2006

MINIMUM FLOW TIME SCHEDULE GENETIC ALGORITHM FOR MASS CUSTOMIZATION MANUFACTURING USING MINICELLS

Phanindra Kumar Chadalavada
University of Kentucky, phanindrach@gmail.com

ABSTRACT OF THESIS

MINIMUM FLOW TIME SCHEDULE GENETIC ALGORITHM FOR MASS CUSTOMIZATION MANUFACTURING USING MINICELLS

Minicells are small manufacturing cells dedicated to an option family and organized in a multi-stage configuration for mass customization manufacturing. Product variants, depending on the customization requirements of each customer, are routed through the minicells as necessary. For successful mass customization, customized products must be manufactured at low cost and with short turn around time. Effective scheduling of jobs to be processed in minicells is essential to quickly deliver customized products. In this research, a genetic algorithm based approach is developed to schedule jobs in a minicell configuration by considering it as a multi-stage flow shop. A new crossover strategy is used in the genetic algorithm to obtain a minimum flow time schedule.

KEYWORDS: Minicell, Scheduling, Mass customization, Genetic algorithm, Flow time.

Phanindra Kumar Chadalavada

11/13/2006

MINIMUM FLOW TIME SCHEDULE GENETIC ALGORITHM FOR MASS CUSTOMIZATION MANUFACTURING USING MINICELLS

By

Phanindra Kumar Chadalavada

Dr. Fazleena Badurdeen
Director of Thesis

Dr. L. S. Stephens
Director of Graduate Studies

11/13/2006

RULES FOR THE USE OF THESIS

Unpublished thesis submitted for the Master's degree and deposited in the University of Kentucky Library are as a rule open for inspection, but are to be used only with due regard to the rights of the authors. Bibliographical references may be noted, but quotations or summaries of parts may be published only with the permission of the author, and with the usual scholarly acknowledgements.

Extensive copying or publication of the thesis in whole or in part also requires the consent of the Dean of the Graduate School of the University of Kentucky.

A library that borrows this thesis for use by its patrons is expected to secure the signature of each user.

Name Date

THESIS

Phanindra Kumar Chadalavada

The Graduate School

University of Kentucky

2006

MINIMUM FLOW TIME SCHEDULE GENETIC ALGORITHM FOR MASS CUSTOMIZATION MANUFACTURING USING MINICELLS

THESIS

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science in Mechanical Engineering
in the College of Engineering at the University of Kentucky

By

Phanindra Kumar Chadalavada
Lexington, Kentucky

Director: Dr. Fazleena Badurdeen
Assistant Professor of Mechanical Engineering
University of Kentucky Lexington, Kentucky

2006

To My Parents

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Dr. Fazleena Badurdeen, my advisor, for her continuous support and encouragement throughout my work. The progress of this work would not be achieved without her guidance and numerous suggestions.

I would also like to express thanks to my friends who have been helping me during my studies and I really appreciate their support. A special thanks to my parents and my brother for their continuous encouragement and love without which none of this is possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF FILES

1. Thesis. PDF – 1 MB (File size)

CHAPTER 1

1 INTRODUCTION

1.1 Overview

Traditionally companies have produced either customized products or standardized products. Customized products are generally produced in a manufacturing plant which operates in a low volume high variety environment and standardized products are produced in a manufacturing plant which operates in a low variety high volume environment, known as mass production. Mass production is an efficient way to produce similar products at low cost. Due to the absence of the economies of scale advantage in manufacturing customized products, the process is associated with high costs. Cellular manufacturing was developed as an alternate process by which a medium-large quantity of products could be produced while still accommodating a certain amount of product variety. Mass customization is an attempt to extend this product variety so that each individual customer's need can be catered for.

1.2 Mass Customization:

Mass Customization can be defined as "the customization and personalization of products and services for individual customers at a mass production price" [15] or efficiency. Mass Customization was presented as manufacturing strategy by Pine et al. [5] and was projected as an emerging technology by Teresko [23]. The objective of manufacturing customized products at low cost was considered to be difficult with the manufacturing capabilities that existed when the term was introduced by Davis in his book Future Perfect in 1987 [16]. Mass customization was not possible with the traditional manufacturing methods since those strategies could efficiently produce either standard products by mass production or customized products by job shops.

The existing strategies were no longer options to cater to the high product variety and better quality products demanded. To meet the changes in demand and cope with competition,

companies started to look for strategies such as Cellular Manufacturing (CM) and Just-In-Time (JIT) manufacturing.

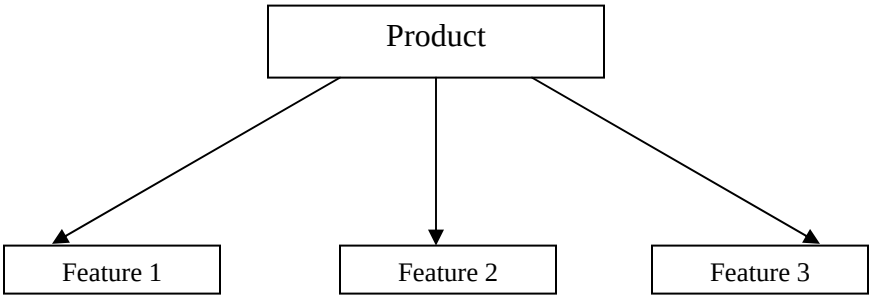
In a cellular manufacturing (CM) environment, machinery is divided into cells and a family of parts or products is produced in each cell. Similar tooling and machines are required to manufacture that part family or product family [29]. This classical cellular system is not efficient when there is a frequent change in product sequence or in composition of part or product family [24] i.e. this system is not favorable in a mass customization environment where there is dynamic change in product demand.

JIT and CM are closely related since a cellular production layout is necessary for implementing JIT strategy. JIT strategy emphasizes on reducing wastage by reducing the amount of inventory and decreasing set-up times. In the above process, while trying to satisfy the objective, JIT attempts to achieve a lot size of one. JIT strategy is primarily applied to industries where similar products or components are manufactured repeatedly [22]. Hence, even though this strategy provides methods to decrease lot size to one, it cannot be efficiently used to produce a single customized product on a make-to-order basis, particularly if products are fabricated after receiving customer orders.

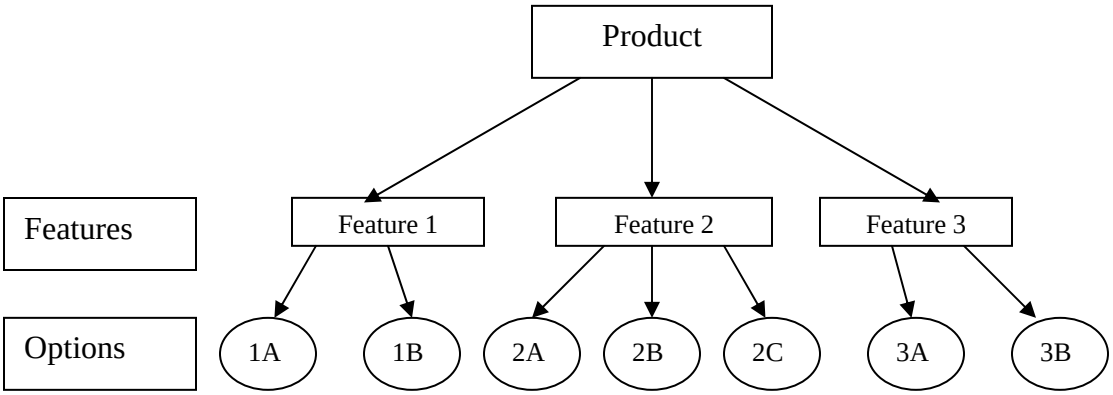
The introduction of Flexible manufacturing system (FMS) provided the ability to make a variety of products with little change over time [4]. The production levels could be increased or decreased by using numerically controlled machines [35] and most of the work in this system is automated including material handling systems. Hence introducing a FMS that can ‘make-to-order’ along with proper supporting infrastructure such as advanced information technology systems will help in increasing the flexibility. The role of information technology such as the Internet provides direct interaction with customer and can help in increasing the responsiveness of the company [6]. However, the cost of FMS has prohibited many companies from using it for lower volume manufacturing.

In designing manufacturing systems for mass customization one of the few studies considered the use of minicells [7]. Minicells are proposed for mass customization environment where options (all or few) for various features of the product are fabricated after receiving the customer

order. The final product can be assembled later using these and other standard options fabricated in bulk. A typical traditional product structure can be represented as shown in Figure 1-1(a) where as Figure 1-1(b) represents the product structure as commonly seen in mass customization environment [7]. Product variants differ based on the combination of different options used for the features. The minicells are formed by considering options, which make up a feature, and their processing needs.



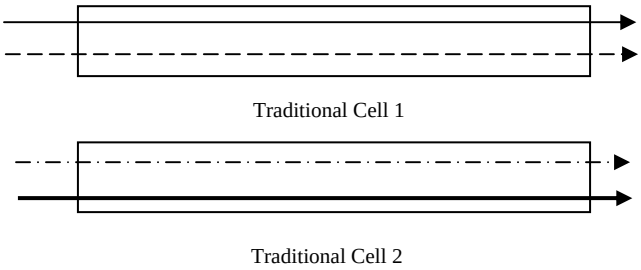
1-1(a): Traditional Product structure



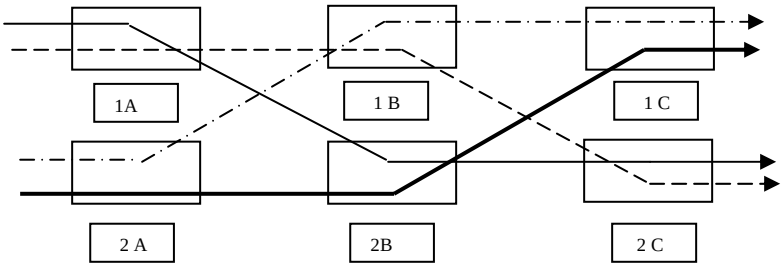
1-1(b): Mass Customization Product structure

Figure 1-1 (Adapted from [17])

In traditional cellular manufacturing each cell is dedicated to a different product family or a specific part family where as a minicell is dedicated for producing option families [7]. Different options would require same processing methods but with different setups or tooling [7]. Depending on the requirement of an option family, machines and operators are grouped together to form a minicell. Hence the function of one traditional cell is divided into more than one minicell as shown in Figure 1-2.



1-2(a): Traditional Cells



1-2(b): Minicells

Figure 1-2 (Adapted from [17])

Feitzinger and Lee [27] discuss the role of modularity for effective mass customization. Pine [4], Baldwin and Clark [57] also emphasize that modularity helps the manufacturing system to cope with the rapid demand changes and increase the flexibility. In this proposed manufacturing system, the minicells are divided into different stages and products can be routed through minicells, as necessary, in a multi-stage manufacturing environment. A modular manufacturing

system results because of the multi-cell, multi-stage environment. Such an environment will help provide the capability needed in a dynamic demand market such as mass customization.

Comparison of traditional cells and minicells [7] has been done to measure the effectiveness of a minicell based manufacturing system design. Two minicells per stage in a 3 stage minicell design and two cell traditional cell manufacturing system designs are compared. The performance of minicells has been found to be better than traditional cells [7] particularly with high variation in product demand. The average flow time has been found to be lower than that observed with traditional cells. The makespan was found to be comparable to that with traditional cell design but with a lower machine count. The results indicate the robustness of minicell designs in a high demand variation environment, similar to the situation expected for mass customization.

Low cost of production and reduced delivery time are two important aspects of mass customization and scheduling plays an important role in achieving the desired results [7]. Reducing delivery time requires a product spending less time in the manufacturing system; minimizing flow time (discussed in next section) can help achieve this. The importance of using better scheduling techniques to improve minicell performance has been exemplified [17]; hence an efficient scheduling strategy can lead to better performance with minicells.

1.3 Scheduling

Scheduling is defined as the process of assigning a set of tasks to resources over a period of time [39]. Scheduling plays an important role in manufacturing industries where it helps in reducing the costs and time to manufacture a product. With respect to the time taken to manufacture a product, two important terms considered in scheduling are makespan and flow time. While makespan is defined as the least time in which all tasks are completed i.e. the time at which the last job leaves the system, flow time is the length of the time that a job remains in the system. Makespan is also known as total throughput time and flow time as lead-time. Flow time can also be represented as the difference of completion time and ready time, ready time being the time when the job is available to be processed.

A flowshop has ‘m’ machines on which ‘n’ jobs have to be processed and each job has to be processed on every machine [37]. Considerable research has been carried out in the flow shop scheduling area over the last few decades with the focus shifting from purely theoretical problems to more general problems like ‘m-machine’ problem [1]. Flow shop scheduling is classified as NP hard [2] and this complexity makes it difficult to have exact solution methods for more than two machines [3].

Minimization of flow time leads to reduced in-process inventory and stable use of resources [14]. As discussed previously, flow time has an effect on the functioning of minicells. The responsiveness of minicells can be related to the objective of minimizing of flow time. This objective has received a lot of attention in recent years because of its relevance to the dynamic production environment [44]. Some solution procedures used to solve such scheduling problems include analytical methods, heuristics and meta-heuristics.

Analytical methods give optimal solutions, but are generally applied to small problems. Heuristics are developed for specific problems and are used to get optimal or near optimal solutions. Various heuristic methods differ based on the objective for optimization being considered and also on the methodology on which they work. Some of the commonly used objective functions in scheduling include, for example, minimization of makespan, minimization of flow time and minimizing number of tardy jobs. Heuristics such as Nawaz, Enscore, and Ham (NEH) heuristic [45] and Campbell, Dudek, and Smith (CDS) heuristic [18] were developed for the makespan objective. For minimizing flow time heuristics such as Rajendran and Zielger (RZ) heuristic [20] and Framinan and Leisten (FL) heuristic [21] were developed.

Various meta-heuristic procedures such as simulated annealing, tabu search and genetic algorithms, which are more general solution procedures, are also being used to solve optimization problems. While heuristics are used for specific problems and cannot be implemented to any general problem meta-heuristic methods are applicable to wider optimization problems. While simulated annealing and genetic algorithms are global search techniques, tabu search is a local search technique [52].

A genetic algorithm [GA] is a search method which uses concepts of evolution and natural selection [9] to find the best solution. The approach is based on the survival of the fittest. The concept is simple but provides robust and powerful adaptive search mechanisms [10]. GA's are preferred when there is no efficient solution method for a NP hard problem since they give a good solution if not an optimal solution [11]. Being an approximate algorithm, they give the solution in much lesser time than deterministic algorithms [12]. This technique has been used widely in many fields for optimization such as scheduling and traveling salesman problem. Considerable work has been done in utilizing the above mentioned solution procedures for scheduling problems [39] [43] [51].

1.4 Research Objective

The objective of this thesis is to develop a Genetic Algorithm with the objective of minimizing the average flow time needed to process a given set of jobs in a minicell configuration. This is approached by considering the minicell based manufacturing system configuration as a multi-stage flow shop.

1.5 Thesis Organization

A literature survey about recent developments in mass customization, genetic algorithms, flowshop scheduling, particularly for minimizing average flow time, are presented in chapter 2. The definitions and methodology followed in developing the GA are then presented in chapter 3. Details about the experimentation conducted and results obtained are discussed in chapter 4. Chapter 5 includes a discussion and conclusions.

CHAPTER 2

2 LITERATURE REVIEW

In this chapter a summary of literature on topics related to the research is presented. First a review of literature on mass customization is presented. This is followed by review of previous work on scheduling and genetic algorithms.

2.1 Mass Customization

Davis [16] introduced the term mass customization through his book 'Future Perfect'. According to Davis, "a one-of-a-kind product can be produced in a batch while still processing a batch by mass customization". This can be only done with advances in manufacturing systems [16]. Even though customized products were offered long before the term mass customization was coined, they were not able to produce them in high quantity. Mass customization requires the ability to produce unique products in high quantity while keeping the production costs lower.

This concept did not get recognition till Pine [4] presented this as a manufacturing strategy. According to Pine [4] mass production strategy was no longer sufficient to cater to the rapidly changing demands of the consumers. Also to survive from the ever increasing competition new strategies had to be developed. Thus the focus shifted away from mass production [4]. With the advances in manufacturing and information technology transition from mass production to mass customization strategy was possible [4]. Even companies which were setup to produce customized products have embraced mass customization strategy to reduce cost and increase responsiveness [8].

Customization is being exercised in many companies for a long time even though the term mass customization is relatively new. The level of customization varies from company to company. It can be as simple as application of postponement strategy, such as by Hewlett-Packard (HP) [27], to pure customization, for example, National Bicycle Industrial company (NBIC) in Japan [26]. In the case of HP, the assembly of power supply component was delayed from production center to distribution center where the appropriate component would be installed according to the market to which the component has to be shipped [27]. This saved a lot of inventory while

introducing certain degree of customization into the process. NBIC, on the other hand, applied pure customization in some of their processes. Certain parts in the assembly of the bicycle such as the frame were custom fabricated after receiving the order whereas the rest were standard parts. The customer involvement at the initial stages is essential for this form of mass customization.

Gilmore and Pine [56] presented four types of mass customization namely collaborative, adaptive, transparent and cosmetic. Collaborative mass customizer interacts with customer and customizes the product to his needs. Adaptive mass customizer offers a product without any dialogue with customer but the product being offered can be customized by the user after he purchases it. The product can be altered according to its use by the customer. When a manufacturer can predict the customers' needs and manufactures customized products, it is known as transparent customization. Even though there is no need for interaction with customer in this technique, their needs have to be correctly assessed. When a single product is sold to various customers in different ways, it is known as cosmetic customization. Here, the product is packaged and marketed to different customers in different ways.

For mass customization a company should have sufficient information about the customers and their choices. For this, the interaction between the company and customer is very important. This has been lacking with traditional manufacturing. The advances in information technology have enabled bridging this gap by providing means for an efficient and faster communication [6]. Also the improvements in technologies for manufacturing processes [5] and development of concepts such as Group technology (GT), Just-in-time (JIT), Flexible manufacturing systems (FMS) and Lean manufacturing [17] has helped manufacturing systems attain some of the mass customization capabilities.

Group Technology:

Most of the tasks executed in a system can be divided into a number of sets, each set having tasks similar to each other. This advantage of having similar tasks or items together, for example parts and machines or similar groceries shelved together, is utilized in group technology [28]. This concept is applied as classical cellular manufacturing (CM) strategy [24]. In CM, a set of

product families are processed in a manufacturing cell which comprises of a set of machines. These machines have similar tooling required to process that set of product families [29]. As pointed out in previous chapter, classical cellular systems are not efficient when there is a change in product variety. Hence, to overcome such obstacles, different approaches have been proposed such as virtual cells [30], network cells [34] and dynamic cells [31].

- Virtual cells: In this approach the cell does not have a fixed set of machines. The machines in this type of cell are formed by a logical grouping [32]. Instead of being present closer, machines from various departments are grouped logically to complete the manufacturing process according to the job composition [33]. Due to this configuration, communication plays an important role [33] and also material handling becomes an issue [17]. This strategy may not be an ideal choice for mass customization since the product demand varies dynamically and hence the frequent changes in layout will not be feasible [17].

- Network cells: Here the cells are divided in such a way that machines are not grouped together to produce a product family but to complete a part of the total operations required [17]. Hence more than one cell is required to complete the process. Network cell basically has responsibility for certain processes for a few products within the job mix [34]. Such an arrangement increases the flexibility within the process.

- Dynamic cells: In this approach also the machines in a cell are not fixed. But instead of forming a logical cell as in virtual cells, the machines are actually shifted from their physical position to form a cell required to complete processing of products [24]. This would decrease the material handling compared to virtual cells. Though this would increase the flexibility, frequent changes in layout due to dynamic demand in mass customization environment would make this system less efficient [17].

Due to the disadvantage of material handling in virtual cells and the disadvantage of machine movement in dynamic cells, these strategies are not ideal for mass customization. It can be observed that network cells have better advantage than other two approaches. A manufacturing

system could utilize and extend the approach used in network cells to design cells for mass customization [17].

Flexible manufacturing systems:

With flexible manufacturing system (FMS), the system has the capability to produce different products with less change-over time. Also the production levels could be increased or decreased by using numerically controlled machines [35]. This system is generally efficient for medium batch size production. Material handling plays an important role in this system. Though this system has added advantages such as reduced inventory and increased quality levels it also has certain disadvantages. This sophisticated technology uses highly costly machines and also the level of variety it can offer in products is less [36] than what is desired in mass customization environment. This is due to the limited varieties the system can process with restricted number of machines. The introduction of new product mix or new machines can cost immensely. Due to these problems, FMS use has been decreasing [35] as an efficient strategy to produce customizable products. Hence FMS cannot be directly replicated in a mass customization environment but can lend to the development of required manufacturing system design [8] [17].

Lean manufacturing:

Lean manufacturing is a philosophy based on pull systems [13] where a customer order triggers the production of a new item to replace the one that is taken out. In conventional systems the products are produced and stocked as inventory based on a forecast, typically known as push system. The pull system also has inventory present in the system. But there is a cap on the amount of inventory. The strategy of lean manufacturing is to focus on reduction of the seven wastes namely Over-production, Waiting time, Transportation, Processing, Inventory, Motion, Scrap in manufactured products or any type of business [13].

Lean manufacturing is not a manufacturing technology initiative but a philosophy to reduce the inventory levels [58]. The concept of Just-In-Time (JIT) is also an integral part of lean manufacturing. It also tries to increase the flexibility and responsiveness of the system. Even this level of flexibility may not be sufficient in a mass customization environment [35], but

nevertheless the principles of this philosophy can be used to streamline a mass customized manufacturing system and achieve the cost and time improvements.

2.2 Minicell Configuration

In traditional manufacturing system a product structure is represented by the features it is made up of, as was shown in Figure 1-1(a) in Chapter 1. In a mass customization manufacturing environment, typically, the product structure is further extended to include options for each feature. Each feature has a number of options, which are chosen by customers as necessary, leading to an increase in the number of product variants offered. This product structure is shown in Figure 1-1(b) in Chapter 1. In classical cellular manufacturing environment, product families are identified and along with required machines, a manufacturing cell is formed. The cells are so designed that products can be processed completely within the cells. A product-machine matrix lists all the product types and also the machines which are required to process them, for example, Figure 2-1(a). This matrix is used to form product families and machine cells.

