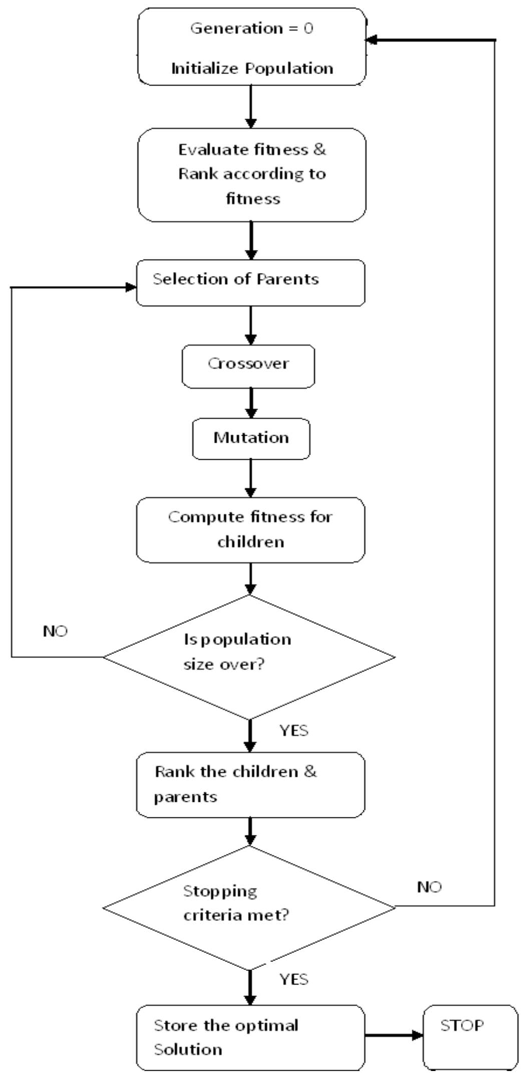


Figure 15: Flowchart for explaining basic steps for GA algorithm

GA developed in 1970s has also been implemented to various manufacturing problems. Paris and Pirreval [32] proposed that GA can be applied to deal with various design options in manufacturing. Tree encoding was used to represent different design options and sub-options. Simulation optimization method is applied to refine the design options. This method facilitates in selecting options since if we use only simulation method, then it involves many hit and trial processes. thus along with numerical optimization, design options can also be optimized using GA. Along with other sub fields of industry, scheduling problem was also optimized by this approach. In 2006, Omar and Baharum [33] proposed GA in solving constrained combinatorial problem of Job Shop Scheduling. Their intention was to reduce make span of the processes. Schedules were represented in the form of a matrix in which rows represented machines and columns represented order of the part. Although, GA takes a lot of time in solving this problem, but it can handle a variety of constraints along with objective functions and its results are predictable as well unlike branch and bound methods. Sankar et al [1] also solved the scheduling problem for a flexible manufacturing system.

In 2010, Y.Yang et al [34] developed a method for the dynamic facility planning by using GA. Since with the product changes, facilities have to be rearranged for reducing material handling costs. So the facility is supposed to be dynamic and is rearranged if the arranging cost is lesser than the reduction in material handling cost.

Apart from the manufacturing management problems of scheduling and layout planning, Nafis and Haque [35] proposed GA for the optimization of process planning parameters for rotational parts. Their purpose was to propose the optimal cutting depth, speed and feed of the machine keeping in mind machine specifications and maximum cutting force. They considered time as objective function and showed that total machining time can be improved by using optimal parameters.

Along with manufacturing systems and mass production industries, optimization using GA has also been applied in many other research areas. Cylvio and Desio [36] used GA to find optimal velocity in a stop and go cruise control system in which acceleration of the cruise has to be specifically controlled. Kazen et al [37] used GA to optimize point to point trajectory planning for a 3-linked robotic arm. Objective was to minimize travelling time in space. They concluded that GA can be used for multi-objective optimization problem in a non-linear constrained environment.

Due to the rapid growth in this area, hybrid GAs was proposed to get more efficient results. Ariffin et al [30] used fuzzy logics to control mutation and crossover rate in GA and applied it on scheduling problem for AGVs used in industries. Likewise Kordoghli and Jmali [38] combined GA with Fuzzy logics to solve scheduling problem in a textile industry. They concluded that the scheduling problem cannot be solved by classical priority based rule and GA along with fuzzy logic gives better results especially when the problem is a combinatorial optimization problem. Adnan Tariq [39] solved problem of cell formation for cellular manufacturing system using hybrid GA. The research was divided into machine cells grouping and sequencing problem. Since each cell behaved in a job-shop way, so sequencing problem was solved in a similar way as of job shop scheduling problem. GA was combined with local search techniques to find the optimal solution.

Cuckoo search

Cuckoo search optimization is a new technique developed in 2010. It has been developed on the breeding behavior of cuckoo. There is a lot of potential in working on this area.

Cuckoo search is a new EA developed in 2010 by Yang and Deb [40] in which cuckoo breeding behavior is used for the development of a new metaheuristic algorithm. Levy flights concept has been given to ensure diversification along with intensification of the solution search space. Levy flight optimization methods have been explained in [41] in detail in which types of levy flights have also been discussed. Along with that simulations have been performed after changing various parameters involved in it to study their effect on objective function.

As compared to GA, cuckoo search has only one random factor involved in it. Cuckoo breeding behavior can be studied in [42] with more detail. It is fascinating that in the way these cuckoo show aggressive reproductive strategy.

They do not make their own nests and lay their eggs in another nest of some other species while (not in all brood parasitism) throw the eggs of host species in order to increase the chances of reproductivity because if host bird discovers the cuckoo egg then it throws it away from its nest or abandon the nest. Some species of cuckoo like "tapera" are evolved in such a way that they are specialized in mimicking the color of their eggs according to host species which further increases chances of reproductivity.

So the bottom line is that cuckoo tries to mimic the host egg continuously while the host tries to find a way to detect the parasitic egg that leads to an arms race each trying to survive out the other (survival of the fittest)

Now to generate new solutions levy flight phenomena has been proposed. It can also be studied in detail in [4]. It is seen that many animals show the phenomena of levy flight. Levy flights is a random walk in which step size/length is based on heavy tailed probability distribution. It can be simply treated as an efficient method in order to obtain a random solution. The general algorithm for CS is shown in figure 16.

```
begin
  Objective function $f(\mathbf{x})$,  $\mathbf{x} = (x_1, ..., x_d)^T$
  Generate initial population of
      $n$ host nests $\mathbf{x}_i$ $(i = 1, 2, ..., n)$
  while ($t$ <MaxGeneration) or (stop criterion)
    Get a cuckoo randomly by Lévy flights
      evaluate its quality/fitness $F_i$
    Choose a nest among $n$ (say, $j$) randomly
    if  $(F_i > F_j)$,
        replace $j$ by the new solution;
    end
    A fraction ($p_a$) of worse nests
        are abandoned and new ones are built;
    Keep the best solutions
        (or nests with quality solutions);
    Rank the solutions and find the current best
  end while
  Postprocess results and visualization
end
```

Figure 16: CS Algorithm [40]

Since the CS approach is not so old so a lot of work can be done on it. GA have been applied to a lot many applications in a variety of fields like in FMS, RMS, scheduling, process planning ,design optimization , cruise control, motion planning etc. So CS can be applied in all of these to further investigate its efficiency. Also the levy flight that is based on heavy tailed probability distribution can be made more efficient by varying and applying different types of heavy tailed distribution and check what are the outcomes.

2.3 Process Planning and Optimization

As discussed in section 2.2, various techniques have been proposed in literature for generating process plans in different situations. Need of optimality is a basic and fundamental concern. Kristina [43] did an overview on optimization techniques used in process planning. He concluded that stochastic hill climbing methods have a chance of getting stuck in a locla minima in case of multiple peaks. Different peaks can be searched by starting from a different random solution each time but still it is unprdictable. Simulated annealing and GA was proposed to be better option for CAPP.

Various mathematical models were developed in the past few years for optimization in process planning.Nourali [44] developed a mathematical model for integrated process planning in a flexible assembly system. The methodology for generating process plans and scheduling was developed for assembly lines. Ismail et al [45] considered optimization problem for process planning in multiple parts flow lines involving parallel assembly lines.

Various evolutionary approaches have also been used for optimization of process plans. Krishna and Rao [46] have used ant colony approaches while a simulated annealig approach has been used by Ma et al [47]. Zhang and Nee [48] also used a simulated annealing approach for process planning optimization. Feature recognition method was used to extraxt features and develop alternative route plans for the part. The methodologies lacked consideration of complex precedence contraints between the part features and the non-conformities which may occur by not satisfying the constraints. Also responsiveness needed for the current manufacturing needs was considered and process plans have been developed for a certain part.

2.4 Summary

This chapter gave us detailed overview of techniques which have been used for generating alternative process plans in CAPP. Along with that, due to need of optimization in area of process planning, optimization search areas were discussed in detail to have a better insight of these techniques. Main focus was given to the techniques which can be applied on manufacturing problems.

Chapter 03: Process Plan Generation

3.1 Introduction

As we discussed, most of the techniques used in past for computer aided process planning were based on knowledge based expert system and feature recognition methods. The methods of cogeneration discussed alternative process plans for a part but optimality criterion were not considered. In this present work, a novel methodology is proposed to generate alternative process plans keeping in mind precedence constraints for the part and to be modelled in a way that algorithms for generating optimal process plans can be applied.

A process plan defines the route plan of all the processes to generate the final part from raw material. It basically consists of a series of operations or a set of machining features that are arranged in a systematic way. Machining feature refers to volume removal from the raw material.

The problem of process planning can be sub-divided into two main categories as follows:

a. Operation Selection
b. Operation sequencing

Operation selection means that after analyzing features in part design, selecting operations and allocating machine resources and tool approach directions for the part. A machining feature can be performed by multiple possible operations and machines or it can happen that none of the machine in the manufacturing system can produce the feature. In that case designer may have to do some modifications in the recommended design.

Operation sequencing is to determine the systematic and correct order of operations in a way that none of the operation is left behind and all the precedence constraints are also maintained.

3.2 Setup Planning

Each operation needs a particular tool and setup to be performed. Setup is related to orientation of the part that refers to tool approach direction (TAD). Setup planning is a major task considering process plan sequence. It is considered to have a large impact on product quality when measured in terms of dimensional variance. The purpose of setup planning is to arrange machining features in an appropriate sequence in order to ensure quality and productivity. Tool approach is considered to be important while determining setups. The tool approach direction is an unobstructed path that a tool can take to access the feature. In a process plan different operations can have different TAD and for machining that feature, setup needs to be changed. Thus a different TAD for a feature implies setup change. Operations having similar tool approach directions should be performed first in order to have minimum setup changes. This will avoid setup error and time require for changing setup will also be reduced. This is the purpose of setup formation. Since we know quality is always an important factor. Product quality is affected by the number of setup changes since each time setup is changed due to a different TAD for a feature, work piece has to be dismounted and mounted again on the fixture which will have an impact on its dimensional accuracy. This factor won't have an effect on the parts where dimensional tolerance is large but in other cases, designer has to be careful. So in those cases, one of the major tasks for the designer is to suggest sequence with minimum setup changes.

3.3 Precedence Constraints

Along with setup formation, precedence constraints between the parts also need to be considered. Process plan designer first do the mechanical preliminary analysis of the part and analyze points mentioned below before sequencing the part:

a. Geometrical design
b. Geometrical tolerances
c. Machining features
d. Surface characteristics
e. Material selection

Different types of constraints that have to be kept in mind are:

a. Datum constraints so that features don't intersect
b. Technological constraints for proper sequencing
c. Geometrical constraints for reference purposes

These constraints will be explained below with the help of examples to have a clear understanding. Figure 17 shows an example of datum constraints. F1 is rough turning while F2 is for finish turning. F3 should be performed after F2 since tolerances are tighter for F3 and if F3 has to be performed first for some reasons, then reduction of tolerances should be done first. So the proper sequence would be : F1 $\rightarrow$ F2 $\rightarrow$ F3

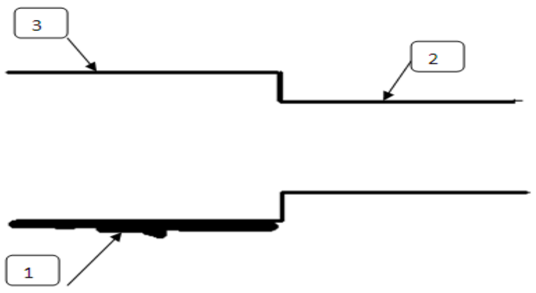


Figure 17: Datum Constraints

Figure 18 gives illustration for technological constraints. Hole 1 has to be performed before hole 2. According to Halevi and Weill [49], hole with more depth and smaller precise diameter should be machined before the hole with a larger diameter and with lesser depth because the straightness of a thinner or more precise hole can be impaired by the larger hole upon their intersection.

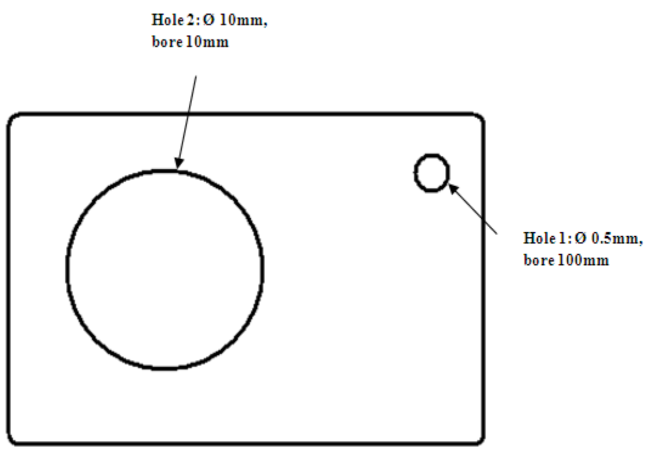


Figure 18: An example of technological constraints

Another example of technological constraint can be observed when a hole needs to be chamfered but since burrs are not allowed in the precise hole, so therefore chamfering must be done before drilling.

Geometrical constraints for reference datum purposes are well explained by figure 19. Since hole 1 works as datum reference for holes surrounding it, hole 1 should be machined first. Also since the depth of hole 1 has to be precise and has to be straight, surface A should be milled first. So the sequence would be:

Milling of surface A → Drilling of Hole 1 → Drilling of nine holes

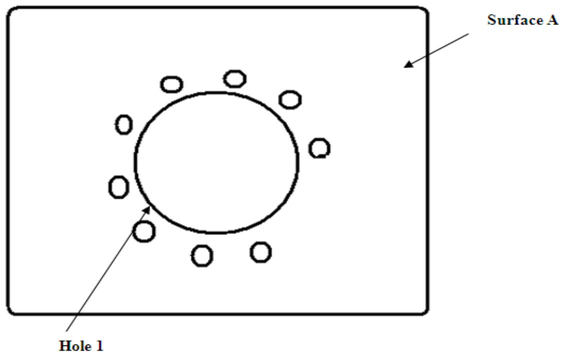


Figure 19: Geometrical Constraints

Another important constraint mentioned by G.H.Ma et al [48] was fixture constraint. This constraint exists between two features when machining one feature before the other one, may cause the other one to be unfixturable. For example consider the figure 20 in which feature 2 is a slot while feature 1 and 3 are milled surfaces on the opposite side. Feature 2 must be milled before the other features. Otherwise the supporting area for milling F2 may not be sufficient.

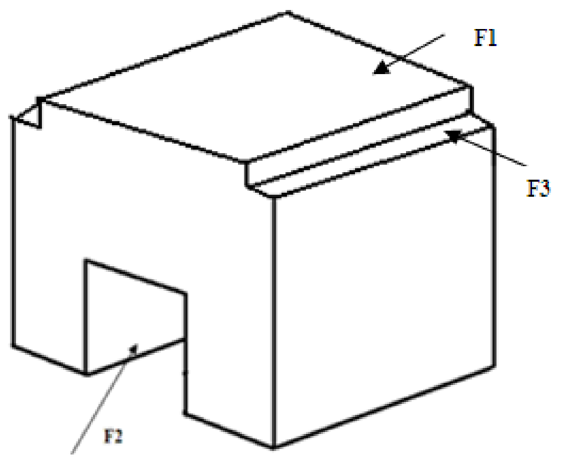


Figure 20: Fixture Constraint

The essence of process planning, as discussed involves determining in what order to perform a select of operations such that the resultant order satisfies the precedence constraints established by both parts and operations. In order to maintain and not to disobey any of the constraint, we would develop a precedence matrix.

Consider a part shown in figure 21. Features which have to be performed on this are

a. Milling
b. Drilling
c. Reaming

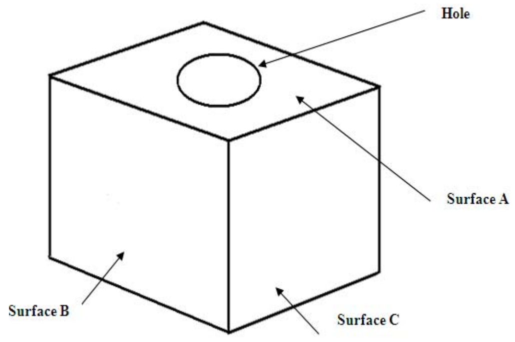


Figure 21: Example Part

Since there could be more than one operation which have to be performed for developing a feature, so description of feature along with operation is given in table 8.

Table 7: Machining features for figure 21

Machining Features	Operations to be performed
Milling (for six surfaces of block)	1.Rough milling of surface A 2.Finish milling of surface A

	3. Rough milling of surface B 4. Finish milling of surface B 5. Rough milling of surface C 6. Finish milling of surface C 7. Rough milling of surface D 8. Finish milling of surface D 9. Rough milling of surface E 10. Finish milling of surface E 11. Rough milling of surface F 12. Finish milling of surface F
Drilling	13. For hole on surface A
Reaming	14. For hole on surface A

Here it can be noted that this scheduling problem is different from the usual travelling salesman problem (TSP). In TSP, if there are n number of cities, then the possible routing options for the salesman would be n!. But in the case of sequencing for a part, options would be less than n!. This is because of the fact that precedence constraints are involved and they make some of the options infeasible. More are the number of constraints, more there are infeasible options.

For generating precedence group matrix (PGM), operations are grouped in such a way that each group represents precedence between operations. The first element of every group has to be performed first. So that element has to be given first priority in a group. It should be noted that there is no precedence among the groups. Since each group represents a constraint (either geometrical or technological) so number of groups in PGM is equal to number of precedence for that part.

Since the datum surface A works as a reference for drill hole, that is why it needs to be performed after milling surface A. All other precedence constraints are logical since finishing cut is performed after rough cut.

To generate feasible plans, 1st element from any group is randomly selected. After that, second element of that group is given 1st priority and it's given zero precedence. For example in start, operations 1, 3, 5, 7, 9 and 11 have 1st priority and any number is chosen randomly from them. Suppose 7 got selected, after that 1, 3, 5, 9, 11 and 8 will have 1st priority and randomly chosen operation would be assigned from them until the sequence is not complete (refer to table [7]). Flow chart would be presented as described in figure 22.

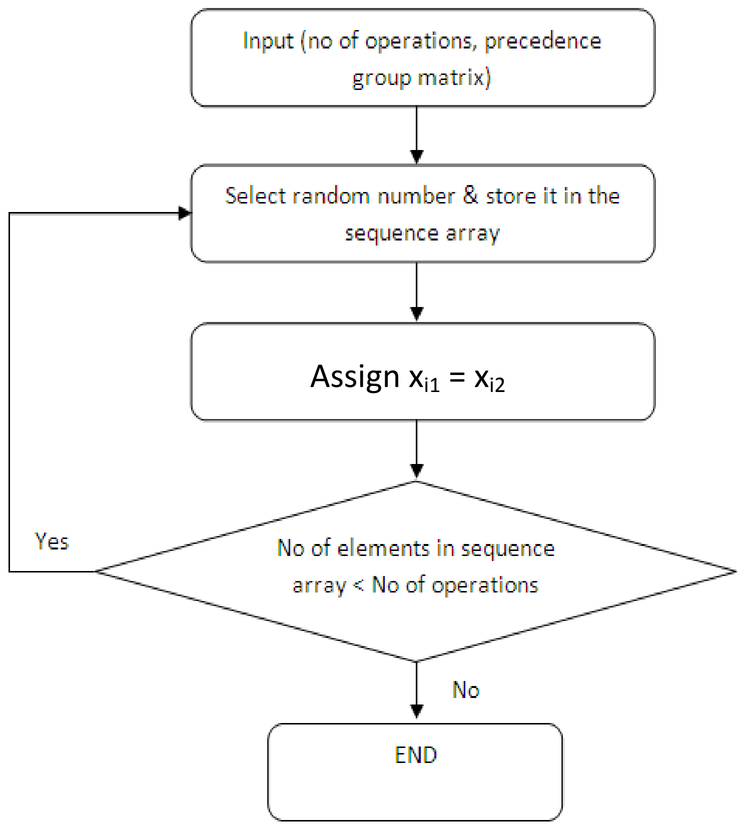

Input (no of operations, precedence group matrix)
Select random number & store it in the sequence array
Assign $x_{i1} = x_{i2}$
No of elements in sequence array < No of operations
Yes
No
END

Figure 22: Flow chart for process plan generation

Where

$${}^{n}\sum_{i=1} X_{ij}$$

n = no of groups in PGM

i = total number of groups in PGM

j= position of element in a group

For example X_{21} represents 1st element of second group of PGM.

A case may arise when there are constraints among features in a way that it may cause repetition of operations in groups. In that case, sequence will contain repetitive elements which will result in an infeasible solution. For this case, the above algorithm may be modified in a way that after selecting each element from PGM, it would be scanned in the sequence array already generated to check for repetition and if repetition occurs, then this element will be discarded and new element would be selected from PGM. The modified flow chart is represented as below in figure 23.

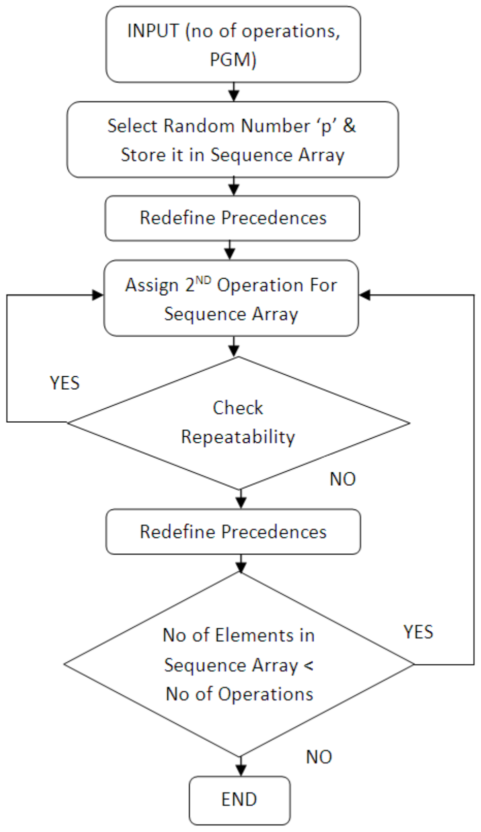

INPUT (no of operations, PGM)
Select Random Number 'p' & Store it in Sequence Array
Redefine Precedences
Assign 2ND Operation For Sequence Array
YES
Check Repeatability
NO
Redefine Precedences
No of Elements in Sequence Array < No of Operations
YES
NO
END

Figure 23: Modified flow chart for process plan generation

The stepwise explanation of the flow chart in figure 23 will be described below

Step 1: Input data

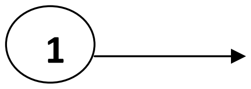
Starting from a part shown in figure 20 the precedence group matrix PGM will be the input shown in table 8.

Table 8: Precedence group matrix for figure 21

Group Number	Precedence between operations
I	1, 2, 13, 14
II	1,3, 4
III	5, 6
IV	7, 8
V	9, 10
VI	11, 12

Step 2: Choose members having zero precedence

The steps having zero precedence are those which come first in the groups defined in PGM. In this example, operation numbers 1,5,7,9 and 11 have zero precedence. So any random operation would be selected from the list. Suppose in this case operation 1 is selected from group I and added in the sequence array.



Step 3: Re assign Precedence

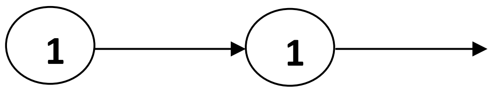
In this step, PGM will be re-ranked. Continuing from step 1 and 2, if Op 1 is selected from the list, then new PGM for the part will be as follows

Table 9: PGM for example part for figure 21

Group Number	**Precedence between operations**
I	2, 13, 14
II	1,3, 4
III	5, 6
IV	7, 8
V	9, 10
VI	11, 12

Step 4: Assign 2nd operation for sequence

Now random element will be selected from new PGM in table 9. Now the operations having zero precedence will be 2, 1, 5, 7, 9 and 11. Here we can see that there is a possibility for operation to be selected twice. Suppose operation 1 is again selected.



Step 5: Checking Repeatability

A check is made that whether the last selected operation is already present in the sequence array or not. If it does like in step 4, then the last selected operation would be discarded and step 4 would be repeated again. Continuing from step 4 described above, Op 1 will be discarded and a new operation would be selected from new PGM in table 9.

$$\begin{tikzcd} 1 \arrow[r] & 5 \arrow[r] & {} \end{tikzcd}$$

Step 6: Check for completion

Sequence will be checked for completion. If number of elements in a sequence array are equal to number of operations, then algorithm would be stopped. If not, then step 3 to step 6 would be repeated again. A complete sequence for the part would be

$1 \rightarrow 5 \rightarrow 2 \rightarrow 3 \rightarrow 7 \rightarrow 9 \rightarrow 6 \rightarrow 11 \rightarrow 10 \rightarrow 13 \rightarrow 14 \rightarrow 12 \rightarrow 4$

For our case study, we consider the part ANSI 101 shown in figure 24.

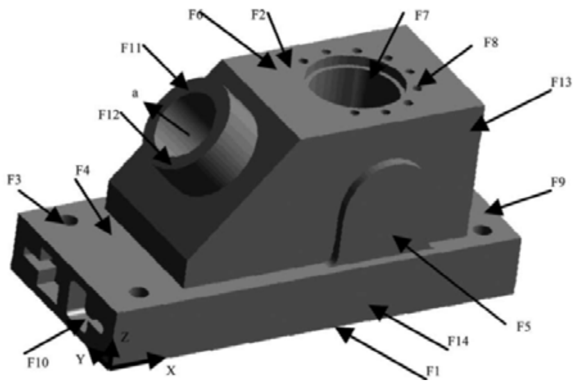


Figure 24: ANSI 101

Features and PGM for the part in figure 24 are given in table 10 and 11 respectively.

Table 10: Machining features for ANSI 101

Operation ID	Feature/ Operation	Operation ID	Feature/ Operation
1	Milling	11	Milling
2	Milling	12	Step
3	Drilling	13	Milling
4	Step Milling	14	Boss
5	Milling	15	Compound hole drill
6	Milling	16	Boring
7	Compound hole drill	17	Reaming
8	Boring	18	Pocket
9	Reaming	19	Pocket
10	Nine holes drill	20	Reaming

Table 11: PGM for ANSI 101

Group Number	Precedence Constraints
I	1,2
II	14,2
III	1,14,6
IV	1,14,2,7
V	14,1,2,10,8
VI	14,1,4,3,9
VII	1,14,18
VIII	1,14,19
IX	1,14,4,5
X	1,14,4,13
XI	1,14,4,11,12
XII	1,14,4,11,12,15,16,17
XIII	1,14,20

Following the algorithm, a randomly generated process plan for this part can be:

$1 \rightarrow 14 \rightarrow 2 \rightarrow 6 \rightarrow 2 \rightarrow 10 \rightarrow 7 \rightarrow 8 \rightarrow 4 \rightarrow 18 \rightarrow 19 \rightarrow 3 \rightarrow 5 \rightarrow 12 \rightarrow 9 \rightarrow 13 \rightarrow 15 \rightarrow 11 \rightarrow 16 \rightarrow 20 \rightarrow 17$

It can be seen that every part satisfies the constraints mentioned in PGM.

3.4 Summary

In this chapter we defined a methodology for modeling alternative generation of process plans which can be used to find the optimal process plan based upon the set criteria. The algorithm developed can be used for prismatic as well as rotational parts. Also it is not limited to simpler parts and can be applied to complex ones involving complex constraints among different machining features.

Chapter 04: Proposed Methodology for Optimization

4.1 Background

This chapter deals with the methodology proposed for dealing with optimization problem through evolutionary approach. As discussed in chapter 2, basic methodology for process planning by genetic algorithms and cuckoo search will be proposed. After the process plan generation, first step is to define the fitness criteria or objective function which we want to optimize.

4.2 Objective Function

Certain needs have to be kept in mind while going for the optimization of process plans. Various criteria for optimization have been used in literature. In [35], the author used minimal machining time for rotational components. [50] used minimal machine changes and tool change as an optimization criteria but it was only for rotational components. Also both these criteria were considered one by one. In current proposed methodology, minimal tool change and set up change is set to be as an objective function. Weightage method has been used for handling multi-objectives because of their dependency on each other.

Tool change matrix has been shown in table 12 for ANSI part shown in figure 24. When there is a tool change from operation 1 to 2 then it would assign penalty 1 in the matrix otherwise zero. For example, row 2 and column 15 of table 12 (i.e. between op 1 and 14) shows a penalty of zero which means that operation 1 and 14 can be performed with same tool (i.e. end milling). Similarly in row 2 and column 4 (i.e. between op 1 and 3), an entry 1 depicts that operation 1 and 3 cannot be performed by using same tool since operation 1 is milling while operation 3 is drilling operation.

Table 12: Tool change matrix for ANSI 101

Op#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	0
2	1	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1	1	1
3	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
4	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
5	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
6	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
7	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
8	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
9	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
10	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
11	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1
12	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1

13	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1
14	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1
15	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1
16	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1
17	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1

No of tool changes in a sequence will be

$$\mathbf{TC_i = {}^{jn-1\rightarrow jn}\sum_{j\rightarrow j+1} TC}$$

Where, TC_i = no of tool changes in sequence i

j = 1, 2 ,3, ... n, n = no of operations in a sequence

Setup change matrix as detailed in section 3.2 will be generated by tool approach directions. If an operation has different TAD from the other, then it would be assigned a penalty of 1, otherwise zero. Setup change matrix for ANSI part is given in table 13.

Table 13: Setup change matrix for ANSI 101

Op#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	0
2	1	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1	1	1
3	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
4	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
5	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
6	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
7	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
8	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
9	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
10	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
11	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1
12	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1
13	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1
14	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1
15	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1
16	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1
17	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1

No of setup changes in a sequence will be

$$\mathbf{SC_i = {}^{j}{}_{n-1}{}^{\rightarrow j}{}_{n} \sum_{j \rightarrow j+1} SC}$$

Where SC_i = no of setup changes in sequence i

n = no of operations in a sequence

Fitness criteria would be combined as

$$\text{Objective Function} = W1 * \mathbf{TC_i} + W2 * \mathbf{SC_i}$$

where W1 and W2 are the weightage factors for tool change and setup change respectively. They will be assigned after keeping in mind considerations of design requirements of product. If tolerance requirements for a part are very tight then it is best for a manufacturer to minimize setup change so as to minimize fixture changes and not to realign the part multiple times for machining features. In that case, W1 will be kept higher. If tooling is expensive then W2 will be kept higher.

4.3 Selection Function

After ranking the initial generation according to the aforesaid objective function, parents have to be selected for creation of next solution strings (i.e. children). It is considered to be very important phenomena since it determines the solution space. Basic purpose of every selection mechanism is to enhance the chances of generating fitter next generation. Various methods have been proposed by researchers for selection method. Sivaraj et al [23] presented an overview of optimization methods. They concluded that the selection pressure which is the process of selection of fitter parents for next generation, is the most critical factor in determining selection mechanism since it determines the diversity of solution space. If it is set too low, then there the convergence towards optimal solution would be too low and if it is set too high, then there will be a faster convergence but diversity will be lost and chances of getting stuck in a local optima will be much higher.

In present research, elitist method proposed by Thierens and Glodberg [51], with a little modification has been used for selection purposes. In this method, the fittest individuals are copied to the next generation and are selected for generating next population as a parent. After generating equal number of children as of population, all are evaluated and ranked accordingly. The fittest ones equal to population size of initial population then survive to the next generation while the other ones are discarded. It is used to avoid the loss of fitter solutions since in this case it can happen that after mutation function, precedence constraints may be violated leading to infeasible and unaccepted solution.

4.4 Crossover

As detailed in chapter 2, parent recombination to generate new feasible solutions is known as crossover. The traditional approaches for crossover as discussed in chapter 2 cannot be used in this problem of process planning. Let us consider an example here where in a string of 8 operations, 4th place is considered as crossover point for single point crossover.

Parent 1 = 1 3 5 7 │ 4 8 6 2

Parent 2 = 2 4 1 3 │ 6 8 7 5

Children 1 = 1 3 5 7 6 8 7 5

Children 2 = 2 4 1 3 4 8 6 2

Here we can see that in both children, elements are repeating. However practically it is not possible since one operation cannot be machined two times in a sequence. So the process plans become infeasible.

This repetition problem was considered somewhat similar as in traveling salesman problem where a salesman has to find shortest path between the cities, and a city cannot be visited more than once. Liang and Chin [52] proposed OX crossover to avoid this problem of repetition resulting in an infeasible solution. This operator is preceded as

Parent 1 = 1 3 4 │2 9 8 6│ 7 5 10

Parent 2 = 10 7 6│ 4 9 8 2│ 3 1 5

After selecting a portion from both parents, copy this sequence in temporary created off spring,

O1 = x x x│2 9 8 6│ x x x

O2 = x x x│4 9 8 2│ x x x

Copy the portion from 2nd crossover point to the end in temporary created parents as

P1'' = 7 5 10 1 3 4 2 9 8 6

P2'' = 3 1 5 10 7 6 4 9 8 2

Now portions selected from O1 and O2 will be subtracted from P2'' and P1'' respectively, resulting in R1 and R2. These R1 and R2 will be copied from the second crossover point in O2 and O1 respectively to replace the x positions. The resulting offspring will be as follows;

C1 = 10 7 4 2 9 8 6 3 1 5

C2 = 7 5 10 4 9 8 2 1 3 6

This proposed method solves the problem of repetition but it may happen that it violates any precedence constraints present between the parts. For example in above example if there was precedence between 2 and 5 such that 2 must be machined before 5, but after crossover it will be violated and the resulting sequence will be infeasible and instead of moving towards better solutions we will get worse ones.