But in a mass customization environment the product variants are decided by the choice of options for each feature. Hence, when there are many options available for features, the number of product variants increase. The machine requirement for some options, even when they belong to different features, might be same. According to [17], due to common processing requirements for these options, forming a manufacturing cell based on options is more likely to be beneficial than having traditional cells based on product families. This approach results in forming smaller manufacturing cells which are dedicated to an option-family rather than a traditional cell dedicated to a product family. Hence [17] used an option-machine matrix to form option families. An example is shown in Figure 2-1(b). These manufacturing cells are referred to as minicells [17]. The option-machine matrix provides the processing times of every option on all the machines required. If an option does not require processing on any machine, the processing time will be zero in the matrix.

	Machine A	Machine B	Machine C	Machine D	Machine E
Product 1	3	5	10	6	3
Product 2	5	4	1	7	5
Product 3	1	4	5	3	2
Product 4	5	4	2	1	6
Product 5	3	5	6	4	3

2-1(a): Product-Machine matrix (in terms of processing times) for Traditional cells

	Machine A	Machine B	Machine C	Machine D	Machine E
Option 1	3	0	10	6	3
Option 2	5	4	1	7	0
Option 3	0	4	5	0	2
Option 4	5	0	2	1	6
Option 5	3	5	6	4	0

2-1(b): Option-Machine matrix (in terms of processing times) for Minicells

Figure 2-1: Option-Machine Matrix

A minicell manufacturing system is divided into stages by separating the matrix into several sub-matrices. In the option-machine matrix shown in Figure 2-1(b), the matrix is divided vertically into stages. Within each stage, the options are combined based on the processing requirement in that stage and option-families are formed. Machines required to process the option family are grouped into a minicell. Due to the multi stage system, the options have to visit at least one minicell in each stage. If there is no processing required in any of the minicells in a stage, then the option can skip that stage.

2.3 Scheduling

The process of assigning resources to some tasks in a period of time is known as scheduling [39]. A flowshop has ‘m’ machines on which ‘n’ jobs have to be processed and each job has to be processed on every machine [37]. The problem of ordering these jobs in an appropriate sequence to process on machines is a main issue in any company. In a pure flow shop all the jobs have to be processed on all machines whereas in a general flow shop few jobs may skip processing on some machines [38]. Some other classifications in scheduling are single machine scheduling, job shop scheduling and parallel machine scheduling.

A flow shop scheduling problem is deterministic when the data about all the jobs is given. The problem becomes stochastic even if there is one variable parameter. The scheduling problem can also be classified based on the arrival of the job into the system. If the job is available at zero time, i.e. start of the process, or when the arrival times are known in advance, then it is classified as static. When the job arrival is random it is known as dynamic. The former case is an ideal condition while the later one is the more general practical situation. The higher the number of variable parameters in a system, the more complex the scheduling problem becomes [25].

A permutation sequence is developed from a scheduling problem when jobs in the same order are processed on all machines [38]. Non-permutation schedule gives the sequence which does not remain same for all machines in a flowshop. Most of the research is associated with finding good permutation schedules because it is less complex compared to finding a non-permutation schedule.

Flow shop scheduling is classified as NP hard [2] for three or more machines. Most of the research studied the two machine problem [40] while heuristics were developed for the NP hard problems [40]. The recent trend in research is to focus on more practical problem such as ‘m’ machine problem [1]. Apart from two machine and ‘m’ machine problem, single machine problem is also studied widely due to its vast application. Computer operations, bottleneck machine in a line or a continuous flow process are areas where single machine scheduling is applicable.

Enumeration methods or analytical methods, heuristics and meta-heuristics are some of the solution procedures used for scheduling problems. While enumeration methods give the optimal solution and are the most efficient, the other two procedures give optimal or near optimal solutions. Due to the nature of the enumeration method’s ability to work well for only small problem sizes [40], heuristics were developed to obtain optimal or near optimal solutions for large problem sizes. All the solution procedures have an objective function that needs to be minimized or maximized. Objective functions which are generally minimized for example include makespan [43] and flow time [21] whereas objective function such as reliability [42] is maximized in scheduling problems. Given a set of jobs, the time at which the last job leaves the system is known as makespan. It is the total completion time required. Considerable research has

been done on this objective [44] since it is related to the throughput [37]. Some of the earliest work on this objective was presented by heuristics such as Campbell, Dudek, and Smith (CDS) heuristic [18] and Nawaz, Enscore, and Ham (NEH) heuristic [45].

Flow time minimization has recently gained a lot of importance in recent research work due to its effect on resources and inventory [37]. Flow time is the length of the time that a job remains in the system. It is also known as lead-time. The Rajendran and Zielger (RZ) heuristic [20] is one of the earlier heuristics for minimizing of flow time. In the RZ heuristic, first the jobs are sequenced according to a priority rule and then sequence is improved by sequential insertion of each job, according to the seed sequence, into the best sequence found so far. Süer et al. [39] presented work on minimizing average flow time with a single machine problem with non-zero ready times of jobs. An evolutionary program was developed to achieve near optimal solutions.

Gupta et al. [46] presented heuristic algorithms for a two machine flowshop problem. The scheduling problem proposed to reduce the total flow time, while keeping makespan of the schedule minimum. Parallel machine scheduling problems are also widely covered because of their applications in general cases. Kravchenko and Werner [47] presented a heuristic algorithm to minimize the mean flow time. They considered unit set up times for each job and the problem consisted of ‘m’ identical parallel machines. Azizoglu and Kirca [48] also considered identical parallel machines problem, but the objective was to minimize the total weighted flow time. In general it is assumed in a scheduling problem that all problems are of same importance. But if the priority levels for different jobs are varying [38], a factor known as the weight is assigned to each job. Thus the total weighted flow time objective holds good for such problems.

Woo and Yim [49] (WY) developed a heuristic for minimization of mean flow time in an ‘m’ machine flow shop. The method consisted of two phases where in jobs are ranked according to ascending sum of processing times in first phase and then partial sequences are obtained by inserting non-scheduled jobs in all possible positions. The best partial sequence is maintained as the solution. This heuristic outperformed the RZ heuristic when the number of jobs considered in the problem was large. The WY heuristic and the RZ heuristic were outperformed by the heuristic developed by Liu and Reeves [50] (LR heuristic). This heuristic also has the jobs sorted in the initial phase and then followed by sequence buildup. But the jobs are ranked in the

ascending order of a combined index function. The index consists of weighted total machine idle time and artificial total flow time. A search method was then used to improve the solution.

Framinan and Leisten [21] (FL) presented a heuristic based on the NEH heuristic. The objective of the heuristic is to minimize the total flow time. This heuristic will be discussed in detail in Chapter 4. This heuristic performed better than the RZ and WY heuristics.

When a solution procedure makes use of an already developed heuristic in one of its steps, it is known as a composite heuristic [37]. These kinds of heuristics are developed to modify or enhance the previous successful heuristics. Allahverdi and Aldowaisan [40] developed one such composite heuristic which combines WY and RZ heuristics. One of their heuristics, IH7, consists of three steps. The first step has the initial solution which is obtained by WY heuristic. The second step produces a schedule by using the second phase of RZ heuristic. Lastly, a local search method is used to improve the solution. This heuristic gives solutions better than those found independently with the two heuristics involved.

Some of the heuristics discussed above use search techniques such as simulated annealing, tabu search or genetic algorithm in their solution methods. Known as meta-heuristics, they are applicable to wide range of problems and are not restricted by problem size as the enumeration methods. Meta-heuristics are classified as approximate algorithms since they obtain near optimal solutions whereas deterministic algorithms obtain optimal solutions. Simulated annealing was first presented by Kirkpatrick et al. [51]. The algorithm starts by generating an initial solution and by initializing parameter ‘T’. This parameter is known as temperature. The algorithm replaces the present solution with a solution from its neighborhood if that solution is better than the current one. The solution is found with help of a function. The algorithm also replaces the current solution with a bad solution from its neighborhood if it satisfies a probability. The probability is a function of parameter ‘T’ and difference in the values of functions. The value of ‘T’ is decreased gradually during the search process, thus at the beginning of the search the solution is replaced frequently and less frequently as the value decreases. The above steps are repeated until a termination criterion is reached. The main advantage of this meta-heuristic is that the problem does not get stuck at local minima because of the inclusion of the probability

function. But the main disadvantage is that the method takes large amount of time to get the solution.

Considerable research was done by using simulated annealing in the field of scheduling. Ruiz-Torres, Enscore, and Barton [59] presented a heuristic based on simulated annealing to minimize the average flow time and number of tardy jobs on identical parallel machine problem. Low, Yeh, and Huang [60] presented a simulated annealing heuristic for flow shop scheduling problems. A heuristic on unrelated parallel machines problem was given by Low [61].

Tabu search is also an iterative procedure which searches for a better solution in a neighborhood. The algorithm contains a list of solutions that have been previously visited and when moving into a new neighborhood, the algorithm excludes these solutions [52]. The algorithm tries to search for the best solution which is not contained in the list. The advantage of this method is also that it avoids local minima but a proper termination condition has to be set, which otherwise may end in the method not providing a good result [53]**.** Apart from scheduling tabu search has its applications in inventory management, telecommunications industry and in design area [53]. Heuristics based on tabu search were developed for flow shop scheduling problems [62] [63]. Grabowski and Wodecki [63] presented a heuristic for minimizing makespan for a permutation flow shop problem. A heuristic for minimizing flow time was developed by Chen, Usher, and Palanimuthu [64].

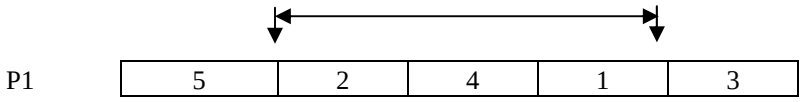
Based on laws of natural selection and survival of the fittest [9], Genetic Algorithms are also an iterative procedure where each iteration is called a generation. In this method the solution is represented by a chromosome and is generally called solution candidate or population (collection of chromosomes). A chromosome consists of genes which represent the encoding of the solution to the problem. Typically in one generation, or iteration, new population is created called off-springs and then each off-spring is modified assuming it will result in improvement of the obtained population. An off-spring is created by combining two chromosomes by applying a crossover operator. This off-spring is then mutated to slightly modify it. A good solution is expected at end of each generation.

The performance of a GA can be affected by combination of selection methods, mutation methods and crossover methods. Various techniques can be found for chromosome representation. Also known as encoding, the coded variables or genes can be represented in various formats such as binary code, real number and letter. The binary code is used generally in classical encoding problems such as capacity lot sizing while the use of real coded GA has increased due to the wider applications.

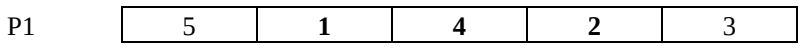
Selection method plays an important role in functioning of GA by differentiating good population from bad and thus has a role in selecting populations which are to be mated or have to be advanced to next generation.

The mutation operation is performed on a single chromosome to slightly alter the gene sequence. Some of the types of mutation strategies available are:

- Inversion mutation [9] [65]:
  In this method, a part of the chromosome is selected and the genes present in that part are inverted. In Figure 2-2, the part of the chromosome selected to be inverted is shown.



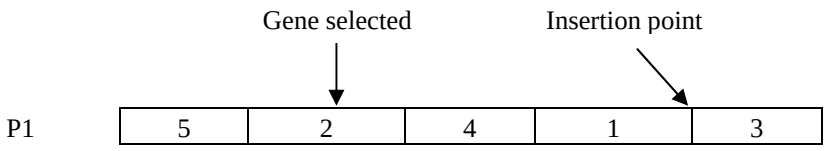
2-2(a): Before mutation



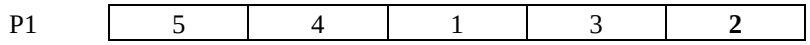
2-2(b): After mutation

Figure 2-2: Inversion mutation

- Insertion mutation [9] [66]:
  In this method a gene is selected randomly and then inserted in a random position. Figure 2-3 illustrates the process.



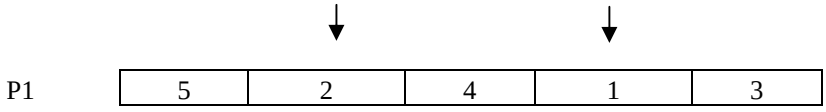
2-3(a): Before mutation



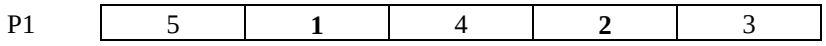
2-3(b): After mutation

Figure 2-3: Insertion mutation

- Reciprocal exchange mutation [9] [67]:
  In this method, two genes are selected at random and then the positions are exchanged.



2-4(a): Before mutation

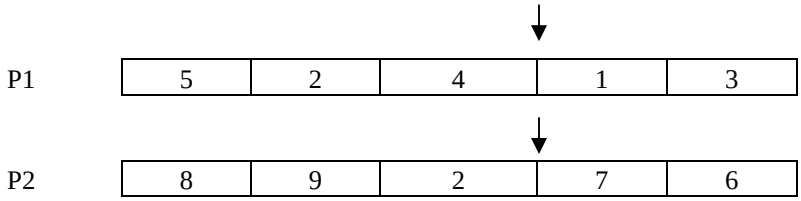


2-4(b): After mutation

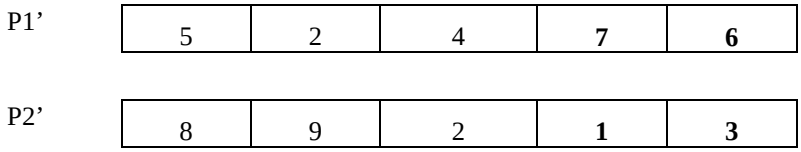
Figure 2-4: Reciprocal exchange mutation

The crossover operator is applied to parent population by selecting two chromosomes at a time to generate children or off-spring by exchanging genes between the two parents. There are various techniques to perform crossover operation. Some crossover strategies are:

- Single cut-point crossover [9]:
  Also known as single point crossover, in this method a crossover point is selected on the parent chromosome and all the genes beyond this point are swapped between the two parent chromosomes.



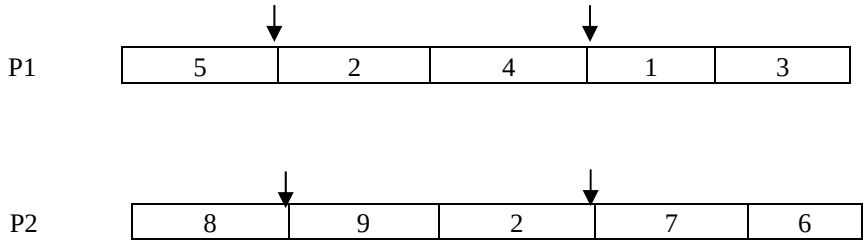
2-5(a): Parent chromosomes



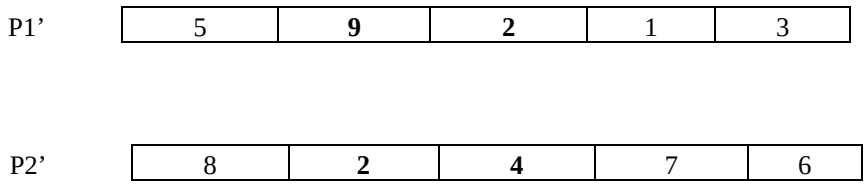
2-5(b): Off-spring

Figure 2-5: Single cut-point crossover

- Two point crossover [68] [69]:
  In this method, two points are selected on the parent chromosomes and the genes present in between the two cut points are swapped.



2-6(a): Parent chromosomes

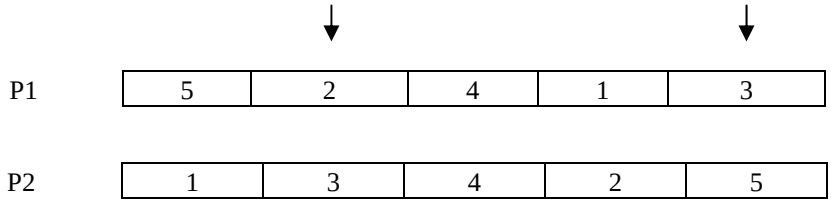


2-6(b): Off-spring

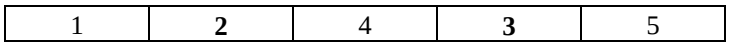
Figure 2-6: Two point crossover

- Order-based crossover [9]:

  In this method a set of positions from one of the parent chromosomes are selected. The values present in those positions in that order replace the same value positions in second parent chromosome. Only one off-spring is generated through this crossover strategy.



2-7(a): Parent chromosomes

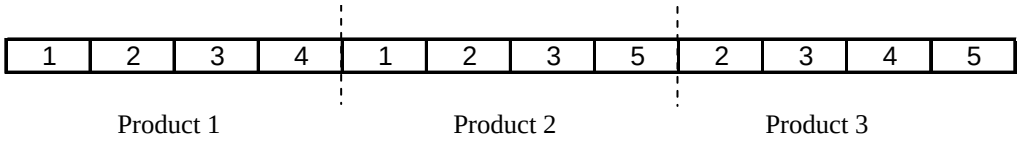


2-7(b): Off-spring

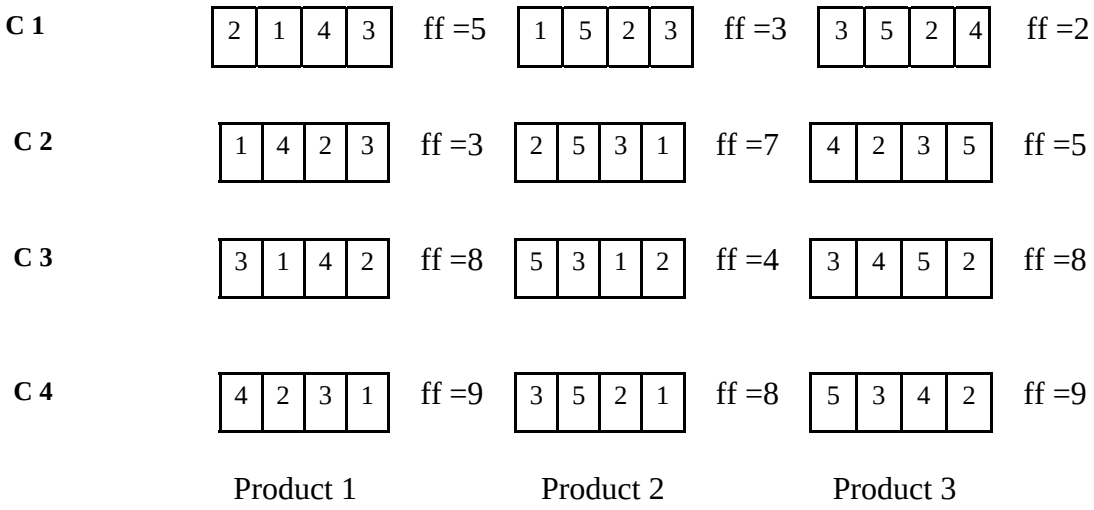
Figure 2-7: Order-based crossover

- Multi-Chromosome crossover:

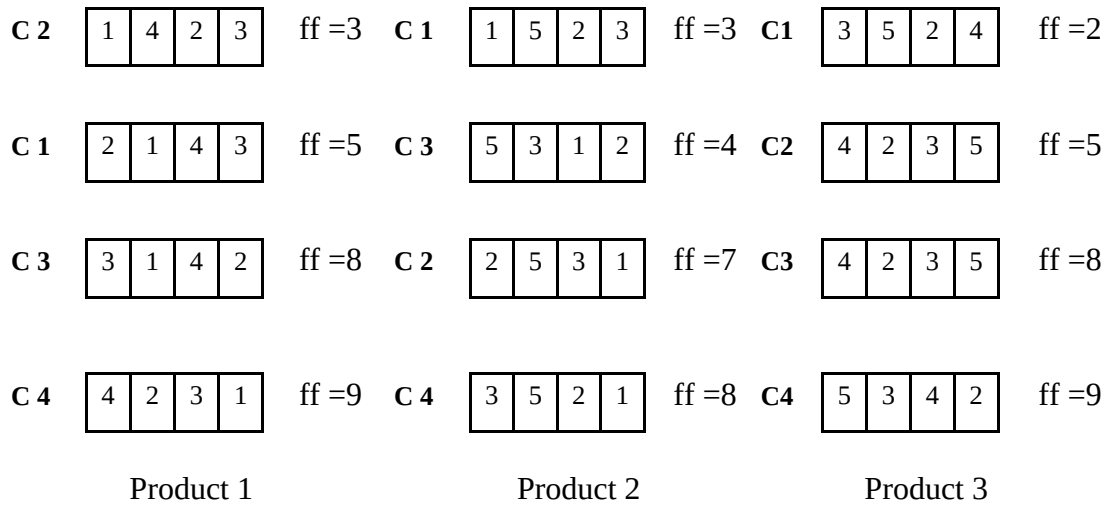
  Süer et al. [55] presented a new crossover technique known as Multi-Chromosome crossover strategy. The crossover is done between two part chromosomes as opposed to the conventional method where crossover is performed between two entire chromosomes. Süer et al. [55] demonstrated this strategy for a multi-product, multi-period capacitated lot sizing problem. In this problem the chromosome has multiple blocks and the blocks are considered as part chromosomes. Each block represents a separate product as shown in Figure 2-8(a). Taking a population size of 4, the part chromosomes for entire population are shown in Figure 2-8(b). The part chromosomes are ranked as shown in Figure 2-8(c). In this crossover strategy, the ranked part chromosomes are divided into three groups namely X, Y and Z groups. The size of each group depends on the ratio i.e. top x % are allotted to group X, the next y % are allotted to group Y and the remaining are allotted to group Z. The first part chromosome from Z group is crossed over with the first part chromosome from X group (the best group), second part chromosome from Z group with second one in X group and so on. The same pattern is followed for Y group also. The part chromosomes in X group are crossed over with in the group.



2-8 (a): Entire chromosome representation



2-8 (b): Part chromosome representation



2-8(c): Ranking part chromosomes according to fitness values (ff)

Figure 2-8: Multi-Chromosome crossover

The production schedule of each product is determined for this period of time to minimize the total cost incurred in manufacturing the products. The GA developed for this problem using the multi-chromosome crossover strategy presented good results. This representation of chromosome is similar to a multi-stage flow shop environment, where a chromosome can be divided into parts, each part representing a stage. The minicell configuration manufacturing system is similar to a multi-stage flow shop environment. This method is thus used to solve the minicell configuration problem.