Due to this reason, position wise crossover is used in present methodology. This will be preceded as

Parent 1 = 1 3 4 │2 9 8 6│ 7 5 10

Parent 2 = 10 7 6│ 4 9 8 2│ 3 1 5

After selecting a portion from both parents, copy the remaining sequence in temporary created off spring,

O1 = 1 3 4│x x x x│ 7 5 10

O2 = 10 7 6│x x x x│ 3 1 5

Now scan parent 2 for O1 and fill the operations present in cut portion of parent 1 in the order which was followed in parent 2. Same method is being adopted for O2. Resulting offsprings will be

O1 = 1 3 4 6 9 8 2 7 5 10

O2 = 10 7 6 4 2 9 8 3 1 5

This technique ensures that precedence won't be disturbed and results produced will be feasible.

4.5 Mutation

Fast and premature convergence is seen to be one of the frequent problems whenever GA is applied to multi-objective problems and results in a getting a local optima. It happens when offspring being produced are no longer better than their parents. This problem is related with loss of diversity. As discussed in chapter 2, mutation operator is used to avoid this situation. This operator is applied after crossover.

Two point swapping is performed on the offspring produced after crossover as follows

O1 = 1 3 │4│ 6 9 8│2│ 7 5 10

Two mutation sites are chosen randomly for swapping purposes. In the above example 4 and 2 have been chosen.

O1'' = 1 3 2 6 9 8 4 7 5 10

However it can be seen that it is possible that after this swapping, a precedence may get violated resulting in an infeasible solution. Carlson [53] summarized three methods of handling infeasible solutions which are

- Eliminating infeasible solutions
- Repairing infeasible solutions
- Applied penalty methods

The elimination method has been shown by several authors as a poor approach. Repair schemes can be applied where there are explicit and independent constraints such as traveling salesman problem. For implicit constraints and combinatorial objective functions, penalty method is proposed to be the most useful method and there are different methods of applying it. Further details can be studies in [53]. In the present proposed methodology, penalty is applied by adding a number in the fitness function. Fitness/objective function is been modified as

Objective Function = $W1 * TC_i + W2 * SC_i + P_i$

P_i = penalty in sequence i

Penalty matrix is developed by keeping precedence constraints for given part in consideration. Whenever a sequence violates any constraint, a penalty of 100 is assigned to it and added into the objective function. Penalty matrix for ANSI 101 part is shown in table 14. For example in matrix, column 3 and row 2 shows an entry of 100. This means that if operation 2 takes place before 1, then a penalty of 100 would be assigned. This is because operation 1 serves as datum for operation 2. Hence operation 1 has to be performed before 2.

Table 14: Penalty matrix for ANSI 101

Op#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	100	100	0	100	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0
4	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

5	100	100	0	100	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0
6	100	100	0	100	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0
7	100	100	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0
8	100	100	0	0	0	0	0	0	0	100	0	0	0	100	0	0	0	0	0	0
9	100	100	100	100	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0
10	100	100	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0
11	100	100	0	100	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0
12	100	100	0	100	0	0	0	0	0	0	100	0	0	100	0	0	0	0	0	0
13	100	100	0	100	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	100	100	0	0	0	0	0	0	0	0	0	100	0	100	0	0	0	0	0	0
16	100	100	0	0	0	0	0	0	0	0	0	100	100	100	100	0	0	0	0	0
17	100	100	0	0	0	0	0	0	0	0	0	100	0	100	100	100	0	0	0	0
18	100	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0
19	100	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0
20	100	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	100	0

4.6 Stopping Criteria

After mutation, generation produced will be evaluated on fitness criteria. After that, solutions sequences having lesser fitness will be discarded keeping population size same. Frequently used stopping criteria in GA are stalling limit, tolerance value and the number of generations.

Stalling limit is the one in which algorithm stops when fitness value stops to change over a defined number of generations. Tolerance limit criteria cannot be used in this case since we don't know the solution space. Studies reveal that when elitist model is used for selection of parents, stalling limit is not suitable as stopping criteria since it shows erratic behavior. So no of generations/iterations is used here. Flow chart for the proposed methodology is presented in figure 25.

4.7 Cuckoo Search Methodology

As detailed in chapter 2, cuckoo search is proposed by Yang and Deb [40]. After the ranking of initial population, new population/solution will be created by Levy Flight phenomena.

Levy flight is a type of random walk and it follows Markov's chain phenomena. It implies that the future value or state for problem under discussion depends upon its present value and not on its past. Randomness is considered to be a very important factor since it plays phenomenal role in exploration and diversification of solution search space. Different methods have been proposed for implementing levy flight and choosing random step size. In current methodology, step size proposed by X.Yang [54] has been adopted. Step length is calculated as follows

$$\text{Step size} = \acute{\alpha} * U / |V|^{1/\beta} \qquad (3.1)$$

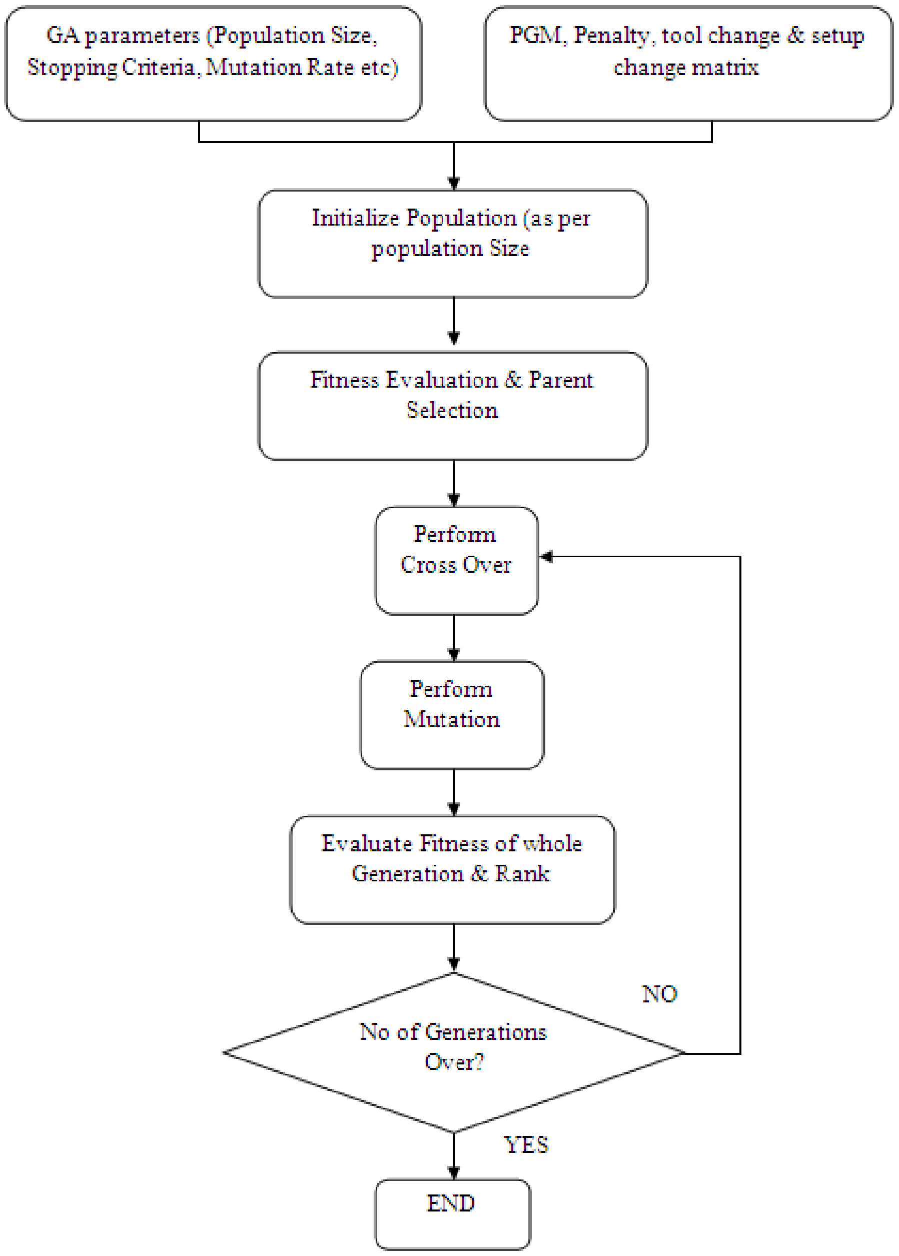

GA parameters (Population Size, Stopping Criteria, Mutation Rate etc)
PGM, Penalty, tool change & setup change matrix
Initialize Population (as per population Size
Fitness Evaluation & Parent Selection
Perform Cross Over
Perform Mutation
Evaluate Fitness of whole Generation & Rank
No of Generations Over?
NO
YES
END

Figure 25: Flowchart for proposed GA Methodology

where U and V are randomly generated numbers between 0 and 1.

$$1 \leq \beta \leq 3 \text{ (range of } \beta \text{ for levy flight)}$$

This step size is added in the previous solution to generate new solution. Since our solution in this case represents a process plan, so there we will use modification in using levy flight. Step size will be multiplied with a factor ά which is related with scales of the problem. In our case we will take it as 10^5. This is done due to incorporate this step length in our process plan. An example is described below

Parent Process Plan = $3 \rightarrow 5 \rightarrow 6 \rightarrow 8 \rightarrow 1 \rightarrow 2 \rightarrow 9 \rightarrow 10 \rightarrow 7 \rightarrow 4$

$\beta = 1.5$, Step length = 23018

For generating child, 8 will be added in 4, 1 will be added in 7 and so on. Since we can see that 8+4 generates 12 which is not possible. In this case it will retain its original number i.e. 4. If after adding numbers, two numbers are repeated in the sequence (for example 8 and 1+7 in this case) then the original position of these two numbers will be swapped. So the new sequence will be

Process plan (child) = $3 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 1 \rightarrow 4 \rightarrow 9 \rightarrow 10 \rightarrow 8 \rightarrow 2$

It may happen that after this swapping, process plan may be infeasible due to violation of precedence constraints. In this case, penalty will be automatically imposed on the sequence resulting in lesser fit solution which will be discarded at the end of each generation while population/solution size will be remain same for each generation. Flow chart for process plan generation by CS methodology is described in figure 26. The proposed CS methodology is explained in steps by taking example of ANSI part.

Step 1: Input

Input in the form of process plans as described in section 3.3, stopping criteria and population size is provided. For understanding the concept, let us consider a process plan for a part having 10 operations as

Parent = $1 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 10 \rightarrow 2 \rightarrow 8 \rightarrow 6 \rightarrow 9$

Step 2: Levy Flight Application

The steps of levy flight are described in detail in figure 27.

Step 2A: First a random number is generated from levy walk. For instance referring to equation

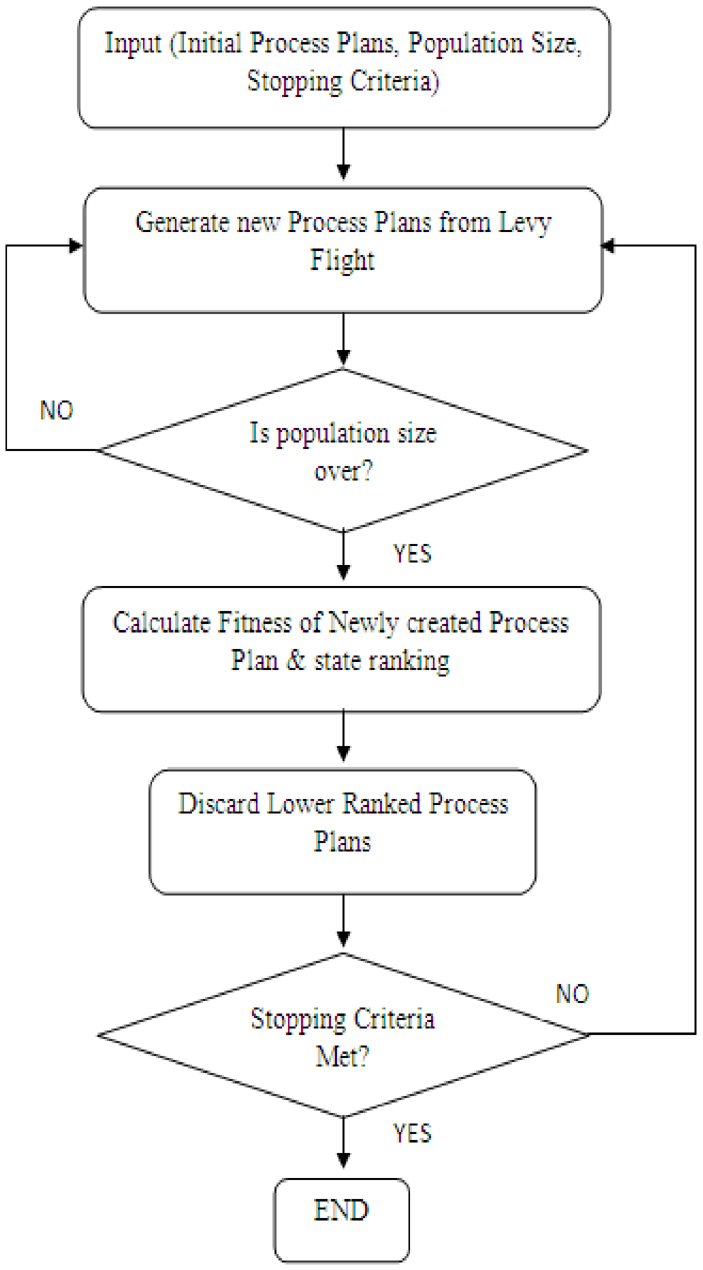

Input (Initial Process Plans, Population Size, Stopping Criteria)
Generate new Process Plans from Levy Flight
NO
Is population size over?
YES
Calculate Fitness of Newly created Process Plan & state ranking
Discard Lower Ranked Process Plans
NO
Stopping Criteria Met?
YES
END

Figure 26: Flowchart for CS

3.1, the random number generated is

β = 3 9 4 2 2

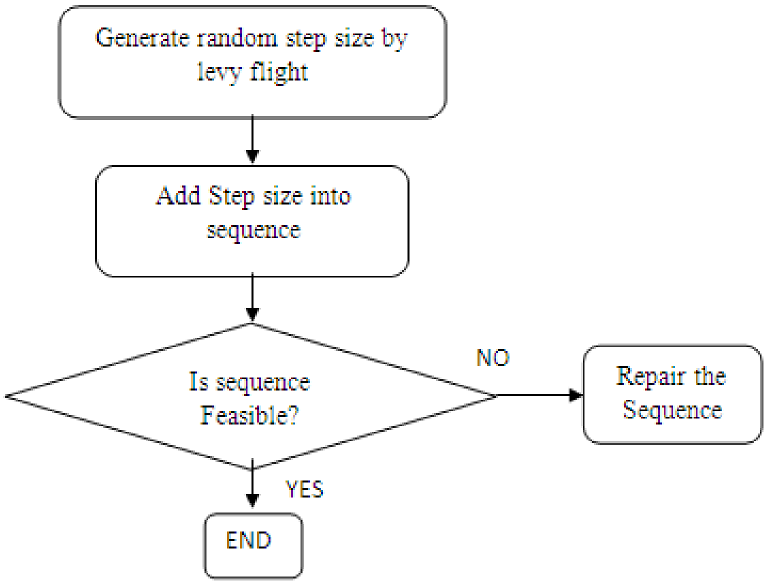

Generate random step size by levy flight
Add Step size into sequence
Is sequence Feasible?
NO
Repair the Sequence
YES
END

Figure 27: flowchart for levy flight

Step 2B: β will be added in the parent process plan.

$$\begin{array}{lcccccccccc} \text{Parent} = & 1 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow & 10 \rightarrow & 2 \rightarrow & 8 \rightarrow & 6 \rightarrow & 9 \\ & & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow \\ & \beta \quad = & 3 & 9 & 4 & 2 & 2 \end{array}$$

Child = 1 → 3 → 4 → 5 → 7 → 13 → 11 → 12 → 8 → 11

Step 2C: the sequence generated will be checked whether to be feasible or not. Continuing from step 2B, we can see that generated child is not feasible since some elements are greater than no of operations. So a repair scheme will be applied. The repair scheme, applied in two steps, will be as follows:

If any element in the array is greater than, less than or equal to zero, then the child new sequence will retain element of its parent sequence. So the child would be modified to:

Child = 1 → 3 → 4 → 5 → 7 → 10 → 2 → 8 → 8 → 9

If any element in the array is repeated, then it would be swapped from its parent sequence. The new sequence will be

Child = 1 → 3 → 4 → 5 → 7 → 10 → 2 → 6 → 8 → 9

Step 3: Population Size

If number of new solutions generated are equal to population size, then algorithm will move forward, otherwise step 2 would be repeated.

Step 4 & 5: Fitness Evaluation

Newly generated process plans will be ranked along with their parents based on their fitness. Solution sequences having lower fitness will be discarded (population size for next generation will remain same).

Step 6: Stopping Criteria

If the generated process plan meets the criteria, then algorithm would stop, otherwise step 2 to 5 would be repeated. Here the stopping criterion is set to be no of generations.

4.8 Summary

In this chapter, methodologies for generating optimized process plans by both CS and GA methodologies. In coming chapter, comparison of these methodologies on test functions as well as at process planning will be performed.

Chapter 05: Comparison

5.1 Background

In chapter 4, two methodologies of evolutionary approach were presented to solve process planning problem. A comparison of between these approaches along with other optimization techniques already been proposed in the literature will help us understand to have a better picture of these proposed methodologies.