CHAPTER 3

3 METHODOLOGY

The general framework of the minicell configuration considered in this research and the methodology followed to develop the genetic algorithm are described in detail in this chapter. The proposed genetic algorithm attempts to reduce the average flow time in a multi-stage minicell configuration, which resembles a flow shop environment. This objective is achieved by trying to obtain an optimal sequence of jobs to be processed on machines at the minicell level.

3.1 Description of Minicell Configuration

A brief description of a minicell configuration has already been provided in Chapter 2. The basis to form minicells and the importance of option-machine matrix has been presented. This section provides the additional information required about minicell configuration. The minicell assignment matrix gives the information about minicell configuration. It is similar to the option-machine matrix (Figure 2-1), but with machine numbers being replaced by number of stages. The values of the matrix indicate the minicell number in which the corresponding option can be processed. An example family formation matrix is shown in Figure 3-1.

		Stage 1	Stage2	Stage3
Feature 1	Option 1	1	3	1
	Option 2	3	2	1
Feature 2	Option 3	2	1	3
	Option 4	1	2	1
Feature 3	Option 5	2	3	2

Figure 3-1: Minicell Assignment matrix

3.2 Operation of Minicell configuration

The product variants are routed through the minicells, as necessary, depending on the options chosen, to complete the processing. Since the processing needs of options do not remain same in all stages, options belonging to one family in one stage can be routed to a different minicell in the next stage. Consider the data presented in Figure 3-2(a) and Figure 3-2(b).

	Feature 1	Feature 2	Feature 3
Product 1	1	3	1
Product 2	3	2	1
Product 3	2	1	3
Product 4	1	2	3
Product 5	2	3	2

3-2(a)

		Stage 1	Stage2	Stage3
Feature 1	Option 1	1	3	1
	Option 2	3	2	1
	Option 3	2	1	3
Feature 2	Option 1	2	1	3
	Option 2	1	3	2
	Option 3	1	2	1
Feature 3	Option 1	2	1	3
	Option 2	1	3	2
	Option 3	2	3	2

3-2(b)

Figure 3-2

Figure 3-2(a) gives the product structure and Figure 3-2(b) gives the minicell assignment details. With this data, the manufacturing system can be represented as shown in the following Figure 3-3. The routing of product 1, product 2 and product 3 is shown in Figure 3-3.

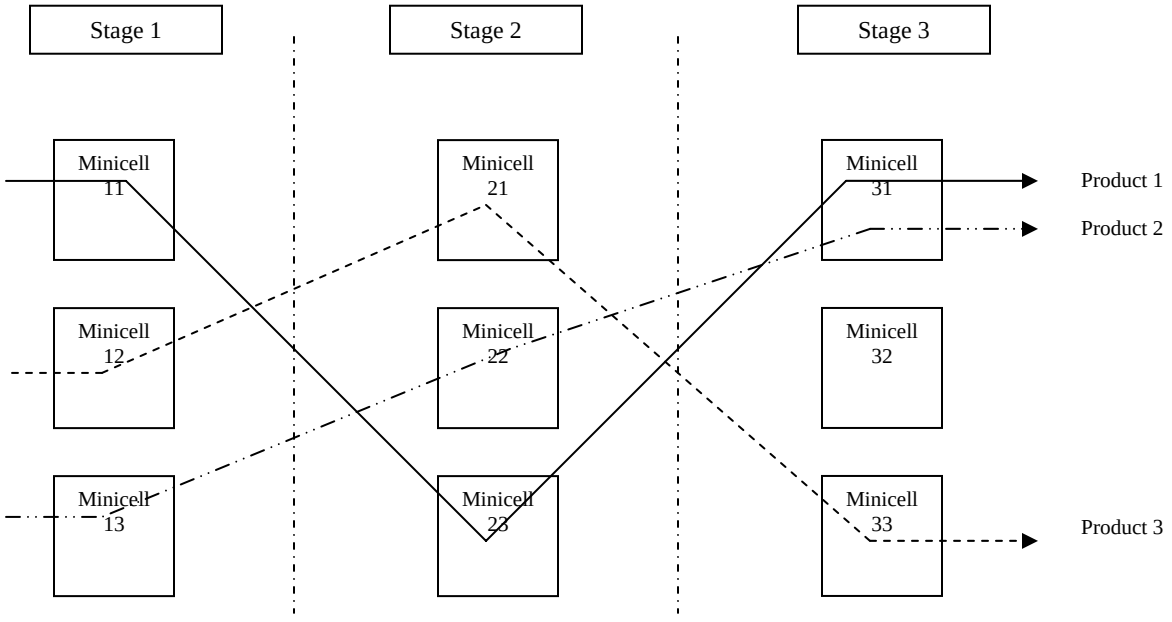


Figure 3-3: Operation of minicell manufacturing system

The above figure explains how the products flow through the system. Three stages are present according to the data in Figure 3-2. The number of minicells per stage is also given in that matrix. The routing of each product as it progresses through the stages is shown. Based on the option number chosen for each product, the product structure varies. At the end of all the stages, the customized options for the product variant are completely processed.

After the routing of all the product variants has been established, the sequence in which these jobs will be processed needs to be determined. Though minicells are smaller than traditional cells, but operate similarly, the same flowshop scheduling techniques can be applied here as well [17]. Either a permutation or a non-permutation schedule can be developed. While a permutation schedule is more suitable for a traditional flowshop problem where all the jobs have to be processed on each machine, the same may not hold good for a minicell configuration. All the jobs do not enter every minicell and thus may also skip some machines in minicells when processing on those machines is not required. Since the option-families also do not remain the same in every stage, developing a permutation schedule may not result in effective minicell functioning. Therefore, in this research a non-permutation schedule is to be established. Developing a schedule with the objective function of minimizing average flow time is considered for the present research problem. Flow time is one of the important factors in evaluating the performance of minicell. Optimizing the considered objective will not only have an effect on the delivery time of the orders [7] but also on the resources [37] in the system. Hence an effective solution method has to be developed to solve the scheduling problem.

3.3 Minimum Flow Time Schedule Genetic Algorithm (MFGA)

The details of the MFGA developed to solve the multi stage flow shop scheduling problem in minicell configuration are outlined in the sections below. Some of the assumptions considered in the scheduling problem are:

- Pre-emption is not allowed
- Set up times are not considered
- Jobs are available at time zero i.e. start of problem

- All jobs are equally important
- Jobs have multiple operations
- Non-permutation sequence is considered

The entire working of the MFGA is represented in a flow chart diagram in Figure 3-4.

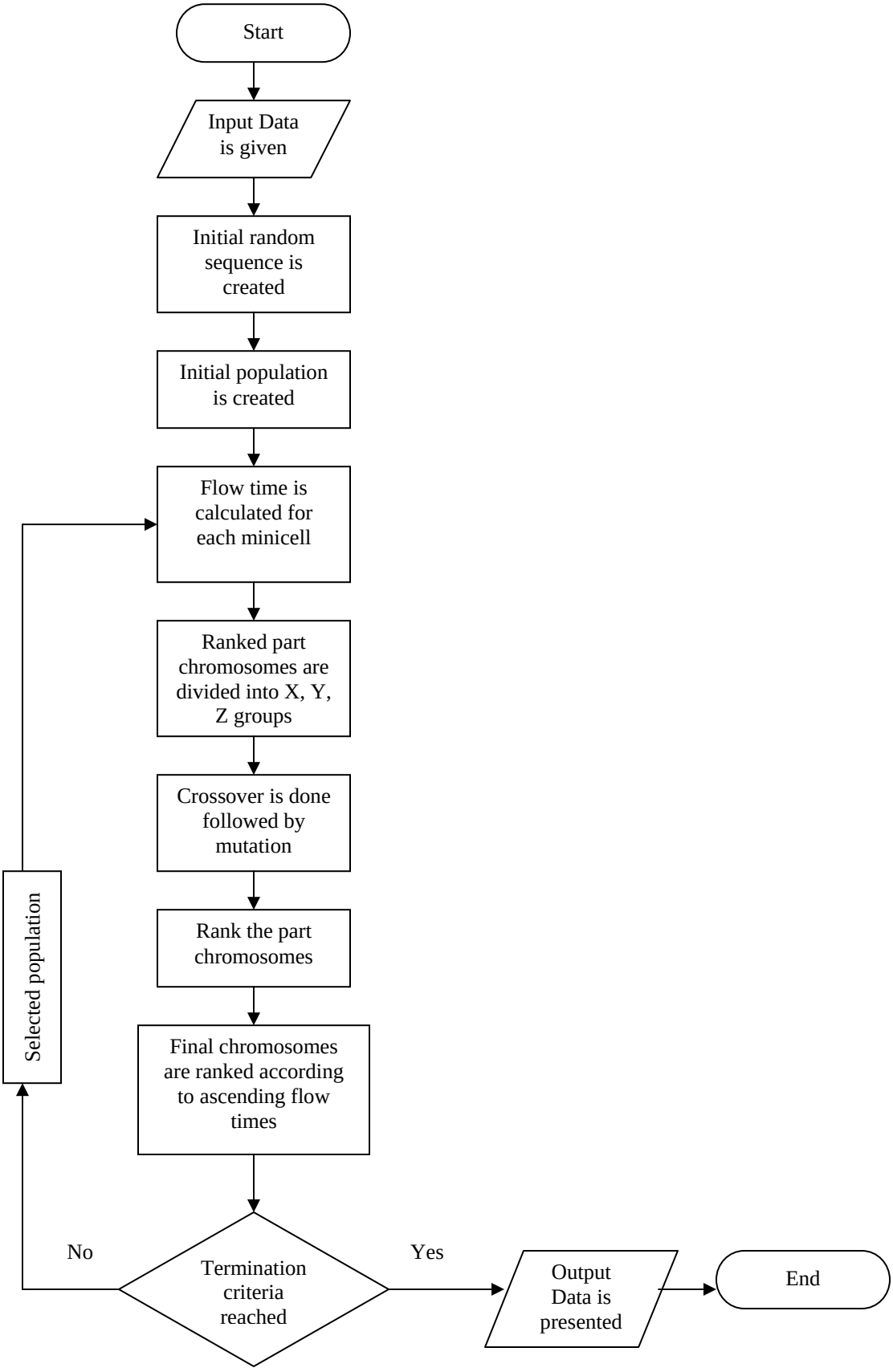


Figure 3-4: Flow chart representation of working of GA

3.3.1 Chromosome Representation

Each chromosome has 'S' blocks, each block representing a separate stage. There are 'mc_s' blocks in every stage, each representing a minicell in that stage. The number of stages and minicells depend on the type of problem. The genes in each minicell section of the chromosome shows the sequence which is to be followed in processing the jobs in that particular minicell. Hence the numbers of genes in minicells do not remain constant throughout the chromosome. Also some jobs may skip some minicells based on the minicell design. Whenever a product variant does not require processing on any of the machines in a minicell, it does not visit that minicell. Hence, minicell skipping can take place. This contributes to the variation of number of genes in different minicells within a single chromosome. For example we consider a 3 stage problem with 2 minicells in each stage. Let the number of jobs be 6. The Figure 3-5 represents one possible chromosome representation.

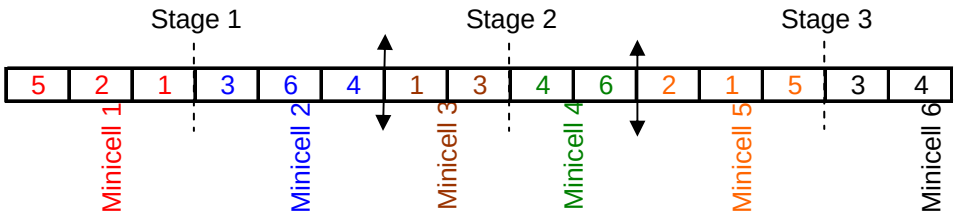


Figure 3-5: Chromosome representation

In the above chromosome, for each minicell, the genes represent the job sequence with the k^{th} job being in position 'K' in sequence. Thus, for example, in minicell 1, job 2 is the 2^{nd} in sequence. Each minicell has a unique sequence. As explained previously, the number of genes in each minicell varies since all jobs do not have to visit every minicell.

3.3.2 Fitness Function

The objective in the MFGA is to minimize the average flow time to complete the product variants. Flow time (f_i) is defined as the amount of time a job remains in the system. It is given

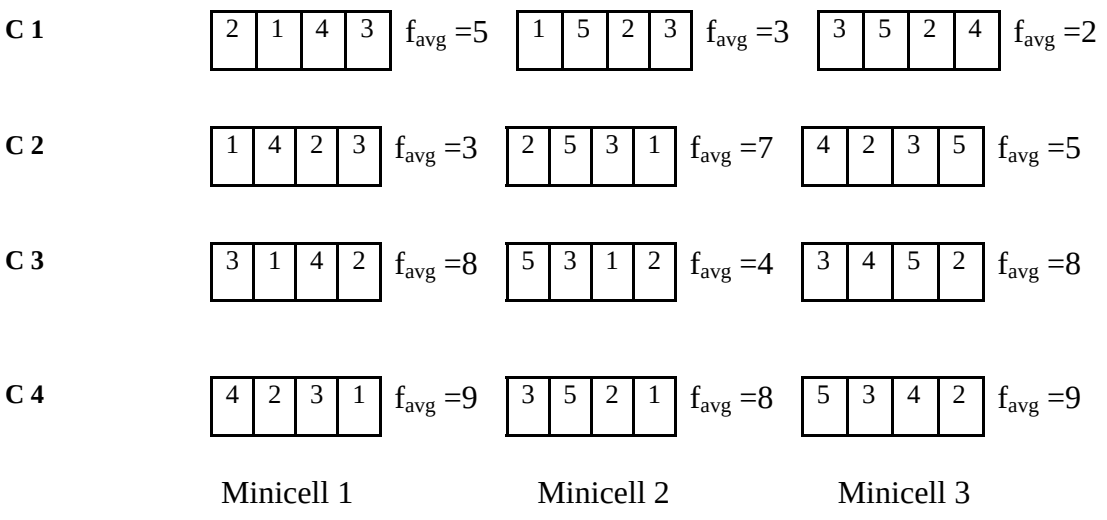
by the difference between completion time (c_i) and ready time (r_i), i.e. $f_i = c_i - r_i$ for job i. The average flow time is calculated by taking the average of flow times of individual jobs. The chromosome with the least fitness function value is the most desired.

3.3.3 Selection and Reproduction

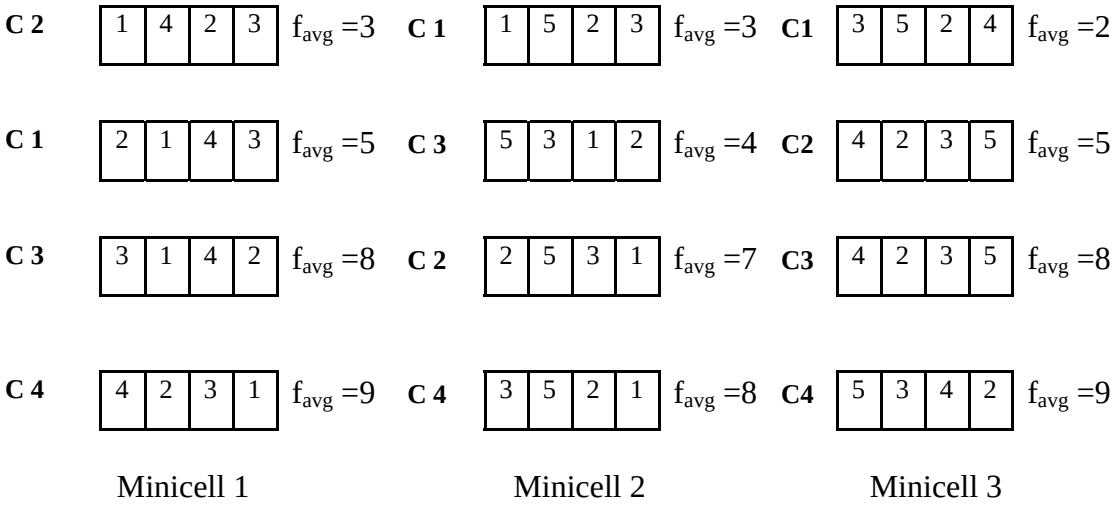
The population size remains same throughout the problem. When selecting parents for genetic operations, the entire initial population is considered for the first generation. From second generation onwards, the off-springs created from the previous generation are taken as parents.

3.3.4 Crossover Strategy

Due to division of chromosomes into stages and subsequent division into minicells, the multi-chromosome crossover strategy [55] was applied in the GA. This strategy has already been explained in Chapter 2. The crossover is done between two part chromosomes as opposed to the conventional method where crossover is performed between two entire chromosomes. In the MFGA, genes for each minicell are considered a separate part. Each part chromosome has a separate fitness function value, average flow time (f_{avg}), depending on its sequence. For example if there are 3 minicells per stage and population size of 4, then for each minicell there are 3 part chromosomes within a stage as shown in Figure 3-6(a). These part chromosomes are ranked according to fitness function value as shown in Figure 3-6(b).



3-6 (a): Part chromosome representation for one stage



3-6(b): Ranking part chromosomes according to fitness values: average flow time (f_{avg})

Figure 3-6

The Multi-Chromosome crossover strategy, where part chromosomes are divided into groups and a crossover proceeds is described below. Let X=50%, Y=25% and Z=25%. For example, the ranked part chromosomes are then grouped as shown in Figure 3-7. Crossover is then applied for each minicell segment separately. Thus for minicell 1, the part chromosome from group Z i.e. C4 is crossed over with C2 as shown in Figure 3-8(c). Similarly C3 from group Y is crossed over with C2 as shown in Figure 3-8(b). For X group, the crossover is done between C2 and C1, within the group, as shown in Figure 3-8(a).

The cut point is randomly determined for each instance and does not remain constant. After the crossover is done, the new part chromosomes are checked to see for repetition of the values in genes. If a number is found more than once then, a repairing action is performed. This number is replaced by the missing number so that the resulting string presents a complete sequence for processing the jobs.

If the number of part chromosomes in Y or Z group are more than those in X group then the sequence of part chromosomes in X group are repeated for the remaining ones. For example, if there are 5 part chromosomes in Y group and only 3 part chromosomes in X group then the first

3 part chromosomes from Y group are crossed over with corresponding ones in X group. For the 4^{th} part chromosome onwards the sequence for X group starts from 1^{st} part chromosome. The above procedure is repeated in every minicell.

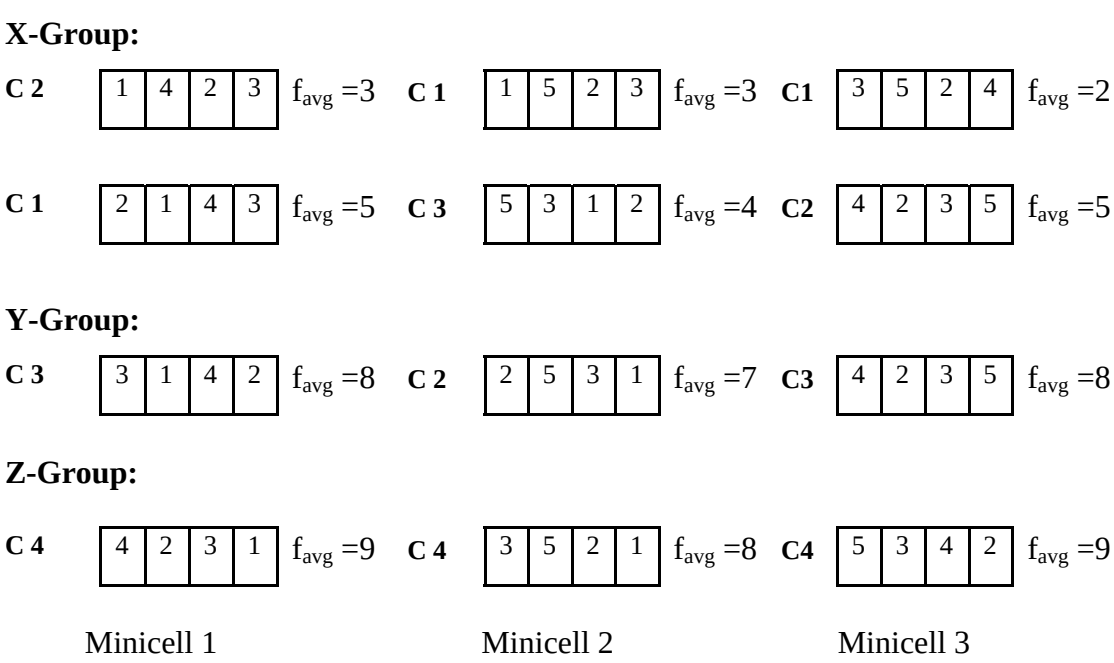


Figure 3-7: Ranked part chromosomes divided into groups

3-8 (a): Group X part chromosomes crossed over within the group

3-8 (b): Group Y part chromosome crossed over with the one in group X

3-8 (c): Group Z part chromosome crossed over with the one in group X

Figure 3-8: Illustration of crossover strategy for part chromosomes of minicell 1

3.3.5 Mutation Operator

The Reciprocal exchange mutation [25] is applied for the MFGA in this research. In general, in this operation two genes are selected at random and their positions are exchanged. Thus we get a new sequence in the chromosome.

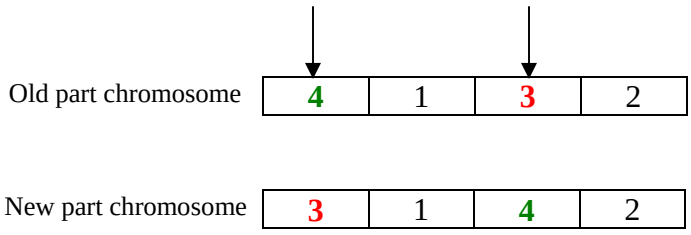


Figure 3-9: Illustration of mutation operation

In the MFGA mutation is applied for each part chromosome, that is minicell, separately. The above figure explains the process involved in Reciprocal exchange mutation. The genes present in the part chromosome are assigned random numbers and the first gene whose random number is less than the mutation probability is chosen for mutation. The other gene is selected randomly from the same minicell or part chromosome and both these genes are exchanged.

3.3.6 Regrouping

When the above two genetic operations are completed all the part chromosomes are joined to form the final single chromosome. The first part chromosome in order from each minicell group

is taken and joined to form one single chromosome. The first part chromosomes are the best ranked ones in previous operations. Similarly, rests of chromosomes are formed in the sequence. The MFGA is terminated after the required number of generations.

3.3.7 Software Program for MFGA

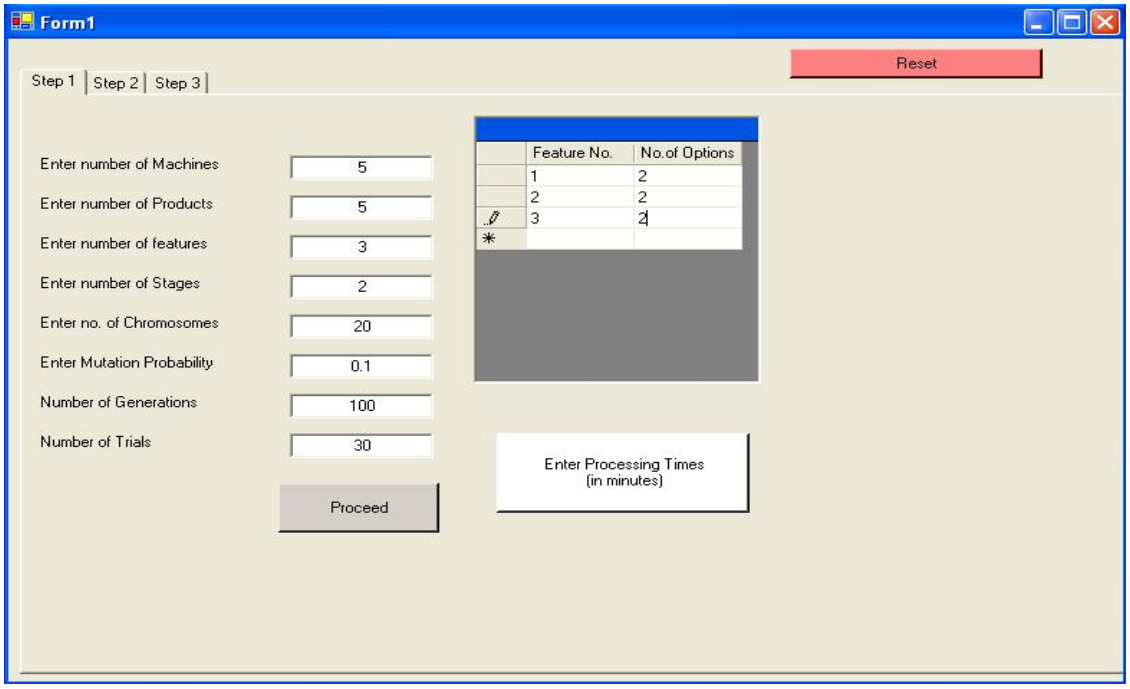
In this section the working procedure of the proposed genetic algorithm is discussed. The following steps give the sequence in which the algorithm proceeds:

3.3.7.1 Initial population

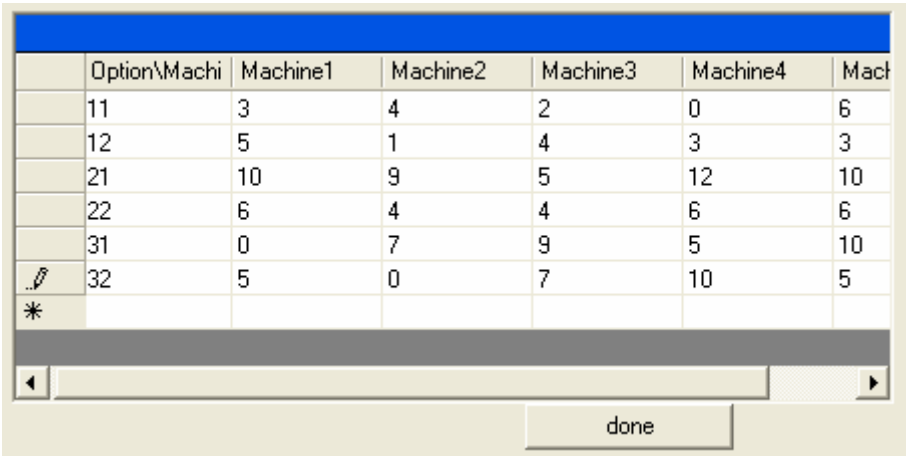
The initial population consists of (C) chromosomes. The first chromosome is generated with the help of the data that user gives as input. The following parameters are the input given by the user:

- Number of machines
- Number of products
- Number of features
- Number of stages
- Number of chromosomes(population size)
- Mutation probability
- Number of generations
- Number of trials
- Processing times of each option on every machine: *Option-machine matrix*
- Product ID (Figure 3-11)
- Family formation matrix

These parameters are given through the interface created for the MFGA shown below.



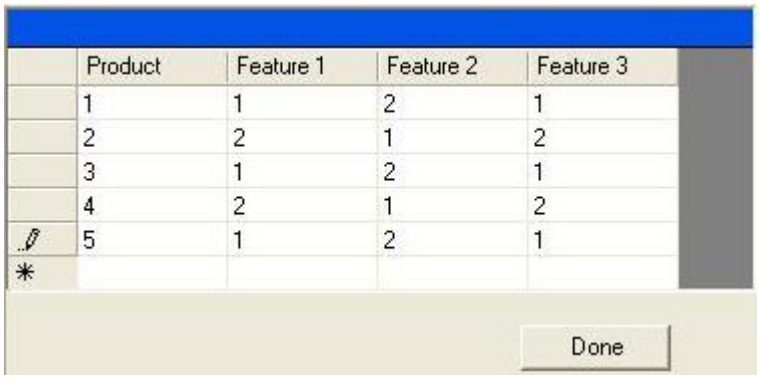
3-10(a): Screenshot of interface; Input data is entered



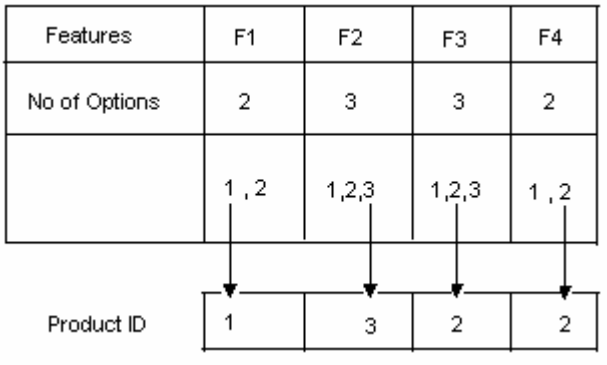
3-10(b): Screenshot of interface; Option-machine matrix

Figure 3-10: Screenshots of Interface

Product ID gives the final representation of the product composition from its product structure (Figure 1-1). It represents the option number of each feature which is required to complete the processing of the product as shown in Figure 3-11.



3-11(a): Product ID



3-11(b): Product ID: Each division represents a feature containing the corresponding option number

Figure 3-11: Product ID

By considering the relation between number of minicells per stage, family formation matrix and the product ID, a sequence is generated for each minicell. When the sequence of each minicell is combined we get the entire scheduling sequence covering all the stages to complete the manufacturing in all minicells. The first chromosome is formed by combining all the sequences obtained for each minicell. The rest of the chromosomes for the initial population are formed by essentially shuffling or randomizing the genes within each minicell. Care is taken that the no extra job is allowed into or omitted from the original sequence.

The above mentioned MFGA was developed by coding in Microsoft Visual Basic.NET. The ease of creating an interface for the user coupled with the ability to present the work as an executable

file contributed to the choice of selecting Visual Basic.NET as the programming language. The interface created for the MFGA and the steps in which it proceeds are explained in Appendix II.

CHAPTER 4

4 EXPERIMENTATION

The experimentations conducted with different problems using the MFGA are presented in this chapter. The test problems were also solved by applying the FL heuristic. The results obtained by using the heuristic are then compared with the results obtained from MFGA to evaluate the effectiveness of the proposed method.

4.1 Initial MFGA testing parameters

Test cases were developed to test the performance of MFGA. Table 4-1 shows the various input parameters considered for all the problems tested.

Table 4-1: Input parameters

	Parameters	Values
1	Number of machines	5
2	Number of features customized	3
3	Number of options per feature	2
4	Number of stages	2
5	Population size	(i)10 (ii)20 (iii)30
6	Mutation probability, (P_m)	(i) 0.1 (ii)0.2 (iii)0.3

Four different strategies were considered with the Multi-chromosome crossover strategy to divide the population into groups. The strategies vary based on percentage of the population assigned to each group i.e. different values of X, Y and Z as shown in Table 4-2 below.

Table 4-2: Four strategies for Multi-chromosome crossover strategy

Strategy	Percentage assigned		
	X	Y	Z
Strategy 1	50	25	25
Strategy 2	25	25	50
Strategy 3	33	33	34
Strategy 4	30	30	40

For each strategy the genetic algorithm was run for 50, 100 and 200 generations while the number of trials remains constant at 30 i.e. for example, each population size and generations combination, repeated with the corresponding mutation probability, is run for 30 times to evaluate the repeatability of MFGA results. Data regarding the formation of option-machine matrix, product ID and family formation matrix is given below.

The following table gives the values of processing times (minutes) of each option on all machines:

Table 4-3: Option-Machine matrix

m/c's / Options	M1	M2	M3	M4	M5
11	3	4	2	0	6
12	5	1	4	3	3
21	10	9	5	12	10
22	6	4	4	6	6
31	0	7	9	5	10
32	5	0	7	10	5

The options are represented by a two digit ID. The first digit represents the feature number. The second digit in the ID corresponds to the option number corresponding to the feature represented by the first digit. For example, 12 denote the second option for feature 1. Hence the processing

time of each product can be calculated by combining the processing times of each option of a feature selected by the user.

The manufacturing system is assumed to have two stages in this problem. The following table gives the assignment of machines shown in Table 4-4 to each stage.

Table 4-4: Assignment of Machines

	Stage 1		Stage 2		
Machines	M1	M2	M3	M4	M5

4.2 FL Heuristic

The multi-stage scheduling problem for minimizing average flow time was also solved by using Framinan and Leisten (FL) heuristic [21]. It is based on the NEH heuristic [45]. Though the NEH heuristic was developed with the objective of minimizing the makespan, the FL heuristic utilizes one of its concepts in developing the heuristic for minimizing total flow time objective. The initial step of ordering the jobs in descending order of sum of processing times and then forming the partial schedules is borrowed into this heuristic. But instead of ordering them in descending order of sum of processing times, they are ranked in ascending order.

The following steps are followed to solve the problem using FL heuristic (adapted from [21]).

Step1:

Set $K = 1$. The jobs are ordered according to ascending sum of processing times. The best job with the shortest processing time is selected and set the partial schedule as current solution S and remaining jobs are stored in a set R.

Step 2:

Update $K = K + 1$

Step 3:

The K^{th} job is then inserted into the k possible slots, and the partial schedule with lowest flow time is retained as best solution. Set this solution as S. This job is then removed from the original ordered list R.

Step 4:

If $k > 2$, a general pair wise interchange is applied to this partial schedule, i.e. all the possible combinations of the jobs are checked, and the best partial solution with respect to flow time is retained as solution S.

Step 5:

If the set R is null then stop, else go to *Step 2*.

Since this heuristic is specifically developed for a permutation flow shop, minor modifications are needed before applying it to scheduling a minicell configuration. This heuristic is individually applied to each minicell as a separate flow shop scheduling problem to get the best possible sequence. The fitness value is the average flow time. In the calculation of fitness value for a given minicell, the completion times for products from the preceding minicell are considered.

The FL heuristic performs better than the RZ heuristic and WY heuristic discussed in previous chapters. This heuristic can also be used as part of a composite heuristic. One such heuristic, IH7-FL [37], outperforms even this heuristic. The problem set prepared for the initial testing of the MFGA are used to obtain the minimum average flow time schedule with the heuristic. These values are then compared with those obtained using the MFGA to evaluate the effectiveness of the latter.

4.3 Test problems

The three test problems are compared in the following sections. The information about the product structure and minicell assignment is given separately for each problem.

4.3.1 Two Product Variant Problem

The input parameters are already given in Tables 4-1 to 4-5. The following table gives the options chosen for each feature, for the five jobs considered in this problem. The number in each column represents the option chosen for the particular feature. For example, Product 2 = option 2 for feature 1 + option 1 for feature 2 + option 2 for feature 3. Each job is assigned a different product ID.

Table 4-5: Product ID

	Feature 1	Feature 2	Feature 3
Product 1	1	2	1
Product 2	2	1	2
Product 3	1	2	1
Product 4	2	1	2
Product 5	1	2	1

Even though there are five products given in the above table, only two distinct product variants exist. While products 1, 3 and 5 constitute one variant (i.e. 1-2-1 combination), products 2 and 4 constitute the second variant (i.e. 2-1-2 combination). In scheduling, each product is considered a separate job because they will be processed on a make-to-order fashion for mass customization.

The following table gives the minicell assignment matrix i.e. information regarding the assignment of options to minicells in each stage.

Table 4-6: Minicell assignment matrix

	Options	Stage 1	Stage 2
Feature 1	11	1	3
	12	2	1
Feature 2	21	1	2
	22	2	1
Feature 3	31	1	2
	32	3	1

From the above table it can be seen that options 11, 21 and 31 are routed into minicell 1 of stage1, options 12 and 22 are routed into minicell 2 while option 32 is routed to minicell 3. In stage 2 options 12, 22 and 32 are routed to minicell1; options 21 and 31 are routed to minicell 2 while option 11 is routed to minicell 3.

The above test problem 1 is solved by using the FL heuristic. The detailed calculations are presented in Appendix I. The result obtained from this calculation is give below in Figure 4-1.

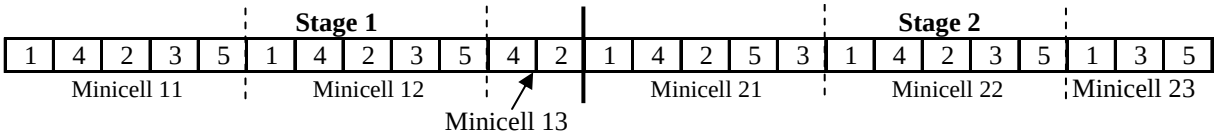


Figure 4-1: Sequence obtained by FL heuristic

The average flow time obtained for the above sequence is: $\mathbf{F_{avg} = 105}$ minutes.

Results from MFGA

The MFGA was then used to solve the same problem. The results obtained after completing the runs for all four strategies, for all combinations of number of generations and different mutation probabilities, are tabulated as shown in Tables 4-7 to 4-13.

The results presented in Table 4-7, Table 4-8, Table 4-9 and Table 4-10 are the best fitness values obtained after 30 trials for Strategy 1, Strategy 2, Strategy 3 and Strategy 4 respectively.

In Table 4-11, the results are presented summarized with respect to the three generation numbers considered. The mean value is the average of all the best fitness values obtained for every population size and mutation probability after 30 trials. From the values it appears that 100 generations gives the best result when both mean and standard deviation are considered. While 200 generations gives the lowest average result for all strategies, except Strategy 3, the standard deviation with 200 generations is generally higher.

Table 4-12 gives the mean and standard deviation, after 30 trials, for average flow time with respect to population sizes. Strategy 1 with a population size 20 gives the best result, with lowest standard deviation value and a low mean fitness value. Population size of 30 also delivers a good result, with a lower mean value but higher standard deviation than with population size of 20. The other combinations do not present such good results.

Table 4-13 presents the results compared with respect to the different mutation probabilities (P_m). The three different values considered are 0.1, 0.2 and 0.3. Strategies 1, 2 and 4 give a good result for P_m=0.1 with Strategy 1 giving the least mean fitness value. Combination of Strategy 2 and P_m=0.2 also gives a comparable result, while other combinations have higher standard deviations.

Table 4-7: Results for Strategy 1; X=50%, Y=25%, Z=25%

	P_m= 0.1			P_m= 0.2			P_m= 0.3		
Pop #	10	20	30	10	20	30	10	20	30
Generations									
50	116	113	116	125	112	114	123	113	114
100	113	113	110	117	114	112	116	111	111
200	108	109	111	115	113	110	102	115	109

Table 4-8: Results for Strategy 2; X=25%, Y=25%, Z=50%

	P_m= 0.1			P_m= 0.2			P_m= 0.3		
Pop Size	10	20	30	10	20	30	10	20	30
Generations									
50	115	116	115	117	116	115	121	121	115
100	111	116	113	112	115	111	112	115	113
200	114	111	109	110	108	109	115	112	114

Table 4-9: Results for Strategy 3; X=33%, Y=33%, Z=34%

	P_m= 0.1			P_m= 0.2			P_m= 0.3		
Pop Size	10	20	30	10	20	30	10	20	30
Generations									
50	117	112	117	114	115	114	116	118	113
100	113	116	115	109	113	110	111	114	113
200	115	111	112	115	112	110	118	115	109

Table 4-10: Results obtained for strategy 4; X=30%, Y=30%, Z=40%

	P_m= 0.1			P_m= 0.2			P_m= 0.3		
Pop Size	10	20	30	10	20	30	10	20	30
Generations									
50	115	110	114	116	115	113	124	117	117
100	112	115	115	104	117	112	119	115	111
200	112	111	107	117	111	113	118	106	113

Table 4-11: Comparison of Results for Number of Generations

	S1			S2			S3			S4		
	50	100	200	50	100	200	50	100	200	50	100	200
Mean	116.2	113	110.2	116.8	113.1	111.33	115.1	112.7	113	115.6	113.3	112
Std.Dev.	4.63	2.34	4.02	2.48	1.83	2.54	2.02	2.29	2.9	3.80	4.33	3.96

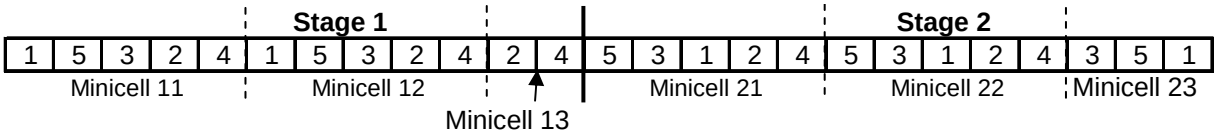
Table 4-12: Comparison of Results for Population sizes

	S1			S2			S3			S4		
	10	20	30	10	20	30	10	20	30	10	20	30
Mean	115	112.55	111.88	114.1	114.4	112.7	114.2	114	112.5	115.2	113	112.7
Std.Dev.	7	1.74	2.315	3.40	3.71	2.44	2.8	2.23	2.60	5.58	3.70	2.7

Table 4-13: Comparison of Results at different Mutation probabilities

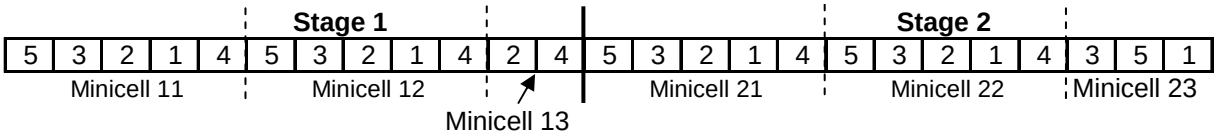
	S1			S2			S3			S4		
	0.1	0.2	0.3	0.1	0.2	0.3	0.1	0.2	0.3	0.1	0.2	0.3
Mean	112.1	114.66	112.6	113.3	112.5	115.3	114.2	112.4	114.1	112.3	113.1	115.5
Std.Dev.	2.84	4.35	5.67	2.5	3.28	3.42	2.27	2.29	3.01	2.73	4.04	5.15

The MFGA has given solutions which are better than the FL heuristic in two instances. In Strategy 1, for a combination of 200 generations, 10 population size and P_m= 0.3 the MFGA has given an average flow time of 102 minutes. In Strategy 4, for a combination of 100 generations, 10 population size and P_m= 0.2 the MFGA gives an average flow time of 104 minutes. The sequences given by the MFGA for the above two cases are presented below:



F_{avg} = 102 minutes

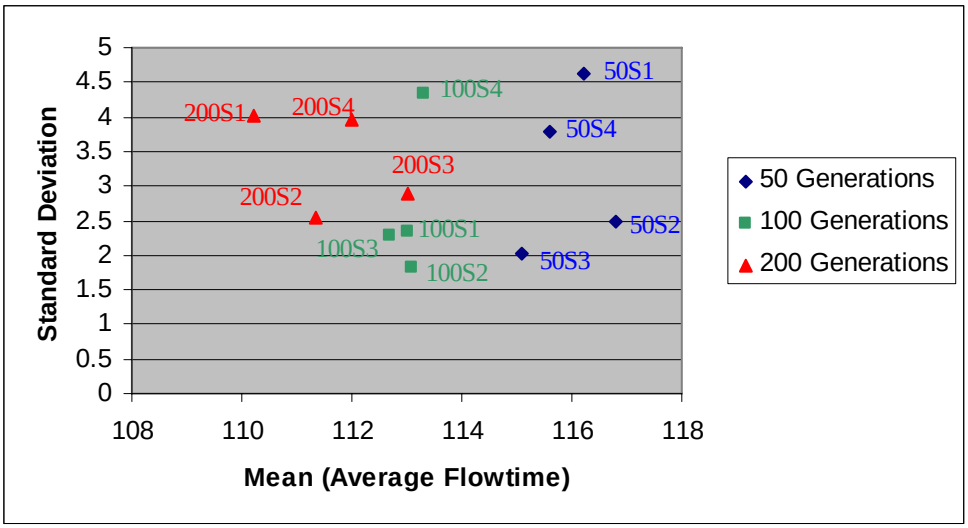
Figure 4-2: Sequence obtained by MFGA for Strategy 1