5.2 Comparison between CS and GA

There are many test functions in literature which are used frequently to test efficiency and characteristics of optimization algorithms. Here we will compare the results of these test functions by applying our developed methodology on both. We took Rastrigin's and sphere function to check the performance of CS and GA. Rastrigin's function as shown in figure 28 has multiple peaks exhibiting local minima and a global minima at 0,0. It is a non-convex function due to which it is hard to find its global minima through classical techniques.

$$f(\mathbf{x}) = An + \sum_{i=1}^{n} \left[x_i^2 - A\cos(2\pi x_i)\right]$$

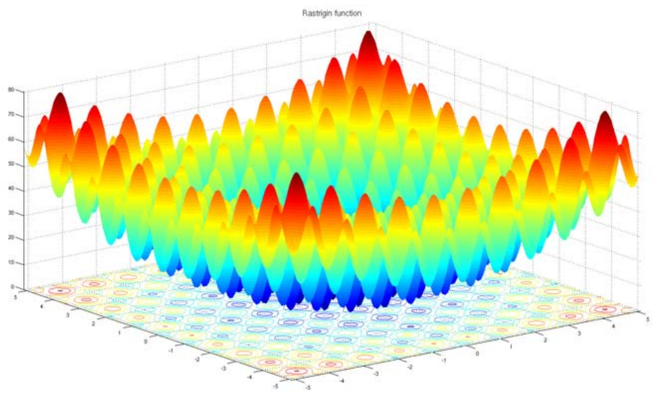


Figure 28: 3-D plot for Rastrigin's Function

Sphere function has only one global minimum as shown in figure 29.

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2.$$

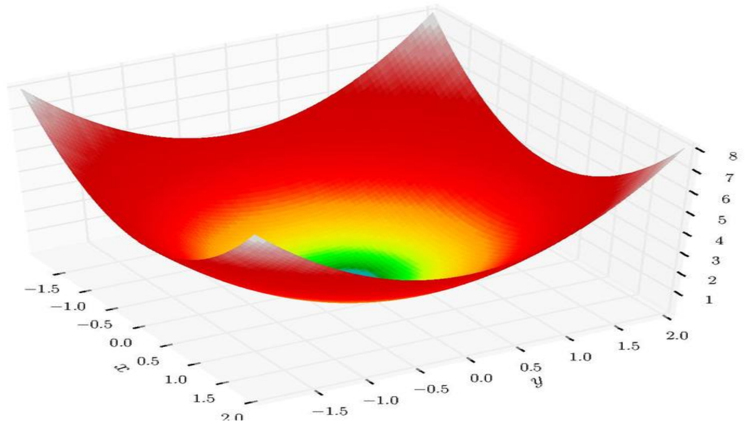


Figure 29: 3-D plot for Sphere Function

Comparison of both GA and CS on these test functions is listed below in table 15.

Table 15: Comparison of GA and CS on test functions

Test Functions	No of Variables	CS	GA
Rastrigin's Function	2	f(x) = 0.18e-12 population size = 20 No of iterations = 56	f(x) = 0.03 population size = 20 No of iterations = 62
Sphere Function	2	f(x) = 0.18e-12 population size = 20 No of iterations = 56	f(x) = 0.18e-12 population size = 20 No of iterations = 56

It can be seen that results obtained through CS on these test functions are better than GA. Now we will apply GA and CS on ANSI part 101 shown in figure 24 (refer to Annex A & B for GA and CS codes repectively). The results obtained from both methodologies are compared in table 16.

Table 16: Comparison of GA and CS on ANSI 101 part

Methodology	Optimum Sequence	Computation Time	Optimum value for combined objective function	No of iterations
GA	14,1,6,2,18,20,7,4,19,13, 3,11,12,15,10,5,8,9,16,17	10 sec	12	110
CS	14,1,2,4,18,6,13,20,7,10, 3,11,8,12,19,5,15,9,16,17	17sec	13	920

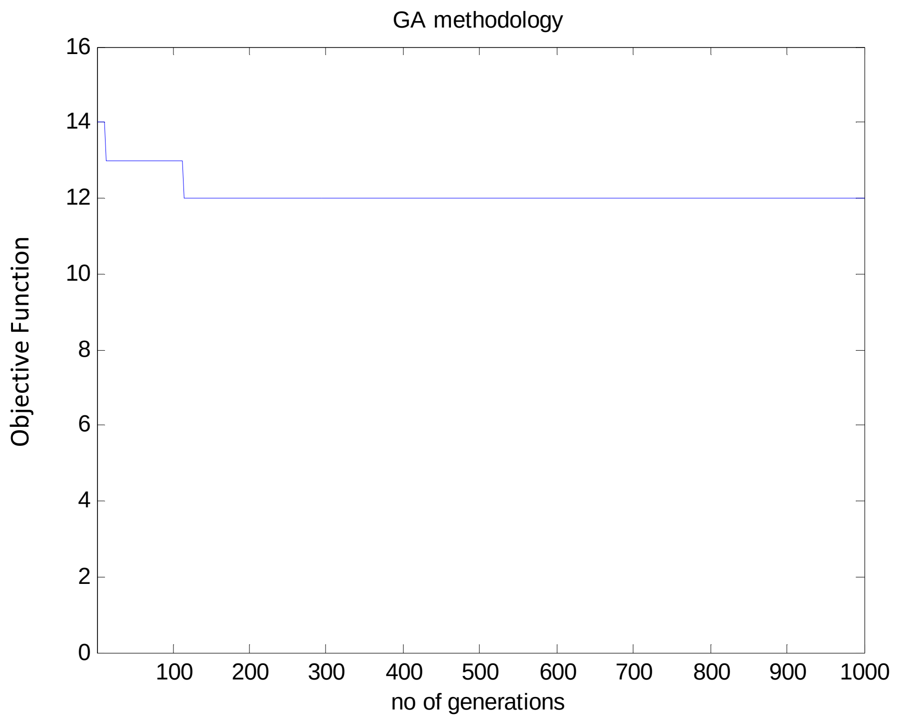


Figure 30: Results obtained for ANSI 101 by GA

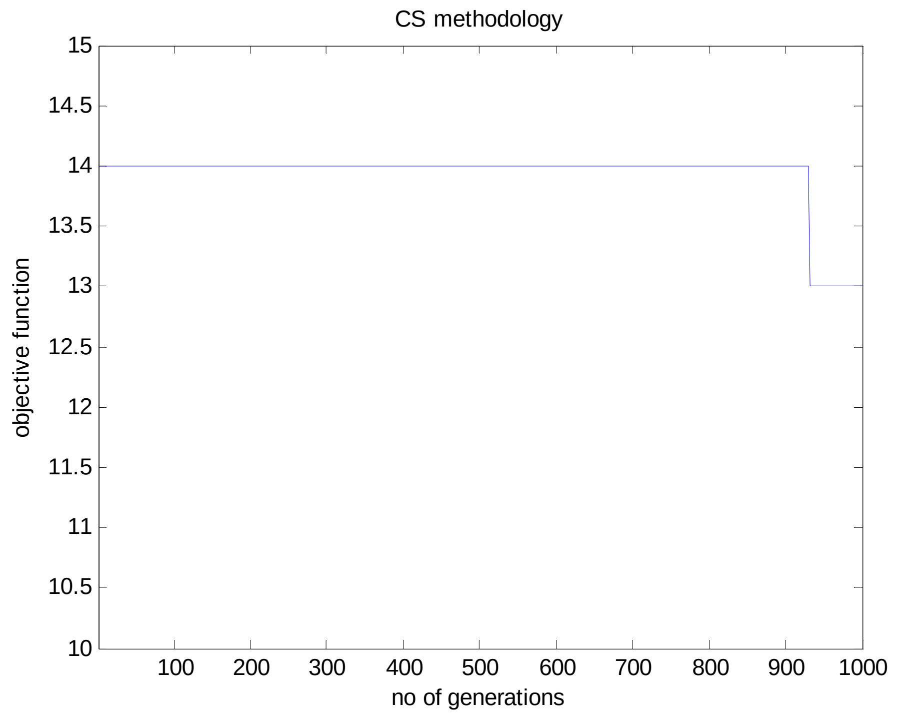


Figure 31: Results obtained for ANSI 101 by CS

5.2.1 Conclusion

It can be observed that when precedence constraints are involved, GA seems to give better results than CS approach. This is due to the fact that in order to apply CS methodology on process planning problems, we have to produce new solutions/sequences via levy flights. But when levy flight is applied on a plan, it violates precedence constraints while repairing itself as described in chapter 4. Penalty would be assigned to that sequence due to which its fitness gets disturbed and while ranking, it gets discarded from the parent generation.

5.3 Comparison of GA methodologies in Process Planning

Another approach for process planning was proposed by Kumar and Deb [55]. Different objective functions were considered separately to be minimized. Elitist model is used for selection purposes. The flow chart of GA methodology proposed is shown in figure 32.

The proposed methodology is applied on part in figure 33 while the operation sequences and TAD for the part is given in 34. The Tool change, setup change and penalty change matrices for the part are described in table 22, 20 and 21 respectively (Refer to Annex-C).

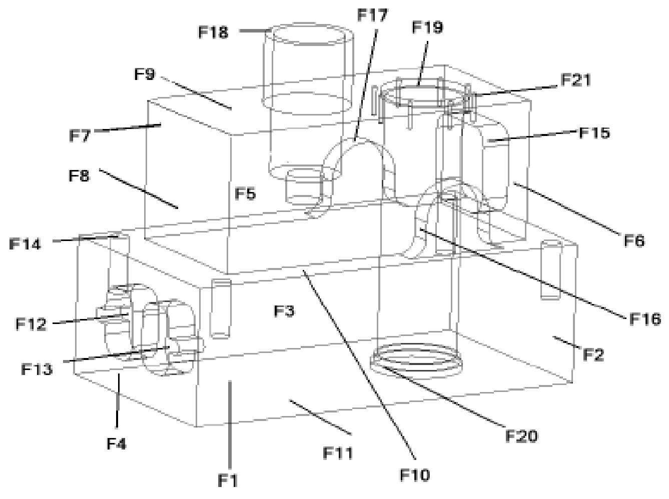


Figure 32: Example Part for Case Study [55]

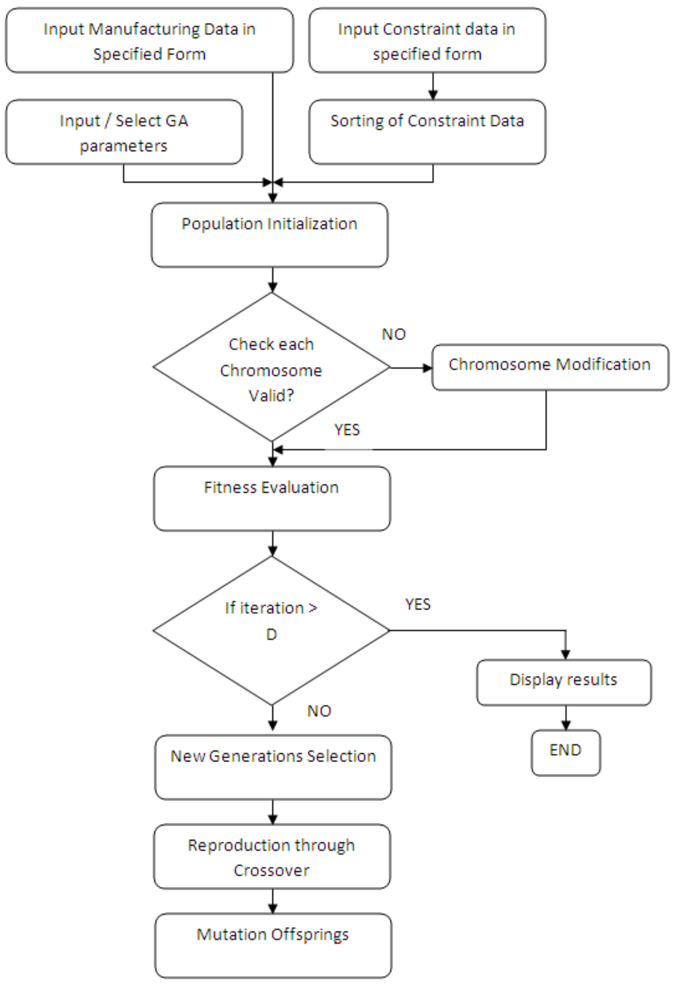

Input Manufacturing Data in Specified Form
Input Constraint data in specified form
Input / Select GA parameters
Sorting of Constraint Data
Population Initialization
Check each Chromosome Valid?
NO
Chromosome Modification
YES
Fitness Evaluation
If iteration > D
YES
Display results
END
NO
New Generations Selection
Reproduction through Crossover
Mutation Offsprings

Figure 333: Flowchart for generation of process plans by GA [55]

Table 17: Input data with machining features for Example part in figure 32

Operation Number	Operation Type	TAD
1	Face Milling	6
2	Face Milling	3
3	Face Milling	5
4	Face Milling	1
5	Face Milling	6
6	Face Milling	3
7	Face Milling	5
8	Face Milling	1
9	Face Milling	2
10	Face Milling	2

11	Face Milling	4
12	End Milling	1
13	End Milling	1
14	Drilling	2
15	Drilling	2
16	Drilling	2
17	Drilling	2
18	End Milling	3
19	End Milling	6
20	End Milling	5
21	End Milling	2
22	Boring	2
23	Drilling	2
24	Boring	2
25	Drilling	4
26	Boring	4
27	Drilling	2
28	Tapping	2

Table 18: PGM for Example part in figure 33

Group No	**Precedence**
I	11 , ALL OPERATIONS
II	9,24
III	9,27,28
IV	25,26,10,9
V	1,5
VI	10,14
VII	10,17
VIII	10,15
IX	10,16
X	21,23,22
XI	5,10
XII	6,10
XIII	7,10

XIV	6,18
XV	4,12
XVI	4,13
XVII	20,7
XVIII	3,7
XIX	25,24
XX	18,4
XXI	4,8

Comparisons of the results obtained by both methodologies are described in table 19 (Refer to Annex-D for MatLab code of proposed GA methodology on part in figure 33).

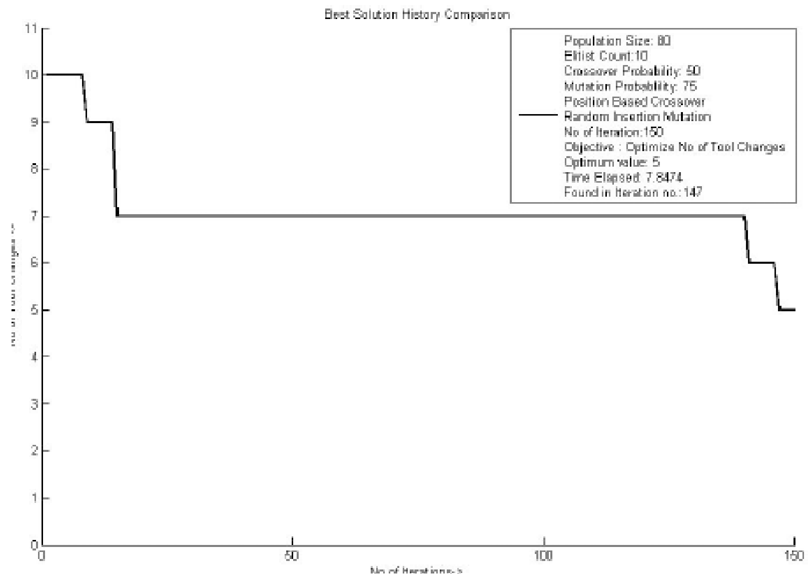
Table 19: Comparison of GA methodologies for generation of process plans

Methodology	Optimum value for tool change	Optimum Sequence	No of iterations
GA by Deb	5	11,2,6,18,19,20,3,7,1,4,8,26,10,9,12,13,21,15,16,17,23,25,27,4, 28,14,5,22	147
Proposed GA	5	11,2,3,19,1,26,6,18,4,8,5,20,7,12,10,9,13,21,25,27,28,23,22,24,17,16, 15,14	15

The results obtained are shown in figures 34 and 35.

5.4 Comparison and Summary

The results obtained from proposed GA methodology in chapter 4 are compared with methodology presented by Deb in [55]. The optimum value for tool change i.e. 5 is obtained in 15 iterations. The generated process plan doesn't violate any constraint and is completely feasible. This is due to the fact that subroutine developed for modifying the sequences after mutation not only increases the processing time, but it also may neglect some of the constraints resulting in an infeasible solution.