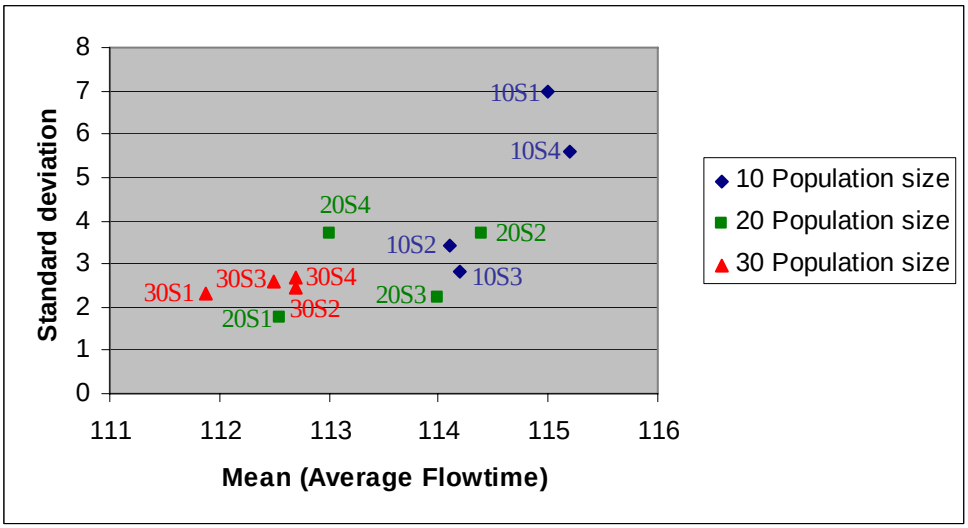
F_{avg} = 104 minutes

Figure 4-3: Sequence obtained by MFGA for Strategy 4

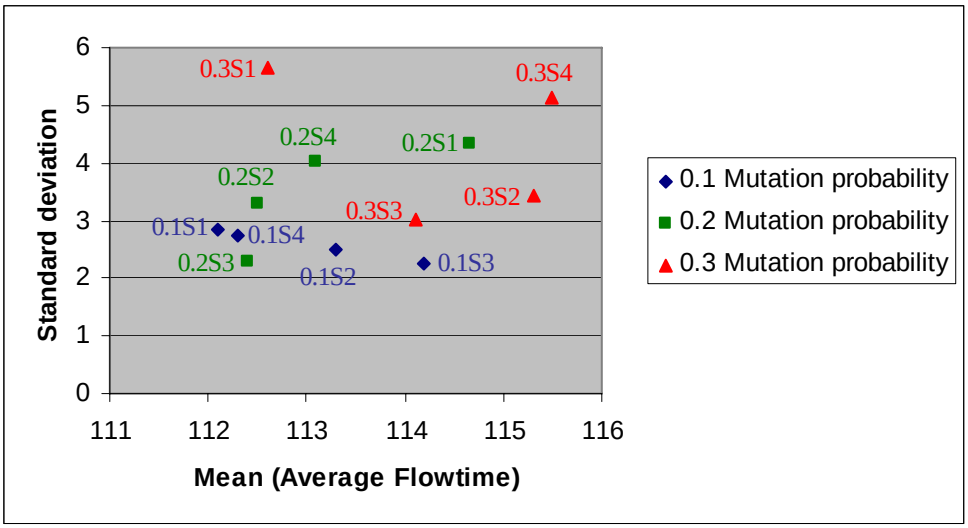
The results obtained with the different strategies, as well as other parameters, are quite variable as can be observed from the results presented. However, the standard deviation is small and, in most cases, the results lie within a short range Scatter plots were developed for different numbers of generations, population sizes and mutation probabilities to identify the most effective combinations that generate the minimum average flow time schedule. Figure 4-4(a) presents the scatter plot for number of generations, Figure 4-4(b) for population sizes and Figure 4-4(c) for mutation probabilities.



4-4(a): Variation of Average flow time with Number of Generations



4-4(b): Variation of Average flow time with Population sizes



4-4(c): Variation of Average flow time with Mutation probabilities

Figure 4-4: Scatter Plots

In the above figures, data points in the lower left region of the graph are the most desired combinations. These scatter plots exposes the data points which have low mean average flow time as well as a low standard deviation. From Figure 4-4(a) it can be observed that 100 generations with Strategies 1, 2 and 3 lie in the desired region. In Figure 4-4(b) strategy 1 and population sizes 20 and 30 lie in the lower left region of the plot making them the desired values. In Figure 4-4(c) a P_m = 0.1 with Strategies 1, 4 and P_m = 0.3 and Strategy 3 lie in the desired region.

Convergence diagrams were also used to evaluate the performance of the MFGA. Convergence diagram for a 50 generations case is shown in Figure 4-5. The average flow time values for the 50 generations are taken from a Strategy 1 test problem with population size of 10 and P_m=0.1. The average flow time values for the 100 generations case shown in Figure 4-6(a) are taken from a Strategy 1 test problem with population size of 10 and P_m=0.3. Figure 4-6(b) represents for 200 generations case and the values are taken from a Strategy 1 test problem with population size of 30 and P_m=0.3.

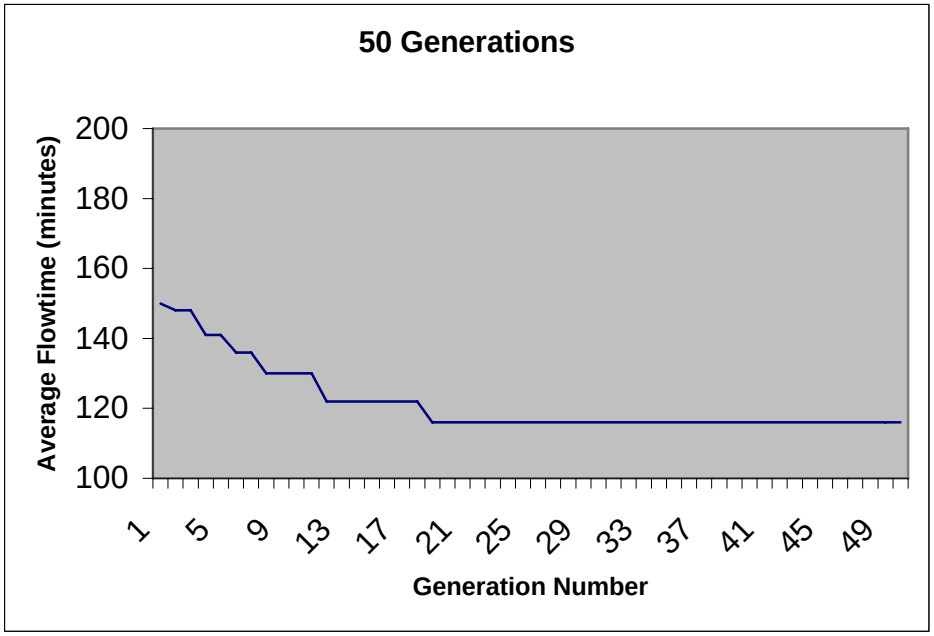
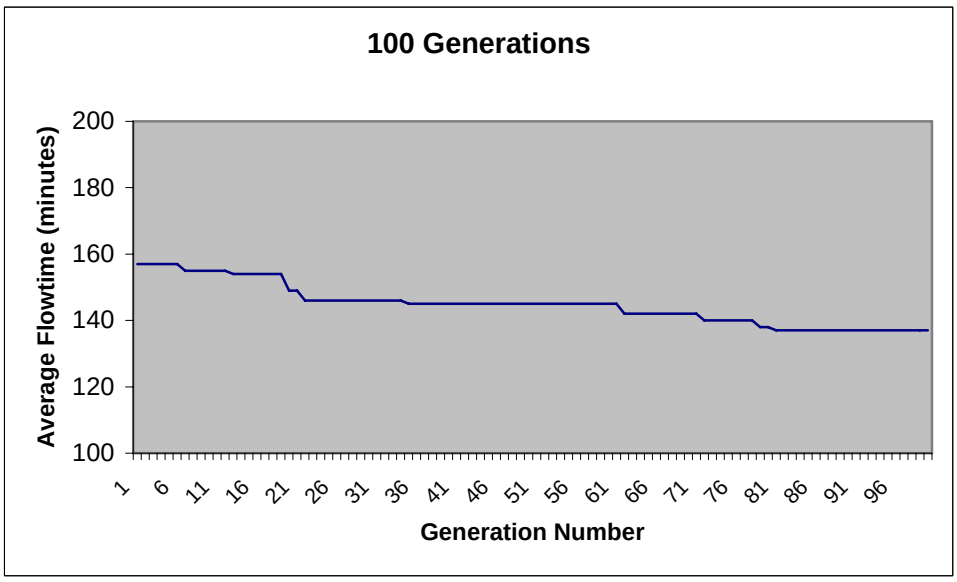
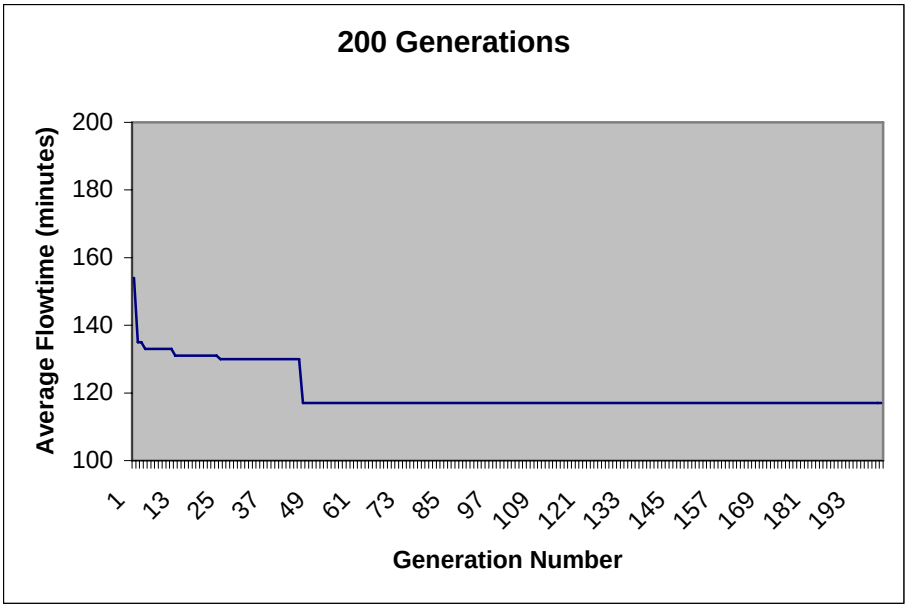


Figure 4-5: Convergence diagram for 50 generations MFGA problem

The following figures present the convergence diagrams for 100 and 200 generations. It can be observed that, in all three cases, the MFGA is converging before 100 generations are completed. This indicates that using the MFGA for more than 100 generations, most likely does not have any significant benefit.



4-6(a): Convergence diagram for 100 generations MFGA problem



4-6(b): Convergence diagram for 200 generations MFGA problem

Figure 4-6: Convergence Diagrams

From the above information it can be deduced that Strategy 1 gives better results when compared to the other ones. This strategy combined with 100 generations, 20 or 30 population size and P_m = 0.1 is likely to present schedules that gives the lowest average flow time with the multi-stage minicell configuration considered.

4.3.2 Six Product Variant Problem

The number of product variants in the problem is increased to six. Table 4-14 gives the product structure. The previous problem had five products but only 2 product variants. In this problem, there are six products and six different product variants. The six different product variants that have to be scheduled are represented in Table 4-14. The assignment of options to minicells is given in Table 4-15.

Table 4-14: Product ID; product structure

	Feature 1	Feature 2	Feature 3
Product 1	1	2	1
Product 2	2	1	2
Product 3	2	1	1
Product 4	1	2	2
Product 5	1	1	2
Product 6	2	2	1

Table 4-15: Minicell assignment

	Options	Stage 1	Stage 2
Feature 1	11	1	2
	12	2	1
Feature 2	21	1	3
	22	3	2
Feature 3	31	2	1
	32	1	3

From the above information, the path of each product variant through the minicell configuration can be identified. The problem was solved by using the FL heuristic as indicated earlier. The schedule obtained from the heuristic is given in Figure 4-7. The average flow time obtained for this sequence is **115** minutes.

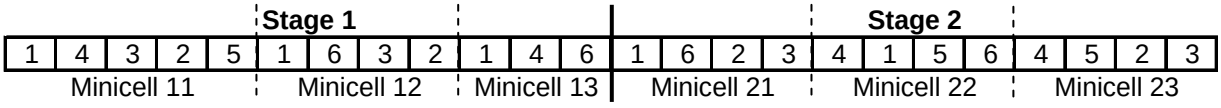


Figure 4-7: Sequence obtained by FL heuristic

The problem was then solved by applying MFGA. All the four strategies (given in Table 4-2) were considered; population size of 20, mutation probability of 0.1 and 100 generations were chosen for the experimentation based on the analysis of previous results. The results obtained from the MFGA are presented below.

Strategy 1:

The sequence obtained is shown in Figure 4-8. The average flow time obtained for this schedule was **107** minutes.

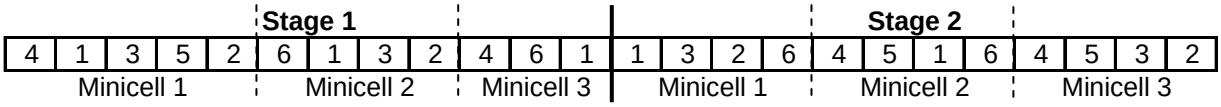


Figure 4-8: Sequence obtained by MFGA for Strategy 1

Strategy 2:

The sequence obtained is shown in Figure 4-9. The average flow time obtained for this schedule was **106** minutes.

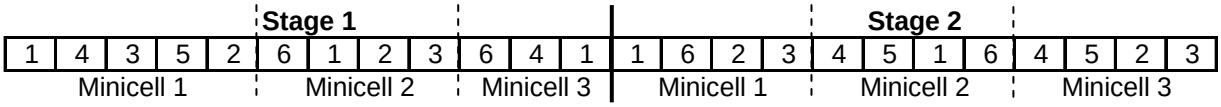


Figure 4-9: Sequence obtained by MFGA for Strategy 2

Strategy 3:

The sequence obtained is shown in Figure 4-10. The average flow time obtained for this schedule was **111** minutes.

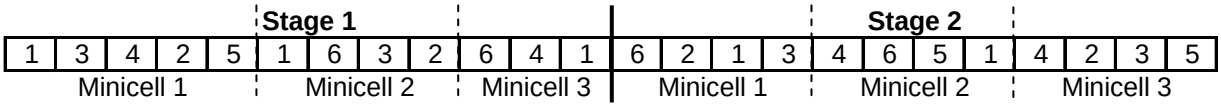


Figure 4-10: Sequence obtained by MFGA for Strategy 3

Strategy 4:

The sequence obtained is shown in Figure 4-11. The average flow time obtained for this schedule was **107** minutes.

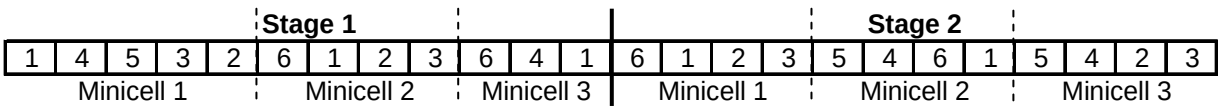


Figure 4-11: Sequence obtained by MFGA for Strategy 4

In addition to the above four strategies the problem was also solved by assigning the following values to X, Y and Z groups:

$$X\ \% = 100$$
$$Y\ \% = 0$$
$$Z\ \% = 0$$

This approach simplifies the crossover in that, now there is only a single group of chromosomes within which crossover will be performed. The approach is still different to traditional crossover because each minicell is considered an independent block.

The sequence obtained for the above parameters is shown in Figure 4-12. The average flow time obtained for this schedule was **107** minutes.

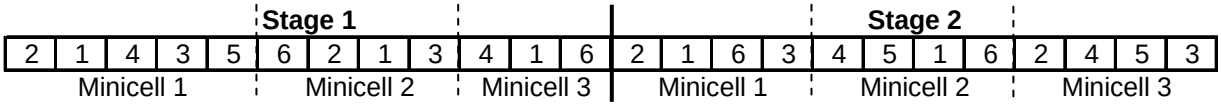


Figure 4-12: Sequence obtained by MFGA for the new Strategy

From the above results it can be observed that the MFGA performs exceedingly well for this problem. All the strategies produce results which are superior to the result given by the FL heuristic.

4.3.3 Eight Product Variant Problem

A third test problem with 10 products and eight different product variants was considered.

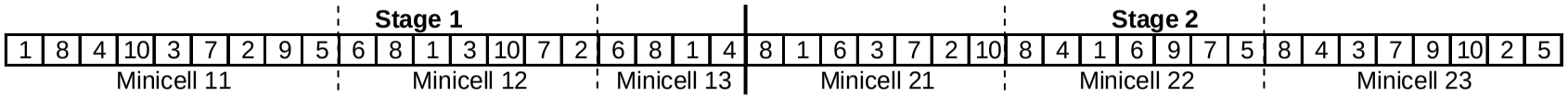
Table 4-16 gives the product structure for all 10 jobs. Products 3 and 10 have same composition while products 5 and 9 have similar composition. Hence, though there are 10 products, only eight different variants exist.

Table 4-16: Product ID; product structure

	Feature 1	Feature 2	Feature 3
Product 1	1	2	1
Product 2	2	1	2
Product 3	2	1	1
Product 4	1	2	2
Product 5	1	1	2
Product 6	2	2	1
Product 7	1	1	1
Product 8	2	2	2
Product 9	1	1	2
Product10	2	1	1

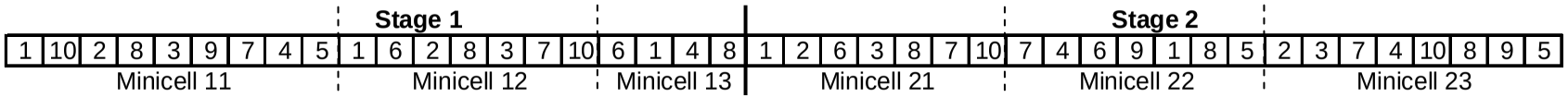
The minicell assignment is given in Table 4-15.

FL heuristic was first applied to solve the problem. The schedule obtained from the heuristic is represented by Figure 4-13. The problem was then solved by MFGA. All the four strategies (given in Table 4-2) were considered; population size of 20, mutation probability of 0.1 and 100 generations were chosen for experimentation. The results obtained from the MFGA for strategies 1-4 are represented by Figures 4-14 to 4-17. The result obtained for the strategy with X % = 100, Y % = 0, Z % = 0 is given by Figure 4-18.



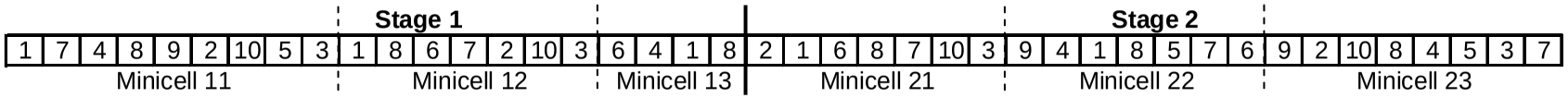
F_{avg} = 135 minutes

Figure 4-13: Sequence obtained by FL heuristic



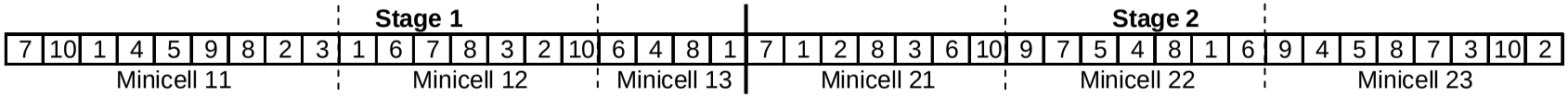
F_{avg} = 187 minutes

Figure 4-14: Sequence obtained by MFGA for Strategy 1



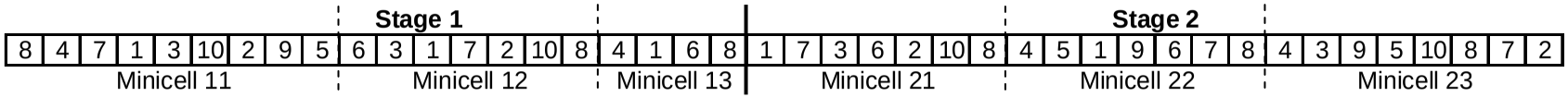
F_{avg} = 180 minutes

Figure 4-15: Sequence obtained by MFGA for Strategy 2



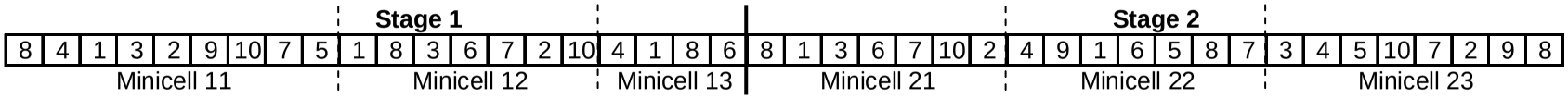
F_{avg} = 185 minutes

Figure 4-16: Sequence obtained by MFGA for Strategy 3



F_{avg} = 192 minutes

Figure 4-17: Sequence obtained by MFGA for Strategy 4



F_{avg} = 173 minutes

Figure 4-18: Sequence obtained by MFGA for Strategy X % = 100, Y % = 0, Z % = 0

From the above results it can be observed that the FL heuristic performed very well for this problem. There is considerable difference in the solution obtained from MFGA and the heuristic value.

CHAPTER 5

5 DISCUSSION AND CONCLUSION

Minicells have been identified as a potential manufacturing system for mass customization. The need to develop better scheduling approach to improve the performance of minicells further has been pointed out in previous research. The minimum flow time schedule genetic algorithm (MFGA) for minicells was designed in this research for effective scheduling of jobs in a minicell configuration. The experimentation and results for the MFGA were presented in Chapter 4. A discussion of results and directions for future research are presented in this section.

5.1 Summary of Results

Results presented in chapter 4 reveal that the MFGA provides better results compared to FL heuristic only for test problem 2. There were two instances when the MFGA gave a better solution than the heuristic in test problem 1 and none for test problem 3. The potential reasons for this variation are discussed in the following sub-sections for each test problem separately.