Population Size: 80
Elitist Count:10
Crossover Probability: 50
Mutation Probability: 75
Position Based Crossover
Random Insertion Mutation
No of Iteration:150
Objective : Optimize No of Tool Changes
Optimum value: 5
Time Elapsed: 7.8474
Found in Iteration no.:147

Figure 34: Results obtained for optimal no of tool changes by GA methodology of Deb [55]

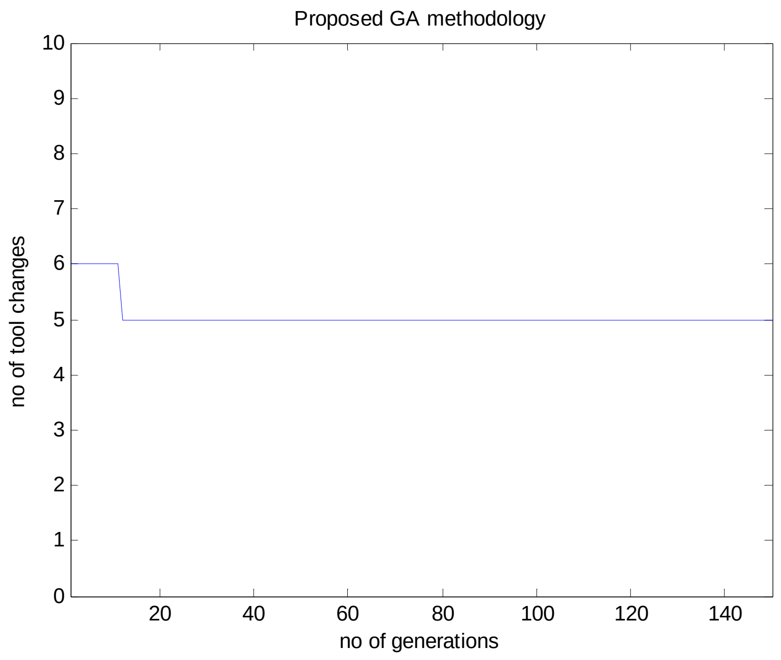


Figure 35: Results obtained for optimal no of tool changes by proposed GA methodology

Chapter 06: Analysis & Future Recommendations

6.1 Analysis & Future Recommendations

This thesis falls in the domain of optimization in advanced manufacturing systems. It aims in generating alternative process plans for a part/part family. The issues addressed in this thesis include not only generation of process plan around a part family, but also to make it reconfigurable so that it can accommodate future needs and can be made responsive. Along with that, search areas in optimization are also discussed in order to yield optimal process plan from the alternatives. The methodology proposed to address the aforesaid issues can be divided as follows:

a. A detailed literature review on techniques proposed for creating alternative process plans such as feature recognition based techniques, co-generation and knowledge based techniques. The literature review yields that generated process plans do not consider optimality criteria.
b. A detailed literature review of search areas in optimization was discussed. It was observed that classical calculus based techniques are not so popular when it comes to solve manufacturing problems.
c. Development of a methodology for generating process plans by keeping the precedence constraints for a part under consideration. Along with that, the proposed methodology was constructed in such a way that later it can accommodate new features in a part.
d. Development of algorithms for generating optimal process plan for defined criteria. The developed algorithms were compared to identify the better methodology. Along with that, the better proposed methodology was compared with algorithm proposed in literature. It was analyzed that due to lesser checks and better methodology for generating initial input process plans, proposed algorithm gives better results.

By using EAs, optimization techniques were combined with manufacturing application. It was analyzed that GA gives better results than CS if precedence constraints are not considered. Moreover, proposed GA methodology provided better results and responsive process plans as compared to the one already existing in literature. Certain future work recommendations which can be an extension to the present work are as follows:

a. Parallel sequencing can be considered for cases in which minimum machine time is required and precision is not an issue.
b. Different machine options can be considered and choice can be provided if the part to be made requires has features that require operation on various machines.

c. Along with the specified criteria, parameter optimization can be performed resulting in suggesting optimal feed, speed and depth of cut for the part. This will require detailed literature review on tooling and cutting force directions.

d. Optimization algorithms can be modified to extend their use in other reconfigurable manufacturing problems for example dynamic facility planning.

Appendix A

This Appendix contains MATLab Code in GA methodology for ANSI 101 part.

MATLab Code (GA Methodology)

```
function []=ansiga()
k=1;
e=1;
pop=6;     %Pop represents population size
parent=1;
cost_m=[0 0 1 1 1 1 1 1 1 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0;0 0 1 0 1 1
1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0;0 1 1 0 1 1 1 1 1 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1;1 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1
1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0
0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1
0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1
1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1
1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
setup_m=[0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1
1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0;0 0 1 0 1 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1
1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0
0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1
0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1
1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1
1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
penality_m=[0     0     0     0     0     0     0     0     0     0     0
      0     0     0     0     0     0     0     0     0;100 0     0     0
      0     0     0     0     0     0     0     0     0     100   0     0
      0     0     0     0;100 0     0     100   0     0     0     0     0
      0     0     0     0     100   0     0     0     0     0     0;100 0
      0     0     0     0     0     0     0     0     0     0     0     100
      0     0     0     0     0     0;100 0     0     100   0     0     0
      0     0     0     0     0     0     100   0     0     0     0     0
      0;100 0     0     0     0     0     0     0     0     0     0     0
      0     100   0     0     0     0     0     0;100 100   0     0     0
      0     0     0     0     0     0     0     0     100   0     0     0
      0     0     0;100 100   0     0     0     0     0     0     0     100
      0     0     0     100   0     0     0     0     0     0;100 0     100
      100   0     0     0     0     0     0     0     0     0     100   0
      0     0     0     0     0;100 100   0     0     0     0     0     0
      0     0     0     0     0     100   0     0     0     0     0
      0;100 0     0     100   0     0     0     0     0     0     0     0
      0     100   0     0     0     0     0     0;100 0     0     100   0
      0     0     0     0     0     100   0     0     100   0     0     0
      0     0     0;100 0     0     100   0     0     0     0     0     0
      0     0     0     100   0     0     0     0     0     0;0   0     0
      0     0     0     0     0     0     0     0     0     0     0     0
      0     0     0     0     0;100 0     0     100   0     0     0     0
      0     0     0     100   0     100   0     0     0     0     0
      0;100 0     0     100   0     0     0     0     0     0     0     100
      0     100   100   0     0     0     0     0;100 0     0     100   0
      0     0     0     0     0     0     100   0     100   100   100   0
      0     0     0;100 0     0     0     0     0     0     0     0     0
      0     0     0     100   0     0     0     0     0     0;100 0     0
      0     0     0     0     0     0     0     0     0     0     100   0
      0     0     0     0     0;100 0     0     0     0     0     0     0
      0     0     0     0     0     100   0     0     0     0     0     0];
while (k<=pop)
p=1;
l1=0;
l2=0;
l3=0;
l4=0;
l5=0;
l6=0;
l7=0;
l8=0;
l9=0;
l10=0;
l11=0;
l12=0;
l13=0;
l14=0;
l15=0;
l16=0;
l17=0;
l18=0;
l19=0;
l20=0;
l21=0;
g1=[1 14 2];              %each g represents group of PGM of ANSI 101
g2=[14 1 2];
g3=[1 14 6];
g4=[1 14 2 7];
g5=[1 14 2 10 8];
g6=[1 14 4 3 9];
g7=[1 14 18];
g8=[1 14 19];
```

```
g9=[1 14 4 5];
g10=[14 1 4 13];
g11=[14 1 2 10];
g12=[1 14 4 11 12 15 16 17];
g13=[14 1 20];
g14=[1 14 20];
g15=[1 14];
g16=[14 1 11 12 15 4];
g17=[1 14 11 12 15 4 6 16];
g18=[14 1 11 12 15 4 6 3 16];
g19=[1 14 6 11 12 15 4 6 16 17];
g20=[14 1 18 6 2 11];
g21=[1 14 19 2];

u1=length(g1);
u2=length(g2);
u3=length(g3);
u4=length(g4);
u5=length(g5);
u6=length(g6);
u7=length(g7);
u8=length(g8);
u9=length(g9);
u10=length(g10);
u11=length(g11);
u12=length(g12);
u13=length(g13);
u14=length(g14);
u15=length(g15);
u16=length(g16);
u17=length(g17);
u18=length(g18);
u19=length(g19);
u20=length(g20);
u21=length(g21);

oo=length(g1)+length(g2)+length(g3)+length(g4)+length(g5)+length(g6)+length
(g7)+length(g8)+length(g9)+length(g10)+length(g11)+length(g12)+length(g13)+
length(g14)+length(g15)+length(g16)+length(g17)+length(g18)+length(g19)+len
gth(g20)+length(g21);
while(p<=oo)
array=[g1(1),g2(1),g3(1),g4(1),g5(1),g6(1),g7(1),g8(1),g9(1),g10(1),g11(1),
g12(1),g13(1),g14(1),g15(1),g16(1),g17(1),g18(1),g19(1),g20(1),g21(1)];
d=size(array);
Index=randi(21);
randVal=array(Index);
if(randVal==0)
p=p;
else
new(p)=randVal;
p=p+1;
if randVal==g1(1)
l1=l1+1;
if(l1==u1)
g1(1)=0;
else
g1(1)=[];
end
elseif randVal==g2(1)
l2=l2+1;
if(l2==u2)
```

```
g2(1)=0;
else
g2(1)=[];
end
elseif randVal==g3(1)
l3=l3+1;
if(l3==u3)
g3(1)=0;
else
g3(1)=[];
end
elseif randVal==g4(1)
l4=l4+1;
if(l4==u4)
g4(1)=0;
else
g4(1)=[];
end
elseif randVal==g5(1)
l5=l5+1;
if(l5==u5)
g5(1)=0;
else
g5(1)=[];
end
elseif randVal==g6(1)
l6=l6+1;
if(l6==u6)
g6(1)=0;
else
g6(1)=[];
end
elseif randVal==g7(1)
l7=l7+1;
if(l7==u7)
g7(1)=0;
else
g7(1)=[];
end
elseif randVal==g8(1)
l8=l8+1;
if(l8==u8)
g8(1)=0;
else
g8(1)=[];
end
elseif randVal==g9(1)
l9=l9+1;
if(l9==u9)
g9(1)=0;
else
g9(1)=[];
end
elseif randVal==g10(1)
l10=l10+1;
if(l10==u10)
g10(1)=0;
else
g10(1)=[];
end
elseif randVal==g11(1)
```

```
l11=l11+1;
if(l11==u11)
g11(1)=0;
else
g11(1)=[];
end
elseif randVal==g12(1)
l12=l12+1;
if(l12==u12)
g12(1)=0;
else
g12(1)=[];
end
elseif randVal==g13(1)
l13=l13+1;
if(l13==u13)
g13(1)=0;
else
g13(1)=[];
end
elseif randVal==g14(1)
l14=l14+1;
if(l14==u14)
g14(1)=0;
else
g14(1)=[];
end
elseif randVal==g15(1)
l15=l15+1;
if(l15==u15)
g15(1)=0;
else
g15(1)=[];
end
elseif randVal==g16(1)
l16=l16+1;
if(l16==u16)
g16(1)=0;
else
g16(1)=[];
end
elseif randVal==g17(1)
l17=l17+1;
if(l17==u17)
g17(1)=0;
else
g17(1)=[];
end
elseif randVal==g18(1)
l18=l18+1;
if(l18==u18)
g18(1)=0;
else
g18(1)=[];
end
elseif randVal==g19(1)
l19=l19+1;
if(l19==u19)
g19(1)=0;
else
g19(1)=[];
```

```
end
elseif randVal==g20(1)
l20=l20+1;
if(l20==u20)
g20(1)=0;
else
g20(1)=[];
end
elseif randVal==g21(1)
l21=l21+1;
if(l21==u21)
g21(1)=0;
else
g21(1)=[];
end
else
end
end
[junk,index] = unique(new,'first');
newe=new(sort(index));
newe
end
pp=length(newe);
cost=0;            % Calculation of fitness function for each process plan
setup=0;
penality=0;
for i=1:pp-1
    x=newe(i)
    y=newe(i+1)
    if(cost_m(x,y)==1)
        cost=cost+1
    else
        cost=cost+0;
    end
    if(setup_m(x,y)==1)
        setup=setup+1;
    else
        setup=setup+0;
    end
end
for i=1:pp-1
    x=newe(i)
    for y=newe(i+1:pp)
        if(penality_m(x,y)==100)
            penality=penality+100
        else
            penality=penality+0;
        end
    end
end
cost_array(parent)=cost;
setup_array(parent)=setup;
penality_array(parent)=penality;
parent=parent+1;
saved(e,:)=newe;
e=e+1;
k=k+1;
end
saved
cost_array
setup_array
```

```
penality_array
w1=input('Enter weight for cost function:');
w2=input('Enter weight for setup function:');
for q=1:length(cost_array)
fitness(q)=(w1*cost_array(q))+(w2*setup_array(q))+(penality_array(q));
end
m=1;
fitting=fitness;
n=(max(fitting)+20);
lll=0;
while((lll)<(length(fitting)))
 [r,c] = find(fitting==max(fitting(:)));
 [row,col] = find(fitting==min(fitting(:)));
minimum=col;
maximum=c;
if (length(minimum)==1)
ranks(m)=minimum;
m=m+1;
fitting(col)=n;
elseif(length(minimum>1))
    jj=length(minimum);
    for indi=1:jj
        ranks(m)=minimum(indi);
        m=m+1;
        fitting(col)=n;
    end
else
end
lll=length(ranks);
end
ranks
ooo=length(ranks);
for i=1:ooo
  k=ranks(i);
  ranked(i,:)=saved(k,:);
end
ranked
generations=1;
grap=1;
while(generations<=1000)
[l b]=size(ranked);
num=1;
i=1;
choose=ceil(b/2);
     indec=randi(choose)         %choosing indexes for creation of children
     indes=indec+(choose-1)
while(i<l)
    f=1;
    one=ranked(i,:);
    two=ranked(i+1,:);
     for j=1:b
         if((indec>j)|(j>indes))
             cop(j)=one(j);
         else
             cop(j)=0;
             unoccupied(f)=j;
             check(f)=one(j);
             f=f+1;
         end
     end
     cop
     pos=1;
for t=1:length(check)
    w=check(t);
      h(pos)=find(two==w);
      pos=pos+1;
end
sorted=sort(h);
for u=1:length(unoccupied)
    cop(unoccupied(u))=two(sorted(u));
end
child(num,:)=cop;
num=num+1;
    i=i+2;
end

i=1;
while(i<l)
    f=1;
    one=ranked(i,:);
    two=ranked(i+1,:);
     for j=1:b
         if((indec>j)|(j>indes))
             cop(j)=two(j);
         else
             cop(j)=0;
             unoccupied(f)=j;
             check(f)=two(j);
             f=f+1;
         end
     end
     cop
     pos=1;
for t=1:length(check)
    w=check(t);
      h(pos)=find(one==w);
      pos=pos+1;
end
sorted=sort(h);
for u=1:length(unoccupied)
    cop(unoccupied(u))=one(sorted(u));
end
child(num,:)=cop;
num=num+1;
    i=i+2;
end

child
[m g]=size(child);
for k=1:m
    ch=randi(100);
    if(ch>20)
        finalized(k,:)=child(k,:);

    else
        temporary=child(k,:)
      pep1=randi(g);
      pep2=randi(g);
      while(pep2==pep1)
          pep2=randi(g);
      end
```

```
            temp=temporary(pep1);
            temporary(pep1)=temporary(pep2);
            temporary(pep2)=temp;

            finalized(k,:)=temporary;
        end
    end
finalized
[c d]=size(finalized);
child1=1;
t=1;
while(t<=c)
cost1=0;
setup1=0;
penality1=0;
j=finalized(t,:);
j1=length(j);
for p=1:(j1-1)
    x1=j(p);
    y1=j(p+1);
    if(cost_m(x1,y1)==1)
        cost1=cost1+1;
    else
        cost1=cost1+0;
    end
    if(setup_m(x1,y1)==1)
        setup1=setup1+1;
    else
        setup1=setup1+0;
    end
end
for p=1:(j1-1)
    x1=j(p);
    for y1=j(p+1:j1)
        if(penality_m(x1,y1)==100)
            penality1=penality1+100
        else
            penality1=penality1+0;
        end
    end
end
cost_array1(child1)=cost1;
setup_array1(child1)=setup1;
penality_array1(child1)=penality1;
child1=child1+1;
t=t+1;
end

cost_array1
setup_array1
penality_array1
r=1;
combo_saved=ranked;
[m n]=size(ranked);
[o p]=size(finalized);
for u=(m+1):(2*o)
    combo_saved(u,:)=finalized(r,:);
    r=r+1;
end
combo_saved
[l b]=size(ranked);
```

```
position=1;
for t=1:l
cost=0;
setup=0;
penality=0;
g=ranked(t,:);
for i=1:b-1
    x=g(i);
    y=g(i+1);
    if(cost_m(x,y)==1)
        cost=cost+1;
    else
        cost=cost+0;
    end
    if(setup_m(x,y)==1)
        setup=setup+1;
    else
        setup=setup+0;
    end
end
for i=1:b-1
    x=g(i);
    for y=g(i+1:b)
    if(penality_m(x,y)==100)
        penality=penality+100;
    else
        penality=penality+0;
    end
    end
end
cost_array(position)=cost;
setup_array(position)=setup;
penality_array(position)=penality;
position=position+1;
end
for q=1:length(cost_array)
combo_fitness(q)=(w1*cost_array(q))+(w2*setup_array(q))+(penality_array(q))
;
end
combo_fitness
 v=1;
 for q=(length(cost_array)+1):(2*length(cost_array1))

combo_fitness(q)=(w1*cost_array1(v))+(w2*setup_array1(v))+penality_array1(v
);
 v=v+1;
 end

combo_fitness
m=1;
combo_fitting=combo_fitness;
n=(max(combo_fitting)+400);
lll=0;
while((lll)<(length(combo_fitting)))
 [r,c] = find(combo_fitting==max(combo_fitting(:)));
 [row,col] = find(combo_fitting==min(combo_fitting(:)));
mini=col;
maxi=c;
if (length(mini)==1)
combo_ranks(m)=mini;
m=m+1;
```

```
combo_fitting(col)=n;
elseif(length(mini>1))
    jj=length(mini);
    for indi=1:jj
        combo_ranks(m)=mini(indi);
        m=m+1;
        combo_fitting(col)=n;
    end
else
end
lll=length(combo_ranks);
end
combo_ranks
ooo=length(combo_ranks);
for i=1:ooo
  k=combo_ranks(i);
  combo_ranked(i,:)=combo_saved(k,:);
end
combo_ranked
[l b]=size(combo_ranked);
nub=l/2;
for p=1:nub
    chosen(p,:)=combo_ranked(p,:);
end
chosen
ranked=chosen;
[l b]=size(ranked);
parent=1;
gener=chosen(1,:);
cost=0;
setup=0;
penality=0;
for i=1:(length(gener)-1)

    x=gener(i);
    y=gener(i+1);
    if(cost_m(x,y)==1)
        cost=cost+1;
    else
        cost=cost+0;
    end
    if(setup_m(x,y)==1)
        setup=setup+1;
    else
        setup=setup+0;
    end
end
    for i=1:b-1
    x=gener(i);
    for y=gener(i+1:b)
    if(penality_m(x,y)==100)
        penality=penality+100;
    else
        penality=penality+0;
    end
    end
end
cost_gener=cost;
setup_gener=setup;
penality_gener=penality;
fitness_gener=w1*cost_gener+w2*setup_gener+penality_gener;
```

```
fitness_graph(grap)=fitness_gener;
grap=grap+1;
generations=generations+1;
fitness_graph
end
chosen
[l b]=size(chosen);
parent=1;
for t=1:l
cost=0;
setup=0;
penality=0;
k=chosen(t,:);
for i=1:b-1
    x=k(i);
    y=k(i+1);
    if(cost_m(x,y)==1)
        cost=cost+1;
    else
        cost=cost+0;
    end
    if(setup_m(x,y)==1)
        setup=setup+1;
    else
        setup=setup+0;
    end
    end
for i=1:b-1
    x=k(i);
    for y=k(i+1:b);
        if(penality_m(x,y)==100)
        penality=penality+100;
    else
        penality=penality+0;
        end
    end
end

cost_array(parent)=cost;
setup_array(parent)=setup;
penality_array(parent)=penality;
parent=parent+1;
end
for q=1:length(cost_array)
fitness(q)=(w1*cost_array(q))+(w2*setup_array(q))+(penality_array(q));
end
fitness
tt=1:1000;
plot(tt,fitness_graph)
fitness
```

Appendix B

This Appendix contains MATLab Code in CS methodology for ANSI 101 part.