5.1.1 Test Problem 1

Extensive experimentation was done for this test problem. All the four strategies were tested with different population sizes, mutation probabilities and number of generations. The results for all the possible combinations are presented and additional work (scatter plots) was done to identify the best combination of input parameters for which the MFGA will give superior results. As noticed earlier, of all the combinations, the MFGA generated a better solution than the heuristic in only two cases. The performance of the algorithm on the whole was not comparable to that of the heuristic. There was a marginal difference in the fitness values obtained with the two methods.

A small test problem with only 10 generations was developed to study the behavior of the MFGA further. All the fitness values generated for each chromosome in a generation for all generations were tabulated. A box plot was then developed and is shown in Figure 5-1.

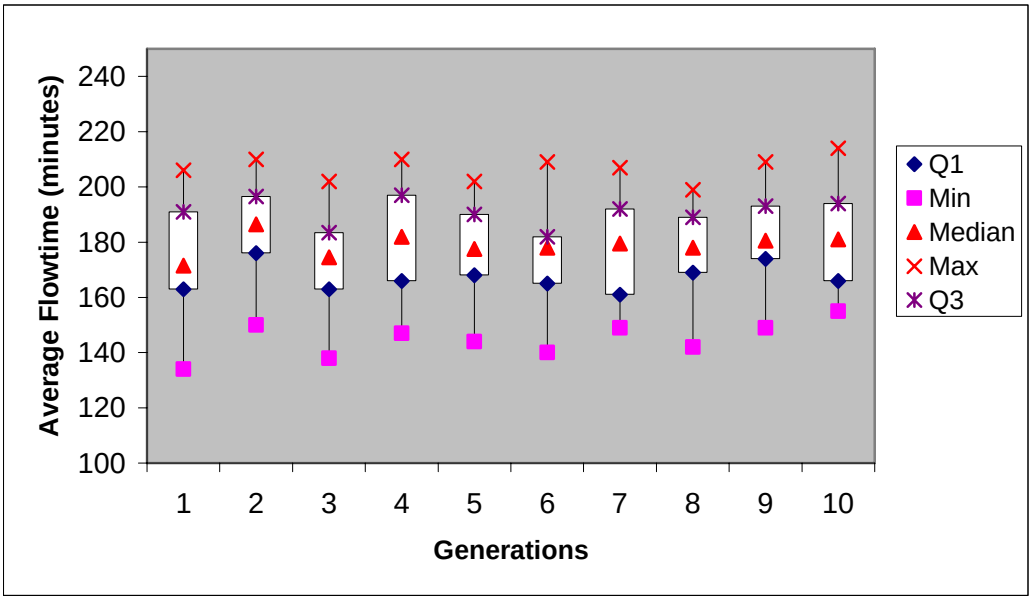


Figure 5-1: Box plot for the test problem

In the above figure Q1 and Q3 represent the lower quartile and upper quartile respectively. It is expected that the fitness function value for subsequent generations would reduce due to convergence. Also, as the GA begins to settle towards exploiting a locality in the solution space, the inter-quartile range in the box plot should likely shorten. However, the results from the box plots show that the fitness values from the MFGA are not converging as desired. The Q1 and minimum values are expected to decrease as the number of generations increase. Also the interquartile range shows variability in the population created by crossover.

The above observations indicate that the solution candidates generated as the algorithm advances are possibly not converging as desired. This raises the concern as to whether the multi-chromosome strategy used in the MFGA is introducing more variability than generating solutions that converge towards the best job sequence. The division of the population into three groups before the genetic operations are performed could be one reason for the observed behavior. A new test problem with all the chromosomes assigned to only one group (X % = 100,

Y % = 0, Z % = 0) was developed. A box plot was then generated for this problem as shown in Figure 5-2.

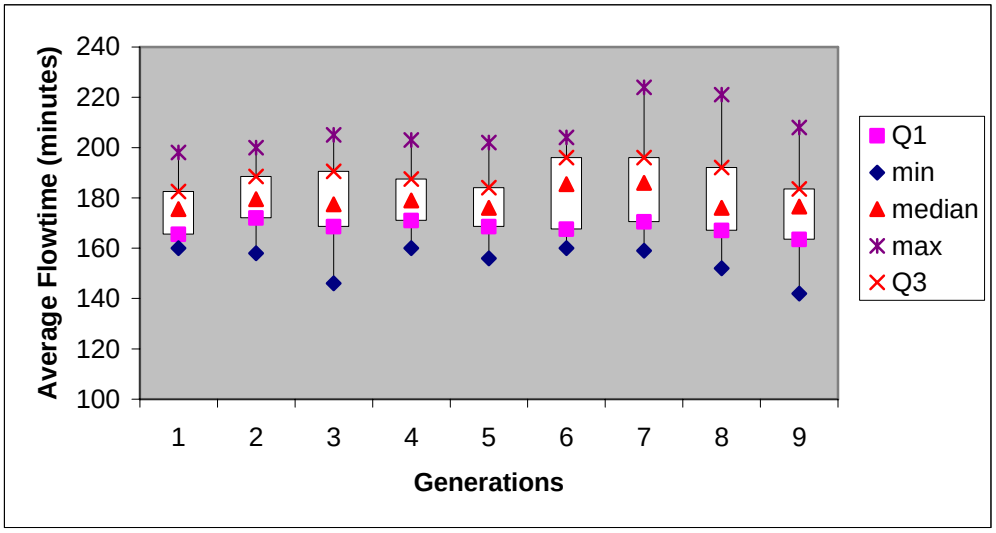


Figure 5-2: Box plot for the test problem (X % = 100, Y % = 0, Z % = 0)

From the above figure it can be observed that the interquartile range is considerably less than that noticed in Figure 5-1 in most cases and also the variation in the value is low. However, still the MFGA does not appear to converge as expected. The fitness value decreases for the first three generations but then the value keeps fluctuating till 7th generation from where it again gradually converges till end of 9th generation. The quality of the solution candidates in this test problem appears to be better than the previous test problem.

5.1.2 Test Problem 2

For the test problem 2, MFGA had solutions which were better than the FL heuristic in all four strategies. The box plot shown in Figure 5-3 is generated to observe the solution candidate evolution through generations. Samples of 11 consecutive generations were selected from Strategy 2 to generate this plot. It can be observed from the figure that except for some few cases, the solutions are converging towards the end.

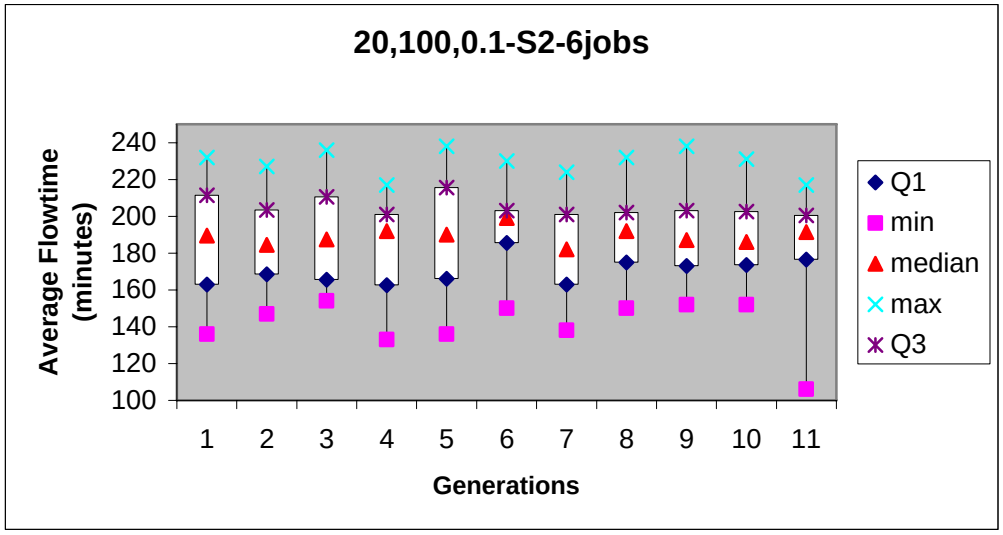


Figure 5-3: Box plot for the test problem 2

5.1.3 Test Problem 3

The FL heuristic outperformed the MFGA with this test problem. The average flow time value obtained from the heuristic was much better than what MFGA has given. The box plot shown in Figure 5-4 is generated to observe the solution candidate evolution through generations. Samples of 11 consecutive generations were selected from Strategy 2 to generate this plot. The results, though similar, appear to be somewhat better than that for problem 1. However, the fact that the MFGA did not give a single solution better than the FL heuristic demands further investigation.

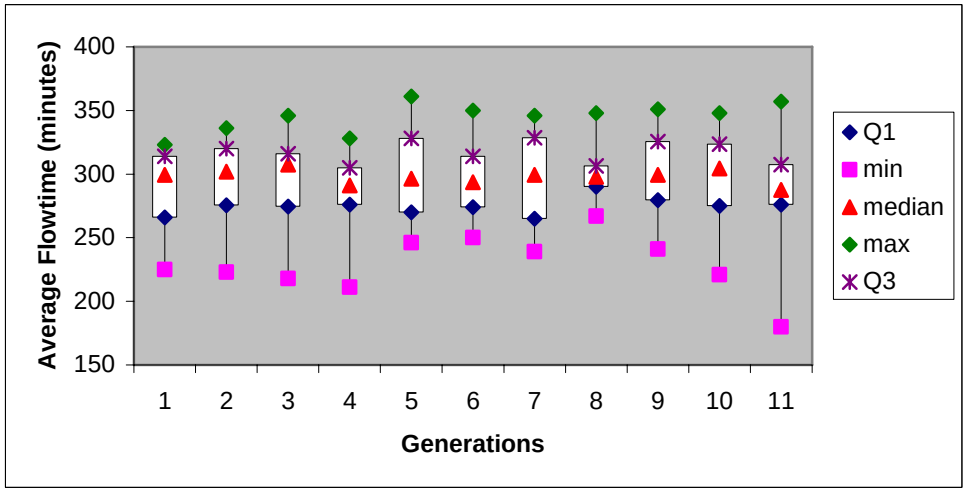


Figure 5-4: Box plot for the test problem 3

5.2 Observations

The FL heuristic has performed well for the first and third test problems while the MFGA results for second test problem were good. It must be noted that while the first and third test problems had repetitive product variants, the second test problem had 6 jobs that were all different product variants with no repetition. This can be one of the reasons for not so effective performance of the MFGA. More detailed experimentation was followed to evaluate causes for the variation.

The test problem with all the solution candidates in one single group (X % = 100, Y % = 0, Z % = 0) has produced better results compared to other strategies tested. However, there is no considerable improvement in the fitness values obtained and further inquiry is needed to analyze the cause for the poor performance of the MFGA.

When the multi-chromosome crossover strategy was first conceptualized for the capacitated lot sizing problem, the chromosome was divided into several parts, each part representing a product. The genetic operations were performed on these part chromosomes and the best ones were combined. Here the optimization of each part chromosome before combining them together led to superior results since the composition of one part chromosome had no effect on any other; each part represented a different product. But in the MFGA, each part chromosome represents a minicell. The schedule developed in one part chromosome has an impact on the effectiveness of the succeeding part chromosome. Hence, retrospectively, even though the part chromosomes obtained after performing genetic operations individually represent the best solution, the final combination may probably not lead to the best result.

As described in Chapter 3, a single chromosome is created initially with the available information and the rest of the population size is created from this single chromosome by randomizing the genes positions. A small test problem was developed to check the variability being introduced in the initial population creation. The problem was repeated three times with a population size of 5. The initial population generated in each problem was then compared. It is observed that in generating the initial population, the MFGA follows a certain pattern and, therefore, the chromosomes are not significantly different. This could have led the MFGA not to

explore the solution space effectively. Therefore, it is likely that the approach used to generate the initial population could have been a contributor to the results observed.

5.3 Conclusion and Future work

In this thesis, MFGA was developed to minimize the average flow time of the jobs in this minicell configuration. A new crossover strategy, previously shown to be superior to the classical crossover, was applied in the MFGA. Although the MFGA gave superior results in some cases, it was not comparable with others. The analyses reveal several more approaches that could be explored to improve the effectiveness of the MFGA in finding optimal solution that minimizes average flow time in a minicell configuration. A more efficient method of searching the entire solution space can be included in the MFGA. Also a better strategy can be developed to combine the part chromosomes to form the final solution candidate.

APPENDIX I

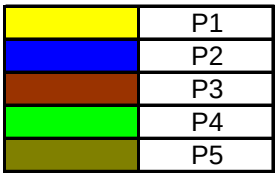
FL Heuristic

The methodology to solve a scheduling problem using the FL heuristic has already been discussed in Chapter 4 on page 40. The working procedure for test problem 1 is presented below.

Stage 1:

Minicell 1 (Machines: M1, M2)

Jobs to be processed: {1, 2, 3, 4, 5}



Mean processing times

P1-14

P2-19

P3-14

P4-19

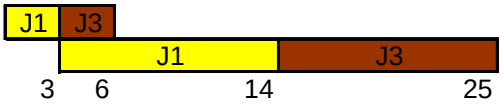
P5-14

Ascending order of jobs: P1<P3<P5<P2<P4

K=1, S= {1}

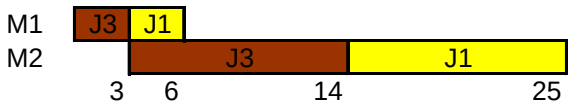
K=2, S= {1-3} {3-1}

{1-3}



F_{avg}=39

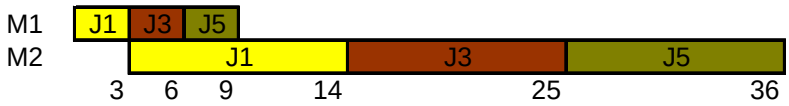
{3-1}



F_{avg}=39

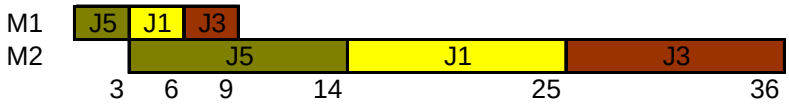
K=3, S= {5-1-3} {1-5-3} {1-3-5}

{1-3-5}



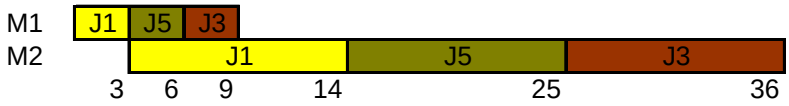
F_{avg}=75

{5-1-3}



F_{avg}=75

{1-5-3}



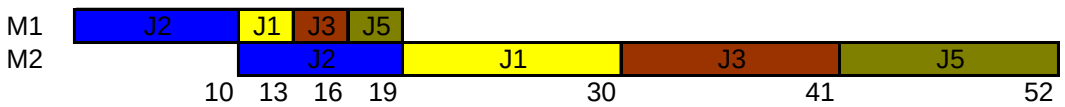
F_{avg}=75

Select {1-3-5}

Alternative sequences {3-1-5}, {5-3-1}, {1-5-3}. All these sequences result in the same fitness value. Hence, {1-3-5} can be retained as the best sequence.

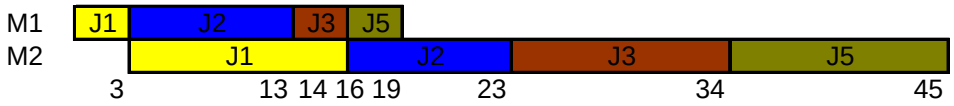
K=4, S= {2-1-3-5}, {1-2-3-5}, {1-3-2-5}, {1-3-5-2}

{2-1-3-5}



F_{avg}=142

{1-2-3-5}



F_{avg}=116

{1-3-2-5}

F_{avg}=118

{1-3-5-2}

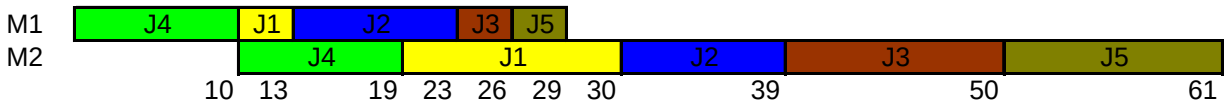
F_{avg}=120

Select {1-2-3-5}

Alternative sequences: {2-1-3-5}, {3-2-1-5}, {5-2-3-1}, {1-3-2-5}, {1-5-3-2}, {1-2-5-3}. Some sequences are shown above and rests are similar to {1-2-3-5}. Hence, {1-2-3-5} can be retained as the best sequence.

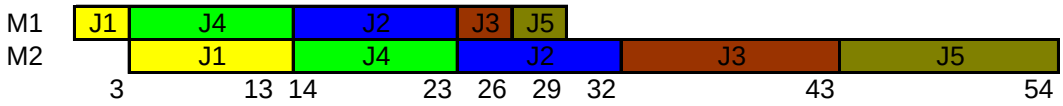
K=5, S= {4-1-2-3-5}, {1-4-2-3-5}, {1-2-4-3-5}, {1-2-3-4-5}, {1-2-3-5-4}

{4-1-2-3-5}



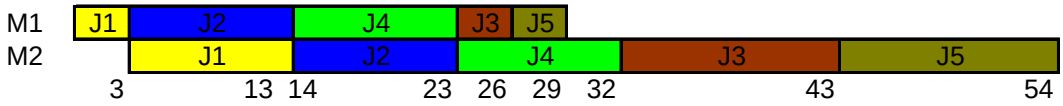
F_{avg}=199

{1-4-2-3-5}



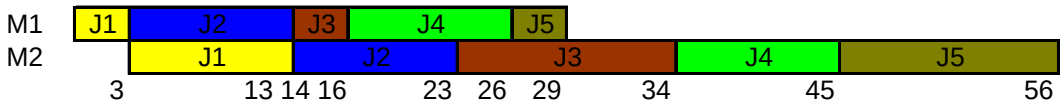
F_{avg}=166

{1-2-4-3-5}



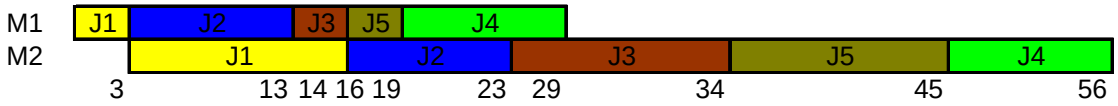
F_{avg}=166

{1-2-3-4-5}



F_{avg}=172

{1-2-3-5-4}

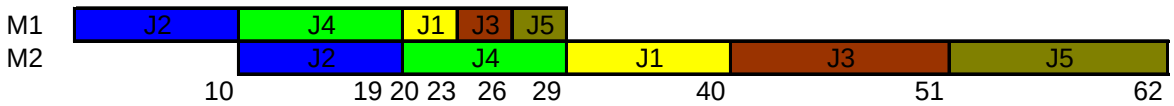


F_{avg}=170

Select {1-4-2-3-5}

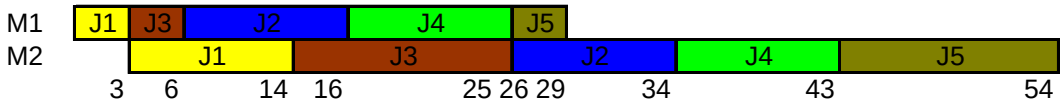
Alternative sequences: {4-1-2-3-5}, {2-4-1-3-5}, {3-4-2-1-5}, {5-4-2-3-1}, {1-2-4-3-5}, {1-3-2-4-5}, {1-5-2-3-4}, {1-4-3-2-5}, {1-4-5-3-2}, {1-4-2-5-3}.

{2-4-1-3-5}



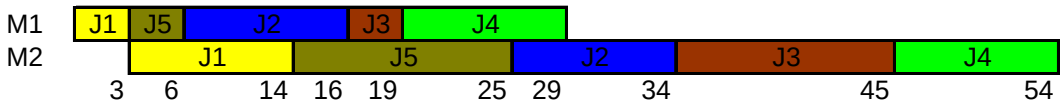
F_{avg}=201

{1-3-2-4-5}



F_{avg}=170

{1-5-2-3-4}



F_{avg}=172

Best sequence: {1-4-2-3-5}

Processing times of jobs from minicell 1:

P1-14

P2-32

P3-43

P4-23

P5-54

Minicell 2 (Machines: M1, M2)

Jobs to be processed: {1, 2, 3, 4, 5}

	P1
	P2
	P3
	P4
	P5

Mean processing times

P1-10

P2-6

P3-10

P4-6

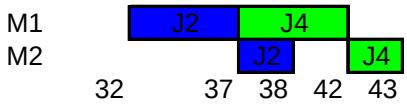
P5-10

Ascending order of jobs: P2<P4<P1<P3<P5

K=1, S= {2}

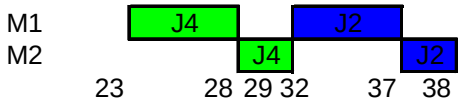
K=2, S= {2-4}, {4-2}

{2-4}



F_{avg}=81

{4-2}

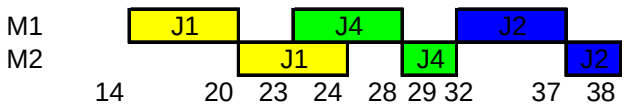


F_{avg}=67

Select {4-2}

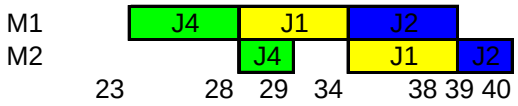
K=3, S= {1-4-2}, {4-1-2}, {4-2-1}

{1-4-2}



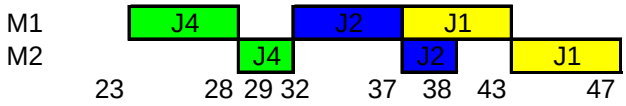
F_{avg}=91

{4-1-2}



F_{avg}=107

{4-2-1}

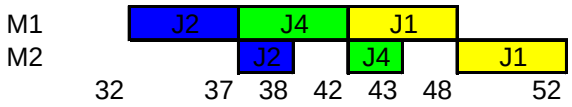


F_{avg}=114

Select {1-4-2}

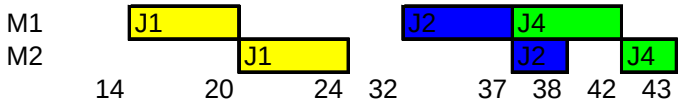
Alternate sequences: {4-1-2}, {2-4-1}, {1-2-4}

{2-4-1}



F_{avg}=133

{1-2-4}

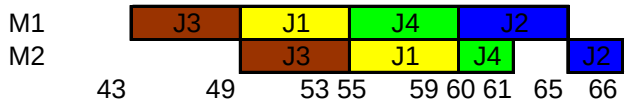


F_{avg}=105

Best sequence: {1-4-2}

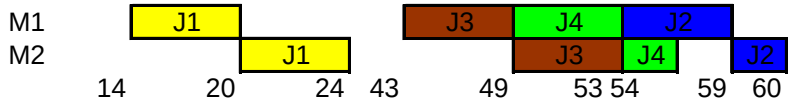
K=4, S= {3-1-4-2}, {1-3-4-2}, {1-4-3-2}, {1-4-2-3}

{3-1-4-2}



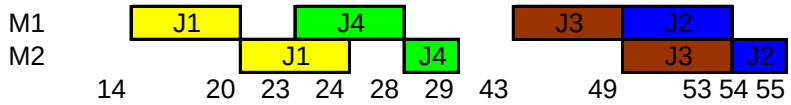
F_{avg}=239

{1-3-4-2}



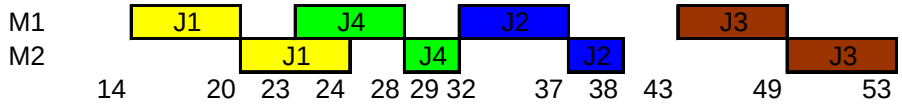
F_{avg}=192

{1-4-3-2}



F_{avg}=161

{1-4-2-3}

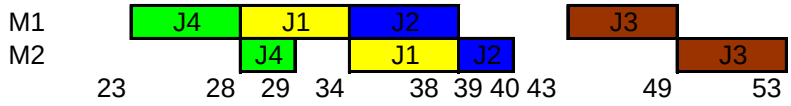


F_{avg}=144

Select {1-4-2-3}

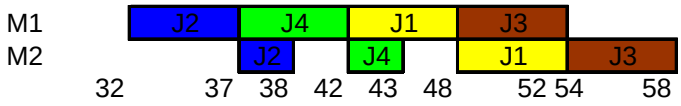
Alternate sequences: {4-1-2-3} {2-4-1-3} {3-4-2-1} {1-2-4-3} {1-3-2-4} {1-4-3-2}

{4-1-2-3}



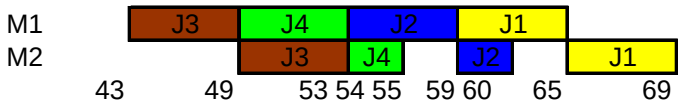
F_{avg}=160

{2-4-1-3}



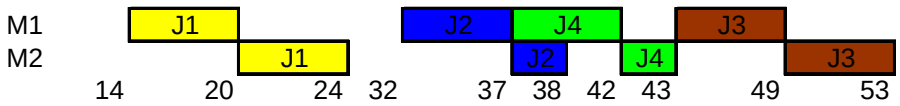
F_{avg}=191

{3-4-2-1}



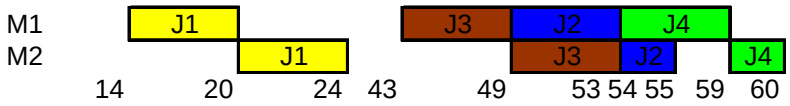
F_{avg}=237

{1-2-4-3}



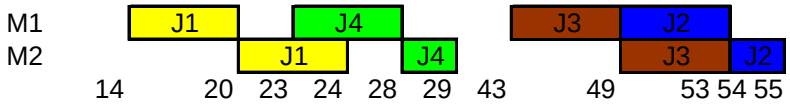
F_{avg}=158

{1-3-2-4}



F_{avg}=192

{1-4-3-2}

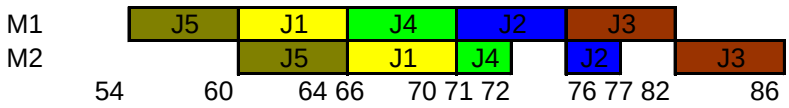


F_{avg}=161

Best sequence: {1-4-2-3}

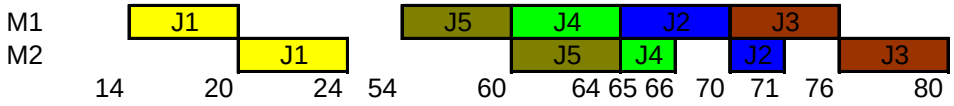
K=5, S= {5-1-4-2-3} {1-5-4-2-3} {1-4-5-2-3} {1-4-2-5-3} {1-4-2-3-5}

{5-1-4-2-3}



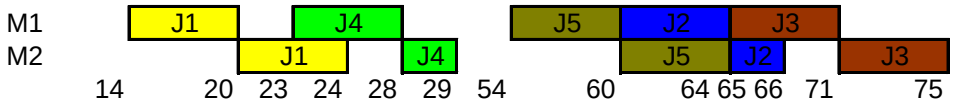
F_{avg}=369

{1-5-4-2-3}



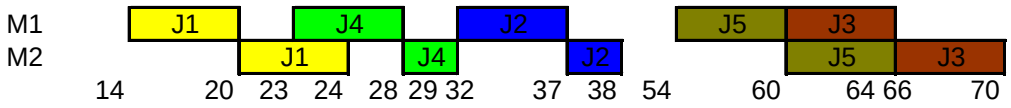
F_{avg}=305

{1-4-5-2-3}



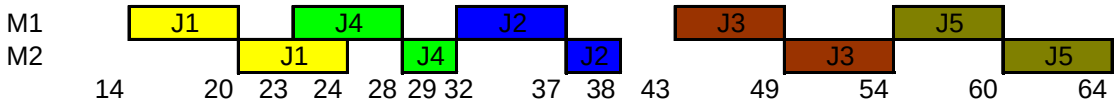
F_{avg}=258

{1-4-2-5-3}



F_{avg}=225

{1-4-2-3-5}

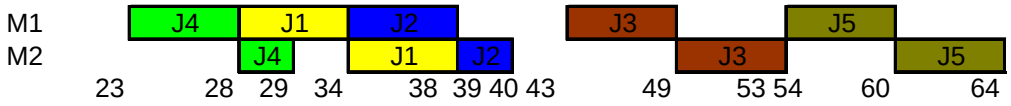


F_{avg}=209

Select: {1-4-2-3-5}

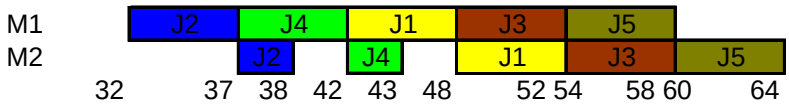
Alternate sequences: {4-1-2-3-5} {2-4-1-3-5} {3-4-2-1-5} {5-4-2-3-1} {1-2-4-3-5} {1-3-2-4-5} {1-5-2-3-4} {1-4-3-2-5} {1-4-5-3-2} {1-4-2-5-3}

{4-1-2-3-5}



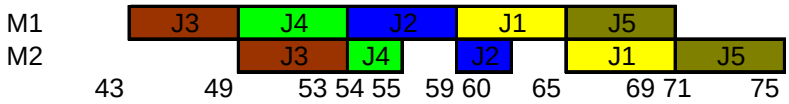
F_{avg}=224

{2-4-1-3-5}



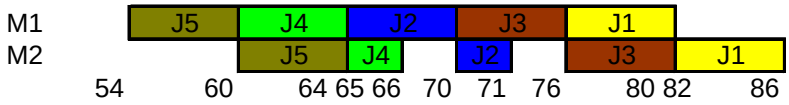
F_{avg}=255

{3-4-2-1-5}



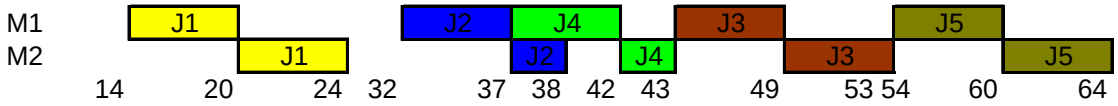
F_{avg}=312

{5-4-2-3-1}



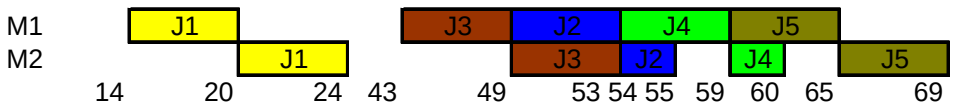
F_{avg}= 367

{1-2-4-3-5}



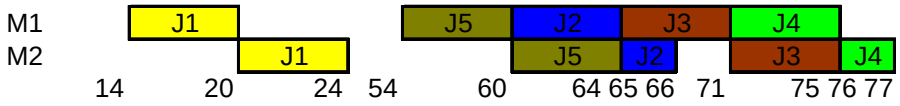
F_{avg}= 222

{1-3-2-4-5}



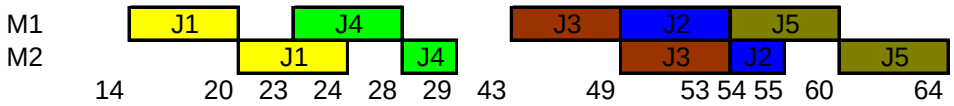
F_{avg}= 261

{1-5-2-3-4}



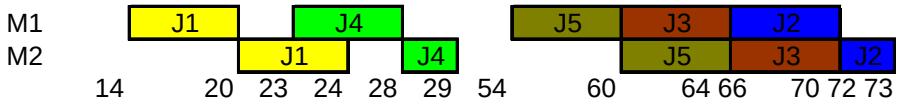
F_{avg}=306

{1-4-3-2-5}



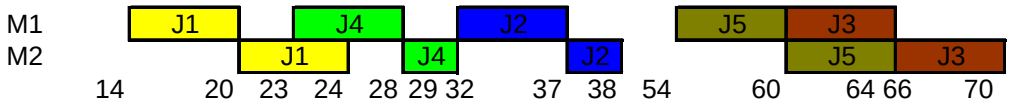
F_{avg}=225

{1-4-5-3-2}



F_{avg}=260

{1-4-2-5-3}



F_{avg}=225

Best sequence: {1-4-2-3-5}

Processing times of jobs from minicell 2:

P1-24

P2-38

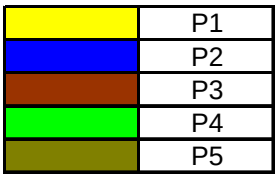
P3-54

P4-29

P5-64

Minicell 3 (Machines: M1, M2)

Jobs to be processed: {2, 4}



Mean processing times

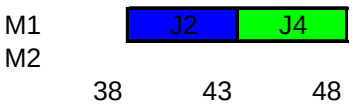
P2-5

P4-5

Ascending order of jobs: P2<P4

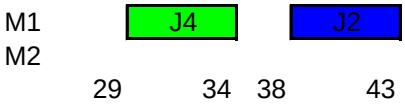
K=2, S= {2-4} {4-2}

{2-4}



F_{avg}=91

{4-2}



F_{avg}=77

Best sequence: {4-2}

Flow times from Stage 1:

P1-24

P2-43

P3-54

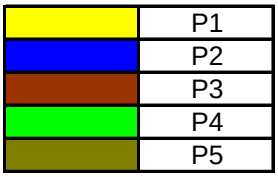
P4-34

P5-64

Stage 2

Minicell 1 (Machines: M3, M4, M5)

Jobs to be processed: {1, 2, 3, 4, 5}



Mean processing times

P1-16

P2-32

P3-16

P4-32

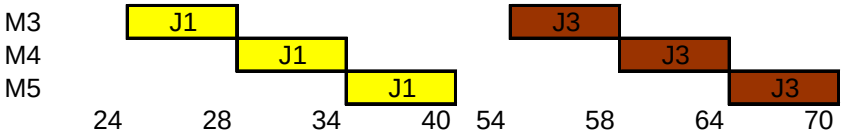
P5-16

Ascending order of jobs: P1<P3<P5<P2<P4

K=1, S= {1}

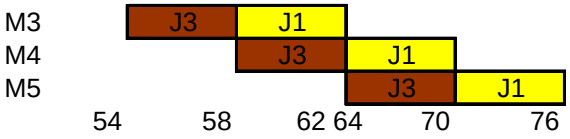
K=2, S= {1-3} {3-1}

{1-3}



F_{avg}=110

{3-1}

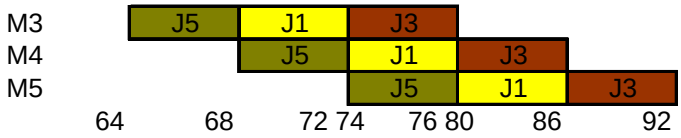


F_{avg}=146

Select {1-3}

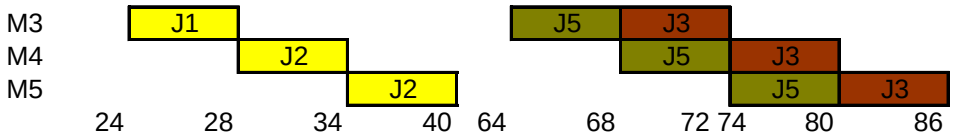
K=3, S= {5-1-3} {1-5-3} {1-3-5}

{5-1-3}



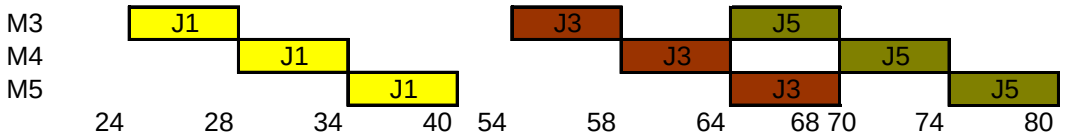
F_{avg}=258

{1-5-3}



F_{avg}=206

{1-3-5}



F_{avg}=190

Select {1-3-5}

Alternate sequences: {5-3-1} {1-5-3} {3-1-5}

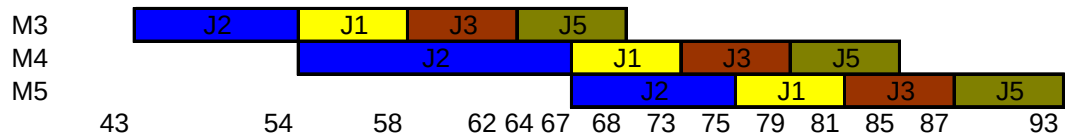
{3-1-5}

F_{avg}=228

Best sequence: {1-3-5}

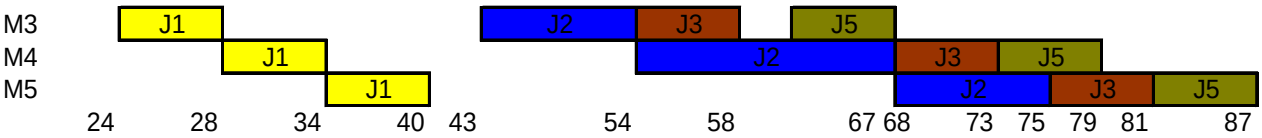
K=4, S= {2-1-3-5} {1-2-3-5} {1-3-2-5} {1-3-5-2}

{2-1-3-5}



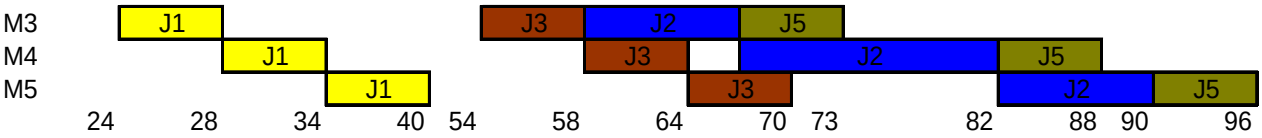
F_{avg}=336

{1-2-3-5}



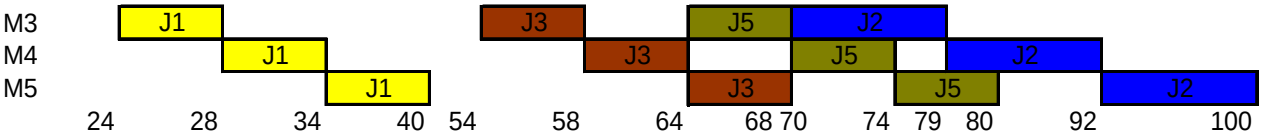
F_{avg}=283

{1-3-2-5}



F_{avg}=296

{1-3-5-2}

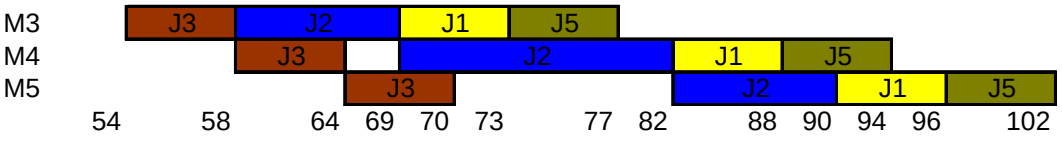


F_{avg}=290

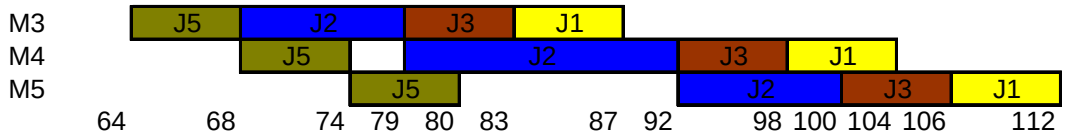
Select {1-2-3-5}

Alternate sequences: {2-1-3-5} {3-2-1-5} {5-2-3-1} {1-3-2-5} {1-5-3-2} {1-2-5-3}

{3-2-1-5}

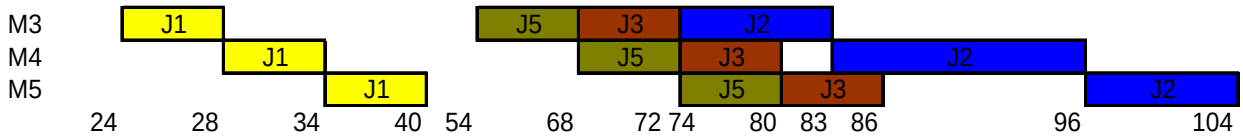


F_{avg}=358
{5-2-3-1}



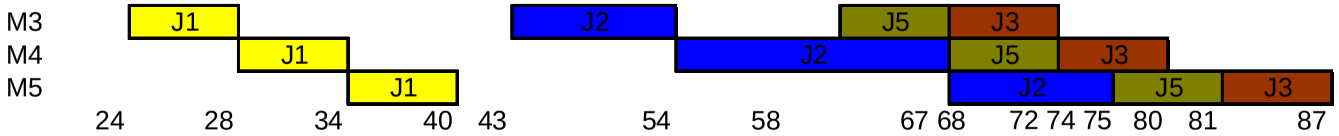
F_{avg}=398

{1-5-3-2}



F_{avg}=310

{1-2-5-3}

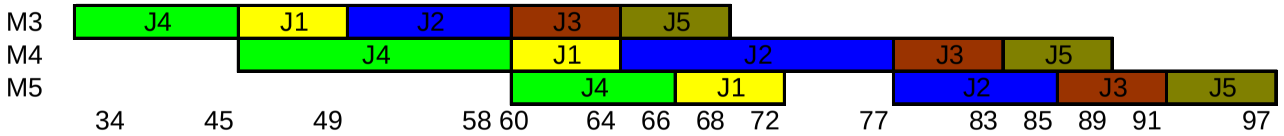


F_{avg}=283

Best sequence: {1-2-3-5}

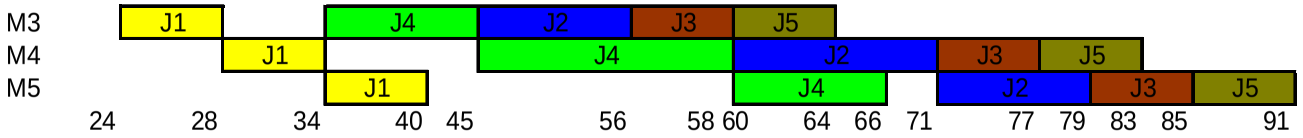
K=5, S= {4-1-2-3-5} {1-4-2-3-5} {1-2-4-3-5} {1-2-3-4-5} {1-2-3-5-4}