MATLab Code : ANSI 101 BY CS methodology

```
function []=ansiga()
k=1;
e=1;
pop=6;
parent=1;
cost_m=[0 0 1 1 1 1 1 1 1 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0;0 0
1 0 1 1 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0;0 1 1 0 1 1 1 1
1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1;1 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0
1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1
1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0
0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1
1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1
0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0];
setup_m=[0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0
1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1
1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 0 1 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0
1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1
1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0
0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1
1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1
0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0];
penality_m=[0     0     0     0     0     0     0     0     0     0
      0     0     0     0     0     0     0     0     0;100 0     0     0
      0     0     0     0     0     0     0     0     0     100   0     0
      0     0     0     0;100 0     0     100   0     0     0     0     0
      0     0     0     0     100   0     0     0     0     0     0;100 0
      0     0     0     0     0     0     0     0     0     0     0
      100   0     0     0     0     0     0;100 0     0     100   0     0
      0     0     0     0     0     0     0     100   0     0     0     0
      0     0;100 0     0     0     0     0     0     0     0     0     0
      0     0     100   0     0     0     0     0     0;100 100   0     0
      0     0     0     0     0     0     0     0     0     100   0     0
      0     0     0     0;100 100   0     0     0     0     0     0     0
      100   0     0     0     100   0     0     0     0     0     0;100 0
      100   100   0     0     0     0     0     0     0     0     0
      100   0     0     0     0     0     0;100 100   0     0     0     0
      0     0     0     0     0     0     0     100   0     0     0     0
      0     0;100 0     0     100   0     0     0     0     0     0     0
      0     0     100   0     0     0     0     0     0;100 0     0
      100   0     0     0     0     0     0     100   0     0     100   0
      0     0     0     0     0;100 0     0     100   0     0     0     0
      0     0     0     0     0     100   0     0     0     0     0
      0;0   0     0     0     0     0     0     0     0     0     0     0
      0     0     0     0     0     0     0     0;100 0     0     100   0
      0     0     0     0     0     0     100   0     100   0     0     0
      0     0     0;100 0     0     100   0     0     0     0     0     0
      0     100   0     100   100   0     0     0     0     0;100 0     0
      100   0     0     0     0     0     0     0     100   0     100
      100   100   0     0     0     0;100 0     0     0     0     0     0
      0     0     0     0     0     0     100   0     0     0     0     0
      0;100 0     0     0     0     0     0     0     0     0     0     0
      0     100   0     0     0     0     0     0;100 0     0     0     0
      0     0     0     0     0     0     0     0     100   0     0     0
      0     0     0];
while (k<=pop)
p=1;
l1=0;
l2=0;
l3=0;
l4=0;
l5=0;
l6=0;
l7=0;
l8=0;
l9=0;
l10=0;
l11=0;
l12=0;
l13=0;
l14=0;
l15=0;
l16=0;
l17=0;
```

```
l18=0;
l19=0;
l20=0;
l21=0;
g1=[1 14 2];
g2=[14 1 2];
g3=[1 14 6];
g4=[1 14 2 7];
g5=[1 14 2 10 8];
g6=[1 14 4 3 9];
g7=[1 14 18];
g8=[1 14 19];
g9=[1 14 4 5];
g10=[14 1 4 13];
g11=[14 1 2 10];
g12=[1 14 4 11 12 15 16 17];
g13=[14 1 20];
g14=[1 14 20];
g15=[1 14];
g16=[14 1 11 12 15 4];
g17=[1 14 11 12 15 4 6 16];
g18=[14 1 11 12 15 4 6 3 16];
g19=[1 14 6 11 12 15 4 6 16 17];
g20=[14 1 18 6 2 11];
g21=[1 14 19 2];

u1=length(g1);
u2=length(g2);
u3=length(g3);
u4=length(g4);
u5=length(g5);
u6=length(g6);
u7=length(g7);
u8=length(g8);
u9=length(g9);
u10=length(g10);
u11=length(g11);
u12=length(g12);
u13=length(g13);
u14=length(g14);
u15=length(g15);
u16=length(g16);
u17=length(g17);
u18=length(g18);
u19=length(g19);
u20=length(g20);
u21=length(g21);

oo=length(g1)+length(g2)+length(g3)+length(g4)+length(g5)+length(g6)
+length(g7)+length(g8)+length(g9)+length(g10)+length(g11)+length(g12
)+length(g13)+length(g14)+length(g15)+length(g16)+length(g17)+length
(g18)+length(g19)+length(g20)+length(g21);
while(p<=oo)
array=[g1(1),g2(1),g3(1),g4(1),g5(1),g6(1),g7(1),g8(1),g9(1),g10(1),
g11(1),g12(1),g13(1),g14(1),g15(1),g16(1),g17(1),g18(1),g19(1),g20(1
),g21(1)];
```

```
d=size(array);
Index=randi(21);
randVal=array(Index);
if(randVal==0)
p=p;
else
new(p)=randVal;
p=p+1;
if randVal==g1(1)
l1=l1+1;
if(l1==u1)
g1(1)=0;
else
g1(1)=[];
end
elseif randVal==g2(1)
l2=l2+1;
if(l2==u2)
g2(1)=0;
else
g2(1)=[];
end
elseif randVal==g3(1)
l3=l3+1;
if(l3==u3)
g3(1)=0;
else
g3(1)=[];
end
elseif randVal==g4(1)
l4=l4+1;
if(l4==u4)
g4(1)=0;
else
g4(1)=[];
end
elseif randVal==g5(1)
l5=l5+1;
if(l5==u5)
g5(1)=0;
else
g5(1)=[];
end
elseif randVal==g6(1)
l6=l6+1;
if(l6==u6)
g6(1)=0;
else
g6(1)=[];
end
elseif randVal==g7(1)
l7=l7+1;
if(l7==u7)
g7(1)=0;
else
g7(1)=[];
```

```
end
elseif randVal==g8(1)
l8=l8+1;
if(l8==u8)
g8(1)=0;
else
g8(1)=[];
end
elseif randVal==g9(1)
l9=l9+1;
if(l9==u9)
g9(1)=0;
else
g9(1)=[];
end
elseif randVal==g10(1)
l10=l10+1;
if(l10==u10)
g10(1)=0;
else
g10(1)=[];
end
elseif randVal==g11(1)
l11=l11+1;
if(l11==u11)
g11(1)=0;
else
g11(1)=[];
end
elseif randVal==g12(1)
l12=l12+1;
if(l12==u12)
g12(1)=0;
else
g12(1)=[];
end
elseif randVal==g13(1)
l13=l13+1;
if(l13==u13)
g13(1)=0;
else
g13(1)=[];
end
elseif randVal==g14(1)
l14=l14+1;
if(l14==u14)
g14(1)=0;
else
g14(1)=[];
end
elseif randVal==g15(1)
l15=l15+1;
if(l15==u15)
g15(1)=0;
else
g15(1)=[];
```

```
end
elseif randVal==g16(1)
l16=l16+1;
if(l16==u16)
g16(1)=0;
else
g16(1)=[];
end
elseif randVal==g17(1)
l17=l17+1;
if(l17==u17)
g17(1)=0;
else
g17(1)=[];
end
elseif randVal==g18(1)
l18=l18+1;
if(l18==u18)
g18(1)=0;
else
g18(1)=[];
end
elseif randVal==g19(1)
l19=l19+1;
if(l19==u19)
g19(1)=0;
else
g19(1)=[];
end
elseif randVal==g20(1)
l20=l20+1;
if(l20==u20)
g20(1)=0;
else
g20(1)=[];
end
elseif randVal==g21(1)
l21=l21+1;
if(l21==u21)
g21(1)=0;
else
g21(1)=[];
end
else
end
end
[junk,index] = unique(new,'first');
newe=new(sort(index));
newe
end
pp=length(newe);
cost=0;
setup=0;
penality=0;
for i=1:pp-1
    x=newe(i)
    y=newe(i+1)
    if(cost_m(x,y)==1)
        cost=cost+1
    else
        cost=cost+0;
    end
    if(setup_m(x,y)==1)
        setup=setup+1;
    else
        setup=setup+0;
    end
end
for i=1:pp-1
    x=newe(i)
    for y=newe(i+1:pp)
        if(penality_m(x,y)==100)
            penality=penality+100
        else
            penality=penality+0;
        end
    end
end
cost_array(parent)=cost;
setup_array(parent)=setup;
penality_array(parent)=penality;
parent=parent+1;
saved(e,:)=newe;
e=e+1;
k=k+1;
end
saved
cost_array
setup_array
penality_array
w1=input('Enter weight for cost function:');
w2=input('Enter weight for setup function:');
for q=1:length(cost_array)
fitness(q)=(w1*cost_array(q))+(w2*setup_array(q))+(penality_array(q)
);
end
fitness
m=1;
fitting=fitness;
n=(max(fitting)+20);
lll=0;
while((lll)<(length(fitting)))
 [r,c] = find(fitting==max(fitting(:)));
 [row,col] = find(fitting==min(fitting(:)));
minimum=col;
maximum=c;
if (length(minimum)==1)
ranks(m)=minimum;
m=m+1;
fitting(col)=n;
elseif(length(minimum>1))
    jj=length(minimum);
    for indi=1:jj
        ranks(m)=minimum(indi);
        m=m+1;
        fitting(col)=n;
    end
else
end
lll=length(ranks);
end
ranks
ooo=length(ranks);
for i=1:ooo
  k=ranks(i);
  ranked(i,:)=saved(k,:);
end
ranked
generations=1;
grap=1;
while(generations<=10)
[l b]=size(ranked);
num=1;
i=1;
[l b]=size(ranked);
num=1;
i=0;
[m g]=size(fitness);
for k=1:m
    beta=3/2;
sigma=(gamma(1+beta)*sin(pi*beta/2)/(gamma((1+beta)/2)*beta*2^((beta
-1)/2)))^(1/beta);
u=randn(size(fitness))*sigma;
    v=randn(size(fitness));
    step=u./abs(v).^(1/beta);
step1=100000*step;
b1=round(step1)
end

b=zeros(pop,5);
for i=1:10,
    for j=1:10,
        for k=1:pop,

      if i==1 && j==1
          b(k,i) = round(b1(k)/1e4);

      end
      if i==2 && j==2
              b(k,i)=round(rem(b1(k),1e4)/1e3);

      end
      if i==3 && j==3
           b(k,i)=round(rem(rem(b1(k),1e4),1e3)/1e2);

      end
      if i==4 && j==4
                 b(k,i)=round(rem(rem(rem(b1(k),1e4),1e3),1e2)/1e1);
      end
      if i==5 && j==5

b(k,i)=round(rem(rem(rem(rem(b1(k),1e4),1e3),1e2),1e1)/1e0);

      end
          end
     end
end

b
 finalized =zeros(pop,20);
 %finalized
ind_a=numel(ranked);
ind_b=numel(b);


 for i=1:ind_a
    finalized(i)=ranked(i);
    if i>(ind_a-ind_b)
        finalized(i)=ranked(i)+b(i-(ind_a-ind_b));
    end

 for i=(ind_a-ind_b+1):ind_a
    if finalized(i)>max(ranked)
        finalized(i)=ranked(i);
    end
 end
 for i=(ind_a-ind_b+1):ind_a
    if finalized(i)<=0
        finalized(i)=ranked(i);
    end
 end
 ranked
 finalized
 [g mon]=size(finalized)
for k=1:50
for i=1:g
checker=finalized(i,:);
ranki=ranked(i,:);
for j=1:mon-1
    for l=2:mon
        if (checker(j)==checker(l) && j~=l)
            checker(j)=ranki(l);
            checker(l)=ranki(j);
        end
    end
end
finalized(i,:)=checker;
end
end
 end
finalized
[c d]=size(finalized);
child1=1;
```

```
t=1;
while(t<=c)
cost1=0;
setup1=0;
penality1=0;
j=finalized(t,:);
j1=length(j);
for p=1:(j1-1)
    x1=j(p);
    y1=j(p+1);
    if(cost_m(x1,y1)==1)
        cost1=cost1+1;
    else
        cost1=cost1+0;
    end
    if(setup_m(x1,y1)==1)
        setup1=setup1+1;
    else
        setup1=setup1+0;
    end
end
for p=1:(j1-1)
    x1=j(p);
    for y1=j(p+1:j1)
        if(penality_m(x1,y1)==100)
            penality1=penality1+100
        else
            penality1=penality1+0;
        end
    end
end
cost_array1(child1)=cost1;
setup_array1(child1)=setup1;
penality_array1(child1)=penality1;
child1=child1+1;
t=t+1;
end

cost_array1
setup_array1
penality_array1
r=1;
combo_saved=ranked;
[m n]=size(ranked);
[o p]=size(finalized);
for u=(m+1):(2*o)
    combo_saved(u,:)=finalized(r,:);
    r=r+1;
end
combo_saved
[l b]=size(ranked);
position=1;
for t=1:l
cost=0;
setup=0;
penality=0;
```

```
g=ranked(t,:);
for i=1:b-1
    x=g(i);
    y=g(i+1);
    if(cost_m(x,y)==1)
        cost=cost+1;
    else
        cost=cost+0;
    end
    if(setup_m(x,y)==1)
        setup=setup+1;
    else
        setup=setup+0;
    end
end
for i=1:b-1
    x=g(i);
    for y=g(i+1:b)
    if(penality_m(x,y)==100)
        penality=penality+100;
    else
        penality=penality+0;
    end
    end
end
cost_array(position)=cost;
setup_array(position)=setup;
penality_array(position)=penality;
position=position+1;
end
for q=1:length(cost_array)
combo_fitness(q)=(w1*cost_array(q))+(w2*setup_array(q))+(penality_ar
ray(q));
end
combo_fitness
 v=1;
 for q=(length(cost_array)+1):(2*length(cost_array1))

combo_fitness(q)=(w1*cost_array1(v))+(w2*setup_array1(v))+penality_a
rray1(v);
 v=v+1;
 end

combo_fitness
m=1;
combo_fitting=combo_fitness;
n=(max(combo_fitting)+400);
lll=0;
while((lll)<(length(combo_fitting)))
 [r,c] = find(combo_fitting==max(combo_fitting(:)));
 [row,col] = find(combo_fitting==min(combo_fitting(:)));
mini=col;
maxi=c;
if (length(mini)==1)
combo_ranks(m)=mini;
m=m+1;
```

```
combo_fitting(col)=n;
elseif(length(mini>1))
    jj=length(mini);
    for indi=1:jj
        combo_ranks(m)=mini(indi);
        m=m+1;
        combo_fitting(col)=n;
    end
else
end
lll=length(combo_ranks);
end
combo_ranks
ooo=length(combo_ranks);
for i=1:ooo
  k=combo_ranks(i);
  combo_ranked(i,:)=combo_saved(k,:);
end
combo_ranked
[l b]=size(combo_ranked);
nub=l/2;
for p=1:nub
    chosen(p,:)=combo_ranked(p,:);
end
chosen
ranked=chosen;
[l b]=size(ranked);
parent=1;
gener=chosen(1,:);
cost=0;
setup=0;
penality=0;
for i=1:(length(gener)-1)
    x=gener(i);
    y=gener(i+1);
    if(cost_m(x,y)==1)
        cost=cost+1;
    else
        cost=cost+0;
    end
    if(setup_m(x,y)==1)
        setup=setup+1;
    else
        setup=setup+0;
    end

end
for i=1:b-1
    x=gener(i);
    for y=gener(i+1:b)
    if(penality_m(x,y)==100)
        penality=penality+100;
    else
        penality=penality+0;
    end
    end
```

```
end
cost_gener=cost;
setup_gener=setup;
penality_gener=penality;
fitness_gener=w1*cost_gener+w2*setup_gener+penality;
fitness_graph(grap)=fitness_gener;
grap=grap+1;
generations=generations+1;
fitness_graph
end
chosen
[l b]=size(chosen);
parent=1;
for t=1:l
cost=0;
setup=0;
penality=0;
k=chosen(t,:);
for i=1:b-1
    x=k(i);
    y=k(i+1);
    if(cost_m(x,y)==1)
        cost=cost+1;
    else
        cost=cost+0;
    end
    if(setup_m(x,y)==1)
        setup=setup+1;
    else
        setup=setup+0;
    end
    end
for i=1:b-1
    x=k(i);
    for y=k(i+1:b);
        if(penality_m(x,y)==100)
        penality=penality+100;
    else
        penality=penality+0;
        end
    end
end

cost_array(parent)=cost;
setup_array(parent)=setup;
penality_array(parent)=penality;
parent=parent+1;
end
for q=1:length(cost_array)
fitness(q)=(w1*cost_array(q))+(w2*setup_array(q))+(penality_array(q)
);
end
fitness
tt=1:10;
plot(tt,fitness_graph)
end
```

Appendix C

This Appendix contains MATLab Code in GA methodology for the part in figure 33.

MATLab Code : For example part in figure 33 BY CS methodology

```
function []=comparison()
k=1;
e=1;
pop=2;
parent=1;
cost_m=[0  0    0    0    0    0    0    0    0    0    0    0
      0    1    1    1    1    0    0    0    0    1    1    1
      1    1    1    1;0  0    0    0    0    0    0    0    0
      0    0    0    0    1    1    1    1    0    0    0    0
      1    1    1    1    1    1    1;0  0    0    0    0    0
      0    0    0    0    0    0    0    1    1    1    1    0
      0    0    0    1    1    1    1    1    1    1;0  0    0
      0    0    0    0    0    0    0    0    0    0    1    1
      1    1    0    0    0    0    1    1    1    1    1    1
      1;0  0    0    0    0    0    0    0    0    0    0    0
      0    1    1    1    1    0    0    0    0    1    1    1
      1    1    1    1;0  0    0    0    0    0    0    0    0
      0    0    0    0    1    1    1    1    0    0    0    0
      1    1    1    1    1    1    1;0  0    0    0    0    0
      0    0    0    0    0    0    0    1    1    1    1    0
      0    0    0    1    1    1    1    1    1    1;0  0    0
      0    0    0    0    0    0    0    0    0    0    1    1
      1    1    0    0    0    0    1    1    1    1    1    1
      1;0  0    0    0    0    0    0    0    0    0    0    0
      0    1    1    1    1    0    0    0    0    1    1    1
      1    1    1    1;0  0    0    0    0    0    0    0    0
      0    0    0    0    1    1    1    1    0    0    0    0
      1    1    1    1    1    1    1;0  0    0    0    0    0
      0    0    0    0    0    0    0    1    1    1    1    0
      0    0    0    1    1    1    1    1    1    1;0  0    0
      0    0    0    0    0    0    0    0    0    0    1    1
      1    1    0    0    0    0    1    1    1    1    1    1
      1;0  0    0    0    0    0    0    0    0    0    0    0
      0    1    1    1    1    0    0    0    0    1    1    1
      1    1    1    1;1  1    1    1    1    1    1    1    1
      1    1    1    1    0    0    0    0    1    1    1    1
      0    1    0    1    1    0    1;1  1    1    1    1    1
      1    1    1    1    1    1    1    0    0    0    0    1
      1    1    1    0    1    0    1    1    0    1;1  1    1
      1    1    1    1    1    1    1    1    1    1    0    0
      0    0    1    1    1    1    0    1    0    1    1    0
      1;1  1    1    1    1    1    1    1    1    1    1    1
      1    0    0    0    0    1    1    1    1    0    1    0
      1    1    0    1;0  0    0    0    0    0    0    0    0
      0    0    0    0    1    1    1    1    0    0    0    0
      1    1    1    1    1    1    1;0  0    0    0    0    0
      0    0    0    0    0    0    0    1    1    1    1    0
```

```
         0     0     0     1     1     1     1     1     1     1;0   0     0
         0     0     0     0     0     0     0     0     0     0     1     1
         1     1     0     0     0     0     1     1     1     1     1     1
         1;0   0     0     0     0     0     0     0     0     0     0     0
         0     1     1     1     1     0     0     0     0     1     1     1
         1     1     1     1;1   1     1     1     1     1     1     1     1
         1     1     1     1     1     1     1     1     1     1     1     1
         0     1     0     1     0     1     1;1   1     1     1     1     1
         1     1     1     1     1     1     1     0     0     0     0     1
         1     1     1     0     1     0     1     1     0     1;1   1     1
         1     1     1     1     1     1     1     1     1     1     1     1
         1     1     1     1     1     1     0     1     0     1     0     1
         1;1   1     1     1     1     1     1     1     1     1     1     1
         1     0     0     0     0     1     1     1     1     0     1     0
         1     1     0     1;1   1     1     1     1     1     1     1     1
         1     1     1     1     1     1     1     1     1     1     1     1
         0     1     0     1     0     1     1;1   1     1     1     1     1
         1     1     0     0     1     1     1     0     0     0     0     1
         1     1     0     0     0     0     1     1     0     0;1   1     1
         1     1     1     1     1     1     1     1     1     1     1     1
         1     1     1     1     1     1     1     1     1     1     1     1
         0];
setup_m=[0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0
1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1
1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0
1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1
1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0
0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1
1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1
0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0];
penality_m=[0    0     0     0     0     0     0     0     0     0
         100   0     0     0     0     0     0     0     0     0     0     0
         0     0     0     0     0     0;0   0     0     0     0     0     0
         0     0     0     100   0     0     0     0     0     0     0     0
         0     0     0     0     0     0     0     0     0;0   0     0     0
         0     0     0     0     0     0     100   0     0     0     0     0
         0     0     0     0     0     0     0     0     0     0     0
         0;0   0     0     0     0     0     0     0     0     0     100   0
         0     0     0     0     0     100   0     0     0     0     0     0
         0     0     0     0;100 0     0     0     0     0     0     0     0
```

```
      0     100   0     0     0     0     0     0     0     100   0     0
      0     0     0     0     0     0     0;0   100   0     0     0     0
      0     0     0     0     100   0     0     0     0     0     0     0
      0     0     0     0     0     0     0     0     0     0;0   0
      100   0     0     0     0     0     0     0     100   0     0     0
      0     0     0     0     0     100   0     0     0     0     0     0
      0     0;0   0     0     100   0     0     0     0     0     0
      100   0     0     0     0     0     0     0     0     0     0     0
      0     0     0     0     0     0;0   0     0     0     0     0     0
      0     0     100   100   0     0     0     0     0     0     0     0
      0     0     0     0     0     0     100   0     0;0   0     0     0
      100   100   100   100   0     0     100   0     0     0     0     0
      0     0     100   100   0     0     0     0     0     0     0
      0;0   0     0     0     0     0     0     0     0     0     0     0
      0     0     0     0     0     0     0     0     0     0     0     0
      0     0     0     0;0   0     0     0     0     0     0     0     0
      0     100   100   0     0     0     0     0     0     0     0     0
      0     0     0     0     0     0     0;0   0     0     100   0     0
      0     0     0     0     100   0     0     0     0     0     0     0
      0     0     0     0     0     0     0     0     0     0;0   0     0
      0     0     0     0     0     0     100   100   0     0     0     0
      0     0     0     0     0     0     0     0     0     0     0     0
      0;0   0     0     0     0     0     0     0     0     100   100   0
      0     0     0     0     0     0     0     0     0     0     0     0
      0     0     0     0;0   0     0     0     0     0     0     0     0
      100   100   0     0     0     0     0     0     0     0     0     0
      0     0     0     0     0     0     0;0   0     0     0     0     0
      0     0     0     100   100   0     0     0     0     0     0     0
      0     0     0     0     0     0     0     0     0     0;0   0     0
      0     0     100   0     0     0     0     100   0     0     0     0
      0     0     0     0     0     0     0     0     0     0     0     0
      0;0   0     0     0     0     0     0     0     0     0     100   0
      0     0     0     0     0     0     0     0     0     0     0     0
      0     0     0     0;0   0     0     0     0     0     0     0     0
      0     100   0     0     0     0     0     0     0     0     0     0
      0     0     0     0     0     0     0;0   0     0     0     0     0
      0     0     100   0     100   0     0     0     0     0     0     0
      0     0     0     0     0     0     0     0     0     0;0   0     0
      0     0     0     0     0     0     0     100   0     0     0     0
      0     0     0     0     0     0     0     100   0     0     0     0
      0;0   0     0     0     0     0     0     0     0     0     100   0
      0     0     0     0     0     0     0     0     100   0     0     0
      0     0     0     0;0   0     0     0     0     0     0     0
      100   0     100   0     0     0     0     0     0     0     0     0
      0     0     0     0     100   0     0     0;0   0     0     0     0
      0     0     0     0     0     100   0     0     0     0     0     0
      0     0     0     0     0     0     0     0     0     0     0;0   0
      0     0     0     0     0     0     0     0     100   0     0     0
      0     0     0     0     0     0     0     0     0     0     0     0
      0     0;0   0     0     0     0     0     0     0     100   0
      100   0     0     0     0     0     0     0     0     0     0     0
      0     0     0     0     0     0;0   0     0     0     0     0     0
      0     100   0     100   0     0     0     0     0     0     0     0
      0     0     0     0     0     0     0     0     0];
while (k<=pop)
p=1;
```

```
l1=0;
l2=0;
l3=0;
l4=0;
l5=0;
l6=0;
l7=0;
l8=0;
l9=0;
l10=0;
l11=0;
l12=0;
l13=0;
l14=0;
l15=0;
l16=0;
l17=0;
l18=0;
l19=0;
l20=0;
l21=0;
g1=[11 1];
g2=[11 2];
g3=[11 3];
g4=[11 2 6 18 4 8];
g5=[11 1 19 5];
g6=[11 3 20 7];
g7=[11 25 26];
g8=[11 2 6 18 4 12];
g9=[11 2 6 18 4 13];
g10=[11 19 1 5];
g11=[11 1 2 25 3 19 26 5 20 7 6 18 4 8 10 9 21 27 28];
g12=[11 1 2 25 3 19 26 5 20 7 6 18 4 8 10 9 21];
g13=[11 25 26 1 3 2 19 5 20 7 6 18 4 8 10];
g14=[11 25 26 1 2 3 19 5 20 6 7 18 4 8 10 9 24];
g15=[11 1 2 25 3 19 26 5 20 7 6 18 4 8 10 9 21 23 22 27 24];
g16=[11 2 25 1 3 26 19 5 20 6 18 7 4 8 10 9 27 21 24 23 22];
g17=[11 2 25 1 26 19 5 20 3 6 7 18 4 8 10 16];
g18=[11 1 3 2 25 26 19 5 6 20 7 18 4 8 10 17];
g19=[11 3 1 25 26 20 19 18 8 4 5 7 10 14];
g20=[11 3 1 25 26 20 19 18 8 4 7 10 15];
g21=[11 1 2 25 3 19 26 5 20 7 6 18 4 8 10 9 21];

u1=length(g1);
u2=length(g2);
u3=length(g3);
u4=length(g4);
u5=length(g5);
u6=length(g6);
u7=length(g7);
u8=length(g8);
u9=length(g9);
u10=length(g10);
u11=length(g11);
u12=length(g12);
u13=length(g13);
```

```
u14=length(g14);
u15=length(g15);
u16=length(g16);
u17=length(g17);
u18=length(g18);
u19=length(g19);
u20=length(g20);
u21=length(g21);

oo=length(g1)+length(g2)+length(g3)+length(g4)+length(g5)+length(g6)
+length(g7)+length(g8)+length(g9)+length(g10)+length(g11)+length(g12
)+length(g13)+length(g14)+length(g15)+length(g16)+length(g17)+length
(g18)+length(g19)+length(g20)+length(g21);
while(p<=oo)
array=[g1(1),g2(1),g3(1),g4(1),g5(1),g6(1),g7(1),g8(1),g9(1),g10(1),
g11(1),g12(1),g13(1),g14(1),g15(1),g16(1),g17(1),g18(1),g19(1),g20(1
),g21(1)];
d=size(array);
Index=randi(21);
randVal=array(Index);
if(randVal==0)
p=p;
else
new(p)=randVal;
p=p+1;
if randVal==g1(1)
l1=l1+1;
if(l1==u1)
g1(1)=0;
else
g1(1)=[];
end
elseif randVal==g2(1)
l2=l2+1;
if(l2==u2)
g2(1)=0;
else
g2(1)=[];
end
elseif randVal==g3(1)
l3=l3+1;
if(l3==u3)
g3(1)=0;
else
g3(1)=[];
end
elseif randVal==g4(1)
l4=l4+1;
if(l4==u4)
g4(1)=0;
else
g4(1)=[];
end
elseif randVal==g5(1)
l5=l5+1;
if(l5==u5)
```

```
g5(1)=0;
else
g5(1)=[];
end
elseif randVal==g6(1)
l6=l6+1;
if(l6==u6)
g6(1)=0;
else
g6(1)=[];
end
elseif randVal==g7(1)
l7=l7+1;
if(l7==u7)
g7(1)=0;
else
g7(1)=[];
end
elseif randVal==g8(1)
l8=l8+1;
if(l8==u8)
g8(1)=0;
else
g8(1)=[];
end
elseif randVal==g9(1)
l9=l9+1;
if(l9==u9)
g9(1)=0;
else
g9(1)=[];
end
elseif randVal==g10(1)
l10=l10+1;
if(l10==u10)
g10(1)=0;
else
g10(1)=[];
end
elseif randVal==g11(1)
l11=l11+1;
if(l11==u11)
g11(1)=0;
else
g11(1)=[];
end
elseif randVal==g12(1)
l12=l12+1;
if(l12==u12)
g12(1)=0;
else
g12(1)=[];
end
elseif randVal==g13(1)
l13=l13+1;
if(l13==u13)
```

```
g13(1)=0;
else
g13(1)=[];
end
elseif randVal==g14(1)
l14=l14+1;
if(l14==u14)
g14(1)=0;
else
g14(1)=[];
end
elseif randVal==g15(1)
l15=l15+1;
if(l15==u15)
g15(1)=0;
else
g15(1)=[];
end
elseif randVal==g16(1)
l16=l16+1;
if(l16==u16)
g16(1)=0;
else
g16(1)=[];
end
elseif randVal==g17(1)
l17=l17+1;
if(l17==u17)
g17(1)=0;
else
g17(1)=[];
end
elseif randVal==g18(1)
l18=l18+1;
if(l18==u18)
g18(1)=0;
else
g18(1)=[];
end
elseif randVal==g19(1)
l19=l19+1;
if(l19==u19)
g19(1)=0;
else
g19(1)=[];
end
elseif randVal==g20(1)
l20=l20+1;
if(l20==u20)
g20(1)=0;
else
g20(1)=[];
end
elseif randVal==g21(1)
l21=l21+1;
if(l21==u21)
```

```
g21(1)=0;
else
g21(1)=[];
end
else
end
end
[junk,index] = unique(new,'first');
newe=new(sort(index));
newe
end
pp=length(newe);
cost=0;
setup=0;
penality=0;
for i=1:pp-1
    x=newe(i)
    y=newe(i+1)
    if(cost_m(x,y)==1)
        cost=cost+1
    else
        cost=cost+0;
    end
    if(setup_m(x,y)==1)
        setup=setup+1;
    else
        setup=setup+0;
    end
end
for i=1:pp-1
    x=newe(i)
    for y=newe(i+1:pp)
        if(penality_m(x,y)==100)
            penality=penality+100
        else
            penality=penality+0;
        end
    end
end
cost_array(parent)=cost;
setup_array(parent)=setup;
penality_array(parent)=penality;
parent=parent+1;
saved(e,:)=newe;
e=e+1;
k=k+1;
end
saved
cost_array
setup_array
penality_array
w1=input('Enter weight for cost function:');
w2=input('Enter weight for setup function:');
for q=1:length(cost_array)
fitness(q)=(w1*cost_array(q))+(w2*setup_array(q))+(penality_array(q)
);
```

```
end
m=1;
fitting=fitness;
n=(max(fitting)+20);
lll=0;
while((lll)<(length(fitting)))
 [r,c] = find(fitting==max(fitting(:)));
 [row,col] = find(fitting==min(fitting(:)));
minimum=col;
maximum=c;
if (length(minimum)==1)
ranks(m)=minimum;
m=m+1;
fitting(col)=n;
elseif(length(minimum>1))
    jj=length(minimum);
    for indi=1:jj
        ranks(m)=minimum(indi);
        m=m+1;
        fitting(col)=n;
    end
else
end
lll=length(ranks);
end
ranks
ooo=length(ranks);
for i=1:ooo
  k=ranks(i);
  ranked(i,:)=saved(k,:);
end
ranked
generations=1;
grap=1;
while(generations<=50)
[l b]=size(ranked);
num=1;
i=1;
choose=ceil(b/2);
     indec=randi(choose)         %choosing indexes for creation of
children
     indes=indec+(choose-1)
while(i<l)
    f=1;
    one=ranked(i,:);
    two=ranked(i+1,:);
     for j=1:b
         if((indec>j)|(j>indes))
             cop(j)=one(j);
         else
             cop(j)=0;
             unoccupied(f)=j;
             check(f)=one(j);
             f=f+1;
         end
     end
     cop
     pos=1;
for t=1:length(check)
    w=check(t);
      h(pos)=find(two==w);
      pos=pos+1;
end
sorted=sort(h);
for u=1:length(unoccupied)
    cop(unoccupied(u))=two(sorted(u));
end
child(num,:)=cop;
num=num+1;
    i=i+2;
end

i=1;
while(i<l)
    f=1;
    one=ranked(i,:);
    two=ranked(i+1,:);
     for j=1:b
         if((indec>j)|(j>indes))
             cop(j)=two(j);
         else
             cop(j)=0;
             unoccupied(f)=j;
             check(f)=two(j);
             f=f+1;
         end
     end
     cop
     pos=1;
for t=1:length(check)
    w=check(t);
      h(pos)=find(one==w);
      pos=pos+1;
end
sorted=sort(h);
for u=1:length(unoccupied)
    cop(unoccupied(u))=one(sorted(u));
end
child(num,:)=cop;
num=num+1;
    i=i+2;
end

child
[m g]=size(child);
for k=1:m
    ch=randi(100);
    if(ch>70)
        finalized(k,:)=child(k,:);

    else
        temporary=child(k,:)
      pep1=randi(g);
      pep2=randi(g);
      while(pep2==pep1)
          pep2=randi(g);
      end

          temp=temporary(pep1);
          temporary(pep1)=temporary(pep2);
          temporary(pep2)=temp;

          finalized(k,:)=temporary;
      end
    end
finalized
[c d]=size(finalized);
child1=1;
t=1;
while(t<=c)
cost1=0;
setup1=0;
penality1=0;
j=finalized(t,:);
j1=length(j);
for p=1:(j1-1)
    x1=j(p);
    y1=j(p+1);
    if(cost_m(x1,y1)==1)
        cost1=cost1+1;
    else
        cost1=cost1+0;
    end
    if(setup_m(x1,y1)==1)
        setup1=setup1+1;
    else
        setup1=setup1+0;
    end
end
for p=1:(j1-1)
    x1=j(p);
    for y1=j(p+1:j1)
        if(penality_m(x1,y1)==100)
            penality1=penality1+100
        else
            penality1=penality1+0;
        end
    end
end
cost_array1(child1)=cost1;
setup_array1(child1)=setup1;
penality_array1(child1)=penality1;
child1=child1+1;
t=t+1;
end

cost_array1
setup_array1
```

```
penality_array1
r=1;
combo_saved=ranked;
[m n]=size(ranked);
[o p]=size(finalized);
for u=(m+1):(2*o)
    combo_saved(u,:)=finalized(r,:);
    r=r+1;
end
combo_saved
[l b]=size(ranked);
position=1;
for t=1:l
cost=0;
setup=0;
penality=0;
g=ranked(t,:);
for i=1:b-1
    x=g(i);
    y=g(i+1);
    if(cost_m(x,y)==1)
        cost=cost+1;
    else
        cost=cost+0;
    end
    if(setup_m(x,y)==1)
        setup=setup+1;
    else
        setup=setup+0;
    end
end
for i=1:b-1
    x=g(i);
    for y=g(i+1:b)
    if(penality_m(x,y)==100)
        penality=penality+100;
    else
        penality=penality+0;
    end
    end
end
cost_array(position)=cost;
setup_array(position)=setup;
penality_array(position)=penality;
position=position+1;
end
for q=1:length(cost_array)
combo_fitness(q)=(w1*cost_array(q))+(w2*setup_array(q))+(penality_ar
ray(q));
end
combo_fitness
 v=1;
 for q=(length(cost_array)+1):(2*length(cost_array1))

combo_fitness(q)=(w1*cost_array1(v))+(w2*setup_array1(v))+penality_a
rray1(v);
```

```
 v=v+1;
 end

combo_fitness
m=1;
combo_fitting=combo_fitness;
n=(max(combo_fitting)+400);
lll=0;
while((lll)<(length(combo_fitting)))
 [r,c] = find(combo_fitting==max(combo_fitting(:)));
 [row,col] = find(combo_fitting==min(combo_fitting(:)));
mini=col;
maxi=c;
if (length(mini)==1)
combo_ranks(m)=mini;
m=m+1;
combo_fitting(col)=n;
elseif(length(mini>1))
    jj=length(mini);
    for indi=1:jj
        combo_ranks(m)=mini(indi);
        m=m+1;
        combo_fitting(col)=n;
    end
else
end
lll=length(combo_ranks);
end
combo_ranks
ooo=length(combo_ranks);
for i=1:ooo
  k=combo_ranks(i);
  combo_ranked(i,:)=combo_saved(k,:);
end
combo_ranked
[l b]=size(combo_ranked);
nub=l/2;
for p=1:nub
    chosen(p,:)=combo_ranked(p,:);
end
chosen
ranked=chosen;
[l b]=size(ranked);
parent=1;
gener=chosen(1,:);
cost=0;
setup=0;
penality=0;
for i=1:(length(gener)-1)

    x=gener(i);
    y=gener(i+1);
    if(cost_m(x,y)==1)
        cost=cost+1;
    else
        cost=cost+0;
    end
    if(setup_m(x,y)==1)
        setup=setup+1;
    else
        setup=setup+0;
    end
end
    for i=1:b-1
    x=gener(i);
    for y=gener(i+1:b)
    if(penality_m(x,y)==100)
        penality=penality+100;
    else
        penality=penality+0;
    end
    end
end
cost_gener=cost;
setup_gener=setup;
penality_gener=penality;
fitness_gener=w1*cost_gener+w2*setup_gener+penality_gener;
fitness_graph(grap)=fitness_gener;
grap=grap+1;
generations=generations+1;
fitness_graph
end
chosen
[l b]=size(chosen);
parent=1;
for t=1:l
cost=0;
setup=0;
penality=0;
k=chosen(t,:);
for i=1:b-1
    x=k(i);
    y=k(i+1);
    if(cost_m(x,y)==1)
        cost=cost+1;
    else
        cost=cost+0;
    end
    if(setup_m(x,y)==1)
        setup=setup+1;
    else
        setup=setup+0;
    end
    end
for i=1:b-1
    x=k(i);
    for y=k(i+1:b);
        if(penality_m(x,y)==100)
        penality=penality+100;
    else
        penality=penality+0;
        end
    end
end

cost_array(parent)=cost;
setup_array(parent)=setup;
penality_array(parent)=penality;
parent=parent+1;
end
for q=1:length(cost_array)
fitness(q)=(w1*cost_array(q))+(w2*setup_array(q))+(penality_array(q)
);
end
fitness
tt=1:50;
plot(tt,fitness_graph)
fitness
```

Appendix D

This appendix contains penalty, tool change and setup change matrix for part in figure 33.

Table 20: Tool change Matrix for example part in 33

Op #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1
4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1
6	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1
7	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1
8	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	0	0	1	1
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
14	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	0	1
15	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	0	1
16	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	0	1
17	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	0	1
18	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
19	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
20	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

22	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	1	1
23	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	0	1
24	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	1	1
25	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	0	1
26	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	1	1
27	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0	0	0	1	1	1	0	0	0	0	1	1	0	0
28	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Table 21: Penalty matrix for example part in 33

Op No.	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
01	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0
5	100	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0
6	0	100	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	100	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0
8	0	0	0	100	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0
10	0	0	0	0	100	100	100	100	0	0	100	0	0	0	0	0	0	0	100	100	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	100	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	100	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	100	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0

23	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	100	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0
25	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	100	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	100	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table22: Setup change matrix for example part in 33

Op #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1
4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1
6	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1
7	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1
8	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	0	0	1	1
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
14	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	0	1
15	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	0	1
16	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	0	1
17	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	0	1
18	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
19	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
20	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

22	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	1	1
23	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	0	1
24	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	1	1
25	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	0	1
26	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	1	1
27	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0	0	0	1	1	1	0	0	0	0	1	1	0	0
28	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

References

[1] S.S Sankar, C. Rajendran, S.Ponanblam... "A multi-objective GA for scheduling a FMS", Int J. of adv manuf. technology (2003) (al. 2003) (al. 2003)

[2] Y.Koren, U.Heisel, F.Jovane, T.Moriwaki, G.Ulsoy, H.V.Brussel.. "Reconfigurable Manufacturing Systems" , Annals of CIRP, Vol 48/2 (1999)

[3] Zhuming Bi, Revisiting System Paradigms from the Viewpoint of Manufacturing Sustainability, Sustainability, Volume 3, 2011

[4] Farayi Musharavati.. "Reconfigurable Manufacturing Systems-What can Industrial Engineering and Management do?" , Industrial Engineering and Management, Volume 1, (2012)

[5] Z.M. Bi, Lihui Wang and Sherman Y.T. Lang, Current status of reconfigurable assembly systems. International Journal of Manufacturing Research, Volume 2, 2007

[6] F.Villeneuve et al, "Feature state approach for operation sequence generation", Intelligent design & Manufacturing in mechanical engineering (1998), pp 93-102

[7] T.C.Chang, R.A.Wysk.. "An introduction to automated process planning",Prentice Hall, New Jersey, (1985)

[8] Baqai, S. Schmidt, J.Y Dantan, A. Siadat, P. Martin.. "Algorithmic Design Methodology for Process Plans and Architectural Configurations of Manufacturing Systems." LCFC, Arts et Métiers ParisTech Metz, 4 Rue Augustin Fresnel, 57078 METZ CEDEX 3, France, (2009)

[9] A.Bensmaine et al, "Process plan generation in reconfigurable manufacturing systems using adapted NSGA-II and AMOSA", IEEE (2011)

[10] D. Sormez and B. Khoshnevis, "Generation of alternative process plans in integrated manufacturing systems", Journal of intelligent manufacturing, (2003), V 14, pp 509-526

[11] Shabaka and H.A ElMaraghy.. "Generation of machine configurations based on product features." International Journal of Computer Integrated Manufacturing, (2007), 20(4):355–369

[12] V. Oduguwa, A. Tiwari, and R. Roy, Evolutionary computing in manufacturing industry: an overview of recent applications, Applied Soft Computing Volume 5, 2005

[13] Franco Busetti..“Simulated Annealing Overview”, (2003)

[14] H.A. Elmaraghy, “ Reconfigurable process plans for responsive manufacturing Systems”, Digital Enterprise Technology, Springer US, (2007), pp.35-44

[15] Prabhu et al, “An operation generator network for computer aided process planning”, Journal of manufacturing systems, 9(4),283-291

[16] V.Krishna et al, “Feature based modeling and automated process plan generation for turning components”, Advances in production engineering and management, 6(2011)3, 153-162

[17] Y.N.Yang et al, “ A prototype of feature based multiple alternative process planning system with scheduling verification”, Computer and industrial engineering, (2011), V 39, pp 109-124

[18] S.H.Huang et al, “An integrated process planning project”, IPPS, (1997), Texas

[19] T. Tolio, D. Ceglarek , H.A. El Maraghy, A. Fischer , S.J. Hu, L. Laperrie` re,S.T. Newman , J.Va´ncza..“Co-evolution of products, processes and production systems.” CIRP Annals Manufacturing technology, (2010)

[20] Syed Maaz Hasan, M. Nadeem Azam, M. Salman Siddiqui, Dr. Aamer A. Baqai.. “An algorithm for the generation/selection of process plans based upon production rate”, International Conference on Advanced Modeling and Simulation, 28-30 (2011)

[21] Zheng Wen, Louis J. Durlofsky, Benjamin Van Roy,and Khalid Aziz, Use of Approximate Dynamic Programming for Production Optimization, Society of Petroleum Engineers, (2011)

[22] Pham et al, “Intelligent optimization techniques: GA, Tabu search, Simulated Annealing and Neural Networks”, Springer-London (2000)

[23] G. Venter, Review of Optimization Techniques, Encyclopedia of Aerospace Engineering,(2010)

[24] Maghsud Solimanpur,Shahram Saeedi,IRaj Mahdavi..”Solving Cell Formation problem in cellular manufacturing using Ant-colony-based optimization”,Intl J Adv Manuf Techol (2010)

[25] Yasuhiro Yamada,Kazuhiro Ookoudo,Yoshiaki Komura..”Layout Optimization of Manufacturing Cells and Allocation Optimization of Transport Robots in Reconfigurable Manufacturing Systems using Particle Swarm Optimization”,Proceedings of the 2003 IEEE/RSJ Intl,Conference on Intelligent Robots and Systems,Las Vegas,Nevada (2003)

[26] Christian Blum, Ant colony optimization: Introduction and recent trends, Physics of Life Reviews, Volume 2, (2005)

[27] Dr.R.Umarani and V.Selvi, "Particle Swarm Optimization evolution:Overview And Applications", International Journal of Engineering Science and Technology, Vol. 2(7), (2010)

[28] R.Sivaraj, Dr.T.R.Chandaran.. "A review of selection Methods in GA" ,R.Sivaraj et al, IJEST

[29] Noor.S, "Operational scheduling of traditional manufacturing systems using genetic algorithms, artificial neural networks and simulation",UK (2007)

[30] M.K.Araffin, M.Badakhshian, S.B.Sulaiman, A.A.Faieza.. "Automated Guided Vehicles Scheduling by Fuzzy GA", Dept of Mechanical Engineering, UPM Darulehsan,43400, Malaysia (2003)

[31] A.Eiben, J.E.Smith, "Introduction to evolutionary computing", Springer-London (1998)

[32] J.J Paris, H Perrival.."Dealing with design options in optimization of manufacturing systems : An evolutionary approach", Int J of Production research (2001)

[33] Mahanim Omar,Adnan Baharum,Yahya Abu Hasan.."A Job-Shop Scheduling Problem (JSSP) using Genetic Algorithm(GA)",Proceedings of the 2nd IMT-GT Regional Conference on Mathematics, Statistics and Applications,June 13-15,Penang (2006)

[34] Chang-Lin Yang,Shan-Ping Chuang,Tsung-Shing Hsu.."A Genetic algorithm for dynamic facility planning in job shop manufacturing",Int J Adv Manuf Technol (2011)

[35] Nafis Ahmed, A.F.S Anwarul Haque.. "Optimization of process planning parameter for rotational components by GA" Int conference of Mech Engg (2001)

[36] Sylvio Celso, Decio Crisol.. "Automobile stop and go cruise control system by GA, ABCM symposium series in Mechatronics, Vol 1,pp 355-362 (2004)

[37] B.I.Kazem, Ali Mahdi, Ali Talib Oudah.. "Motion planning of Robotic Arm by using GA", JJMIE, Vol:2, ISSN 1995-6665, pages 131-136 (2008)

[38] Bessem Kordoghli, Seifeddine Saadallah,Mohamed Jmali, and Noureddine Liouene.. "Scheduling Optimization in a Cloth Manufacturing Factory Using Genetic Algorithm with Fuzzy Logic for Multi-Objective Decision",JTATM, Volume 6, Issue 3, (2010)

[39] Adnan Tariq,"Operational design of a cellular manufacturing system", Pakistan (2010)

[40] X Yang, S.Deb..."Cuckoo search via levy flights", IEEE publications (2009)

[41] Truyen Tran,Trung Thanh Nguyen,Hoang Linh Nguyen.."Global Optimization using Levy Flight" , Proceedings of ICT,Hanoi Sep.24-25 (2004)

[42] Ramin Rajabioun."Cuckoo optimization algorithm", applied soft computing (2011)

[43] Kerkesova Kristina."Optimization Methods in Process Planning",5th International Multidisciplinary Conference,010 26 (2003)

[44] S.Nourali, "A mathematical model for integrated process planning and scheduling in flexible assembly jobshop environment with sequence dependent setup times",Int J. of Math. Analysis (2012), Vol 6, 2117-2132

[45] N.Ismail et al, "Manufacturing process planning optimization in reconfigurable multi-part flow lines", AMME (2008), Vol 31,issue 2

[46] A.G.Krishna and K.M.Rao, "Optimization of operation sequence in CAPP using ant colony algorithm", Int J. of adv manufacturing technology, (2006), Vol 29, 159-164

[47] W.D.Li et al "Hybrid genetic algorithm and simulated annealing approach for the optimization of process plans for prismatic parts", Int J. of production research (2002), Vol 40, 1899-1922

[48] G.H.Ma et al, "A simulated annealing based optimization algorithm for process planning", Int J. of production research, (2000), Vol 38 (12), 2671-2687

[49] G.Halevi, R.D.Weill, "Principles of process planning: a logical approach", Springer (1995)

[50] G.Singh, "Operation sequencing & machining parameters selection for rotational components using genetic algorithm & expert system", India (2006)

[51] D.Theirens and D.Goldberg, "Elitist recombination: an integrated selection recombination GA", Belgium (2002)

[52] Chiung Moon et al, "An efficient genetic algorithm for traveling salesman problem" Eurepeon journal of operational research, Elsevier (2000)

[53] S.E.Carlson, "A general methods for handling constraints in GA", Virgina, USA (1999)

[54] X.S.Yang, "Nature inspired metaheruistic algorithms", Luniver press, U.K (2008)

[55] C.Kumar and S.Deb, "Generation of optimal sequencing of machining operations in setup planning by genetic algorithms", Journal of advanced manufacturing systems, (2012), Vol 11 ,67-80

[56] Choosak Pornsing,Arnat Wattanasungsuit..''Genetic algorithm approach to the quality-related assembly line balancing problem'',Proceedings of the International Multi Conference of Engineers and Computer Scientists ,Vol 2,Hong Kong (2008)

[57] J.Jerald, P.Asokan, R.Saravanan. "Scheduling optimization of FMS using PSO", Int J of Adv Manuf tech (2005)

[58] Shanshikant Burnwal, Sankha Deb. "Scheduling optimization of FMS using Cuckoo search-based approach" , Int J of Adv Manuf (2012)

[59] Yves CRAMA. "Combinatorial Optimization Models for Production Scheduling in Automated Manufacturing Systems", Ecole d' administration des affairs, university de liege, Belgium (2012)