{4-1-2-3-5}



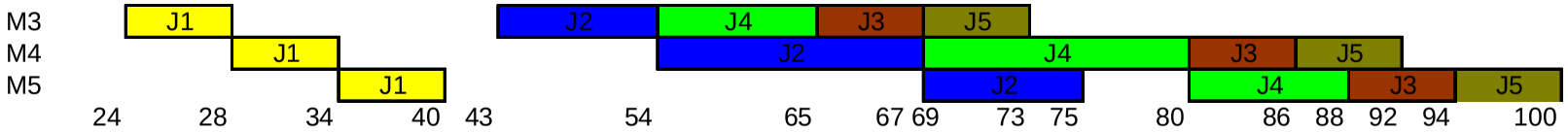
F_{avg}=411

{1-4-2-3-5}



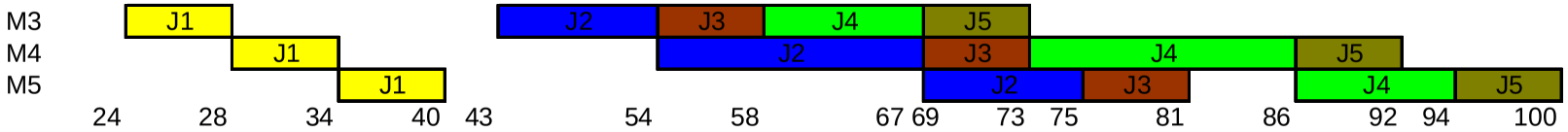
F_{avg}=361

{1-2-4-3-5}



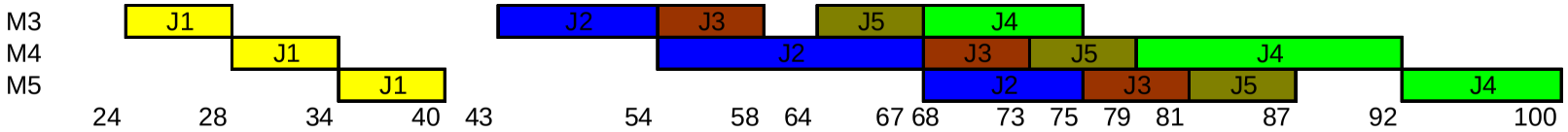
F_{avg}=397

{1-2-3-4-5}



F_{avg}=390

{1-2-3-5-4}

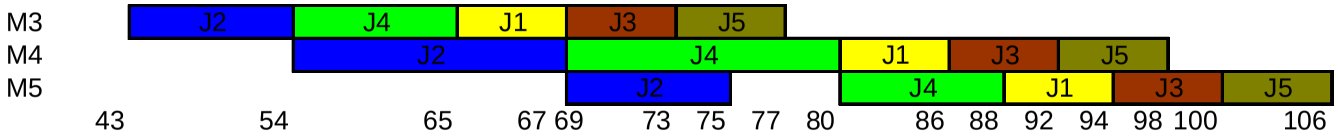


F_{avg}=383

Select: {1-4-2-3-5}

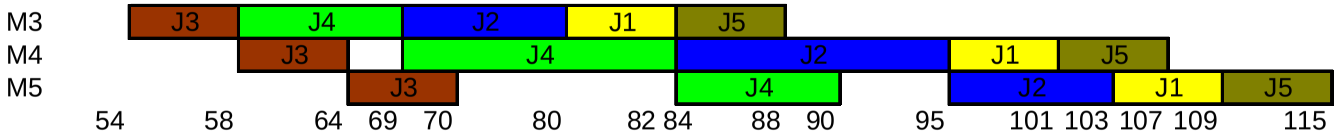
Alternate sequences: {4-1-2-3-5} {2-4-1-3-5} {3-4-2-1-5} {5-4-2-3-1} {1-2-4-3-5} {1-3-2-4-5} {1-5-2-3-4} {1-4-3-2-5} {1-4-5-3-2} {1-4-2-5-3}

{2-4-1-3-5}



F_{avg}=463

{3-4-2-1-5}

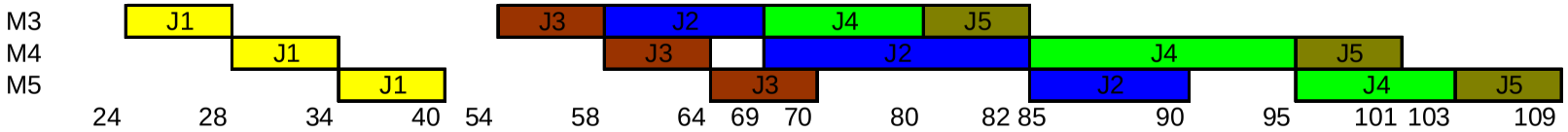


F_{avg}=487

{5-4-2-3-1}

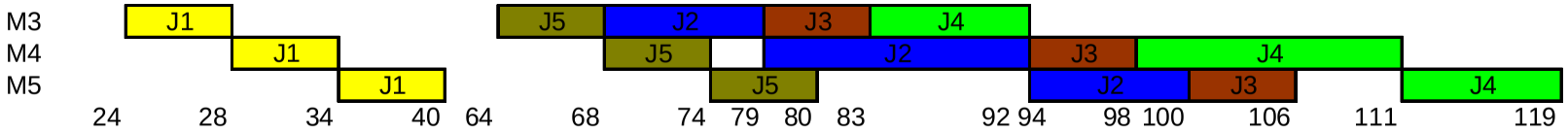
F_{avg}=537

{1-3-2-4-5}



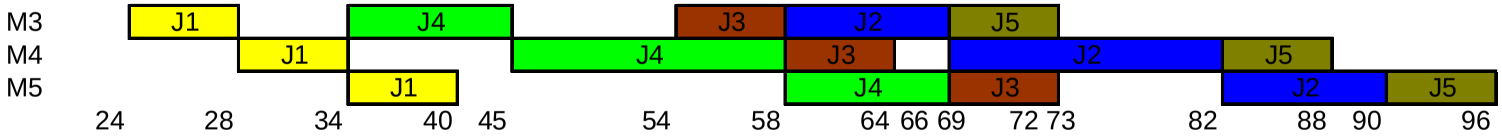
F_{avg}=412

{1-5-2-3-4}



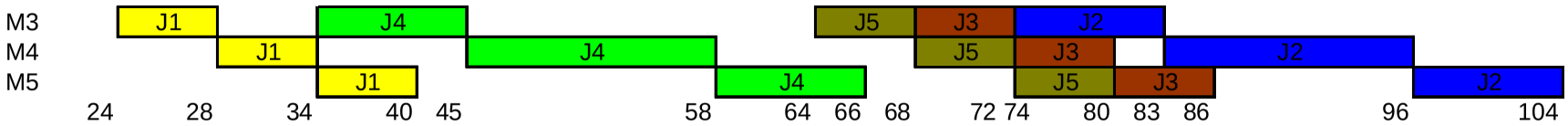
F_{avg}=445

{1-4-3-2-5}



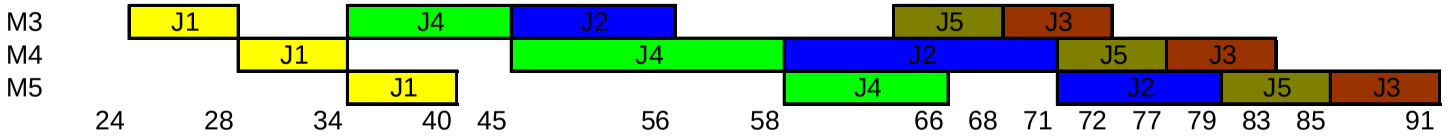
F_{avg}=364

{1-4-5-3-2}



F_{avg}=376

{1-4-2-5-3}



F_{avg}=359

Best sequence: {1-4-2-5-3}

Flow times from minicell 1:

P1-40

P2-79

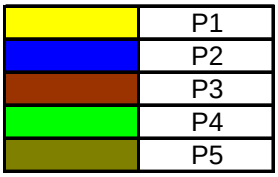
P3-91

P4-64

P5-85

Minicell 2(Machines: M3, M4, M5)

Jobs to be processed: {1, 2, 3, 4, 5}



Mean processing times

P1-24

P2-27

P3-24

P4-27

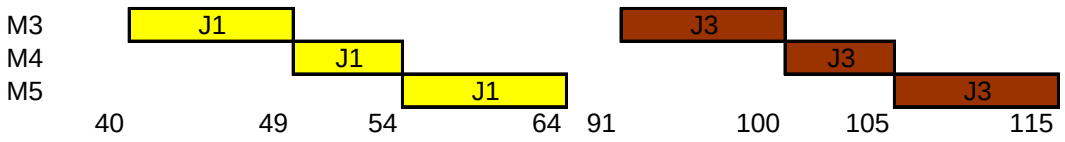
P5-24

Ascending order of jobs: P1<P3<P5<P2<P4

K=1, S= {1}

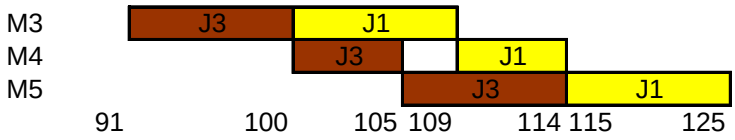
K=2, S= {1-3} {3-1}

{1-3}



F_{avg}=179

{3-1}

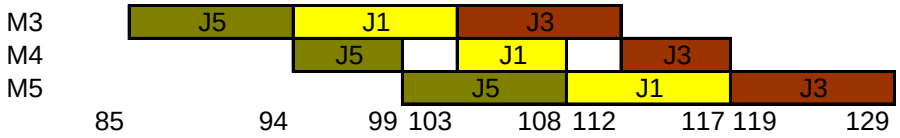


F_{avg}=240

Select {1-3}

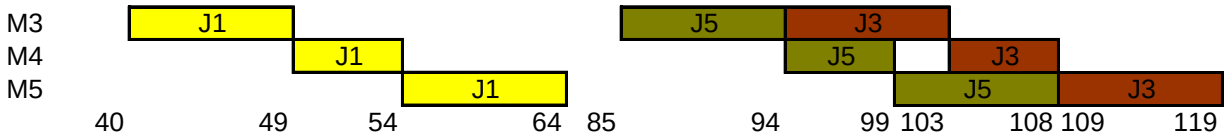
K=3, S= {5-1-3} {1-5-3} {1-3-5}

{5-1-3}



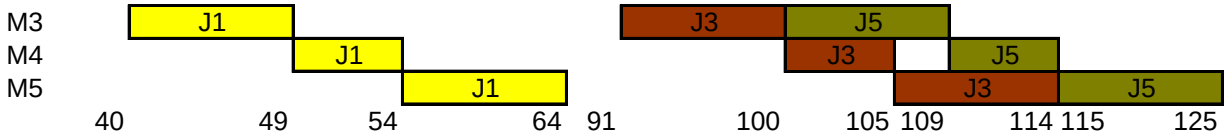
F_{avg}=357

{1-5-3}



F_{avg}=292

{1-3-5}

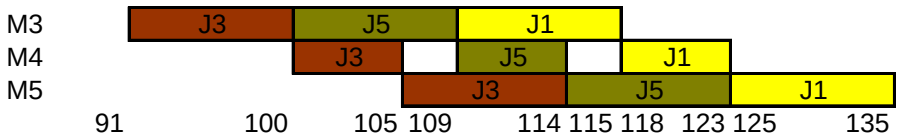


F_{avg}=304

Select {1-5-3}

Alternate sequences {5-1-3} {1-3-5} {3-5-1}

{3-5-1}

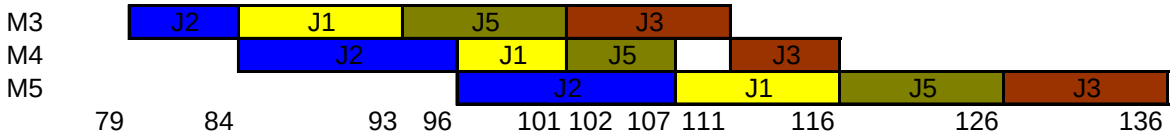


F_{avg}=375

Best sequence {1-5-3}

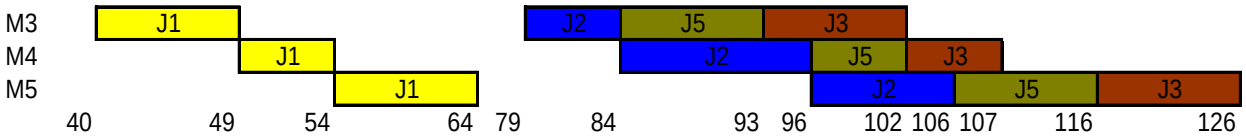
K=4, S= {2-1-5-3} {1-2-5-3} {1-5-2-3} {1-5-3-2}

{2-1-5-3}



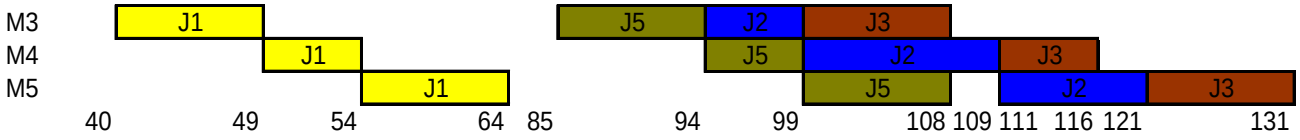
F_{avg}=484

{1-2-5-3}



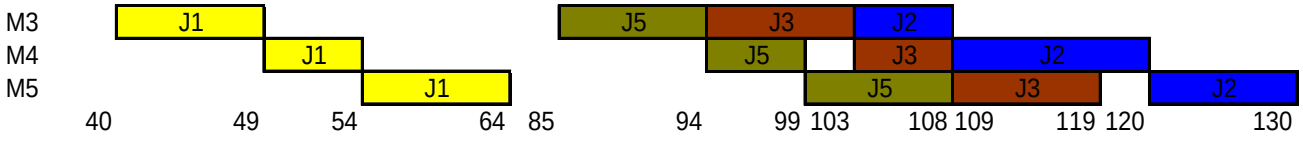
F_{avg}=412

{1-5-2-3}



F_{avg}=425

{1-5-3-2}

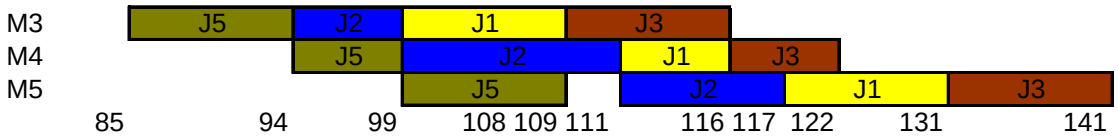


F_{avg}=422

Select {1-2-5-3}

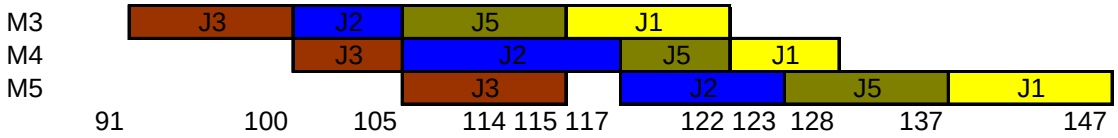
Alternate sequences {2-1-5-3} {5-2-1-3} {1-5-2-3} {1-3-5-2} {1-2-3-5} {3-2-5-1}

{5-2-1-3}



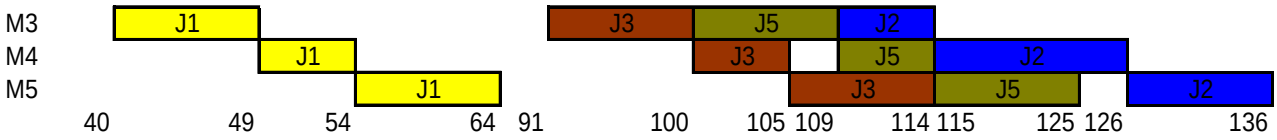
F_{avg}=502

{3-2-5-1}



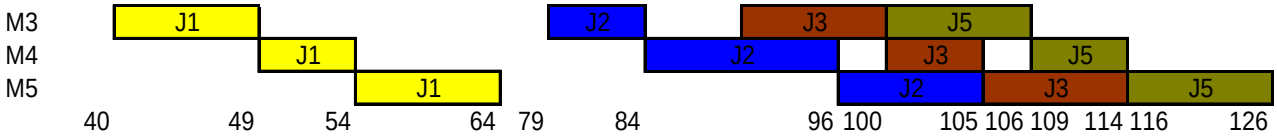
F_{avg}=526

{1-3-5-2}



F_{avg}=440

{1-2-3-5}

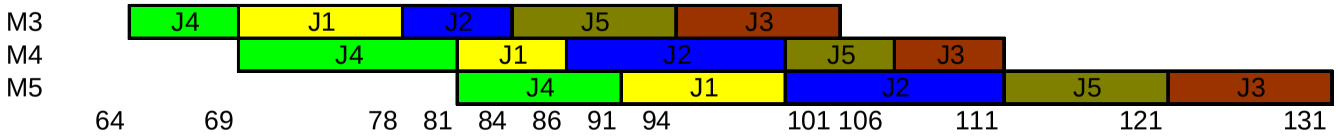


F_{avg}=412

Best sequence {1-2-5-3}

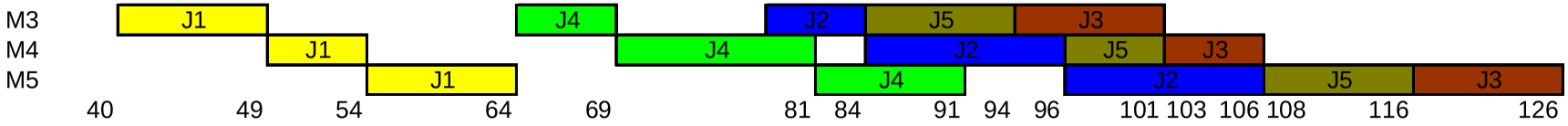
K=5, S= {4-1-2-5-3} {1-4-2-5-3} {1-2-4-5-3} {1-2-5-4-3} {1-2-5-3-4}

{4-1-2-5-3}



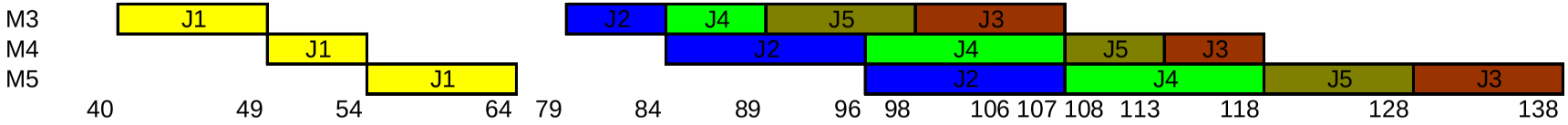
F_{avg}=526

{1-4-2-5-3}



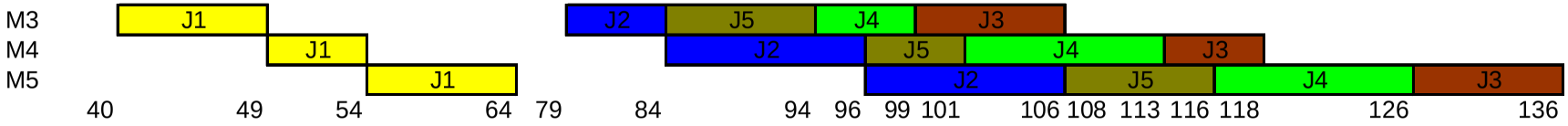
F_{avg}=503

{1-2-4-5-3}



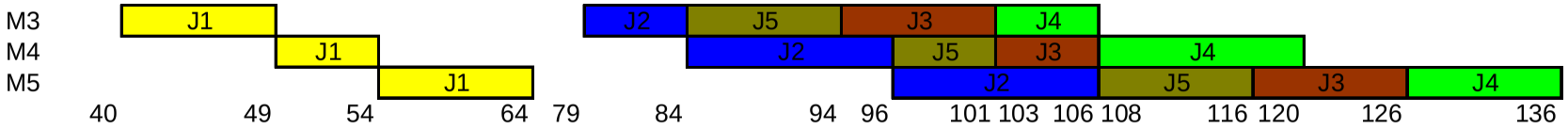
F_{avg}=554

{1-2-5-4-3}



F_{avg}=548

{1-2-5-3-4}

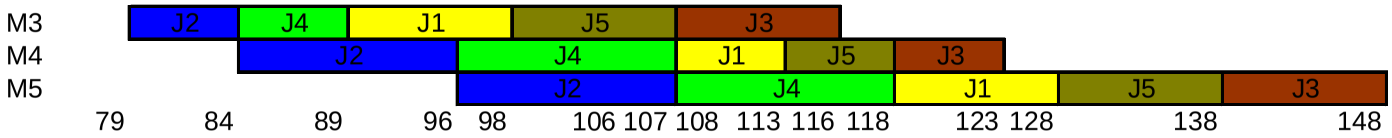


F_{avg}=548

Select {1-4-2-5-3}

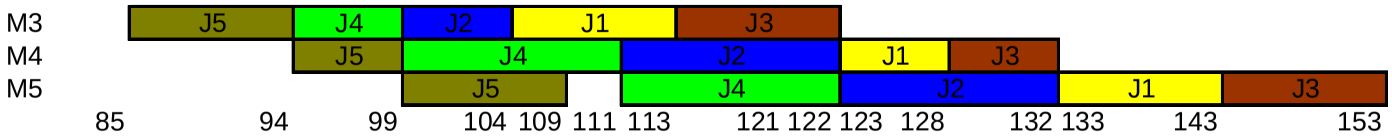
Alternate sequences: {4-1-2-5-3} {2-4-1-5-3} {5-4-2-1-3} {3-4-2-5-1} {1-2-4-5-3} {1-5-2-4-3} {1-3-2-5-4} {1-4-5-2-3} {1-4-3-5-2} {1-4-2-3-5}

{2-4-1-5-3}



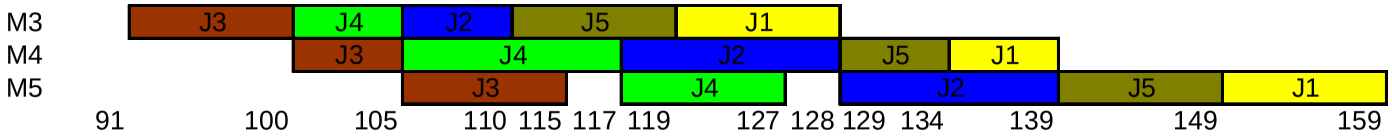
F_{avg}=438

{5-4-2-1-3}



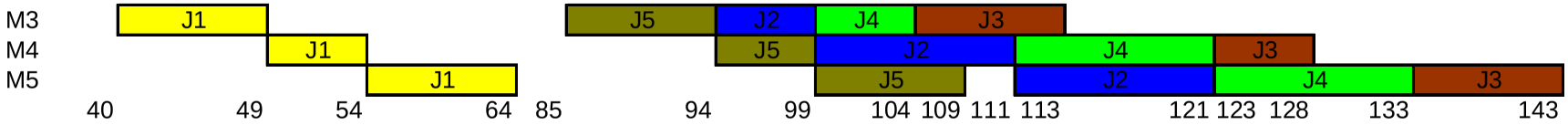
F_{avg}=659

{3-4-2-5-1}



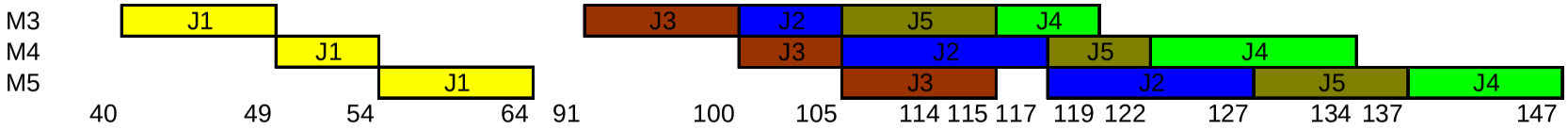
F_{avg}=689

{1-5-2-4-3}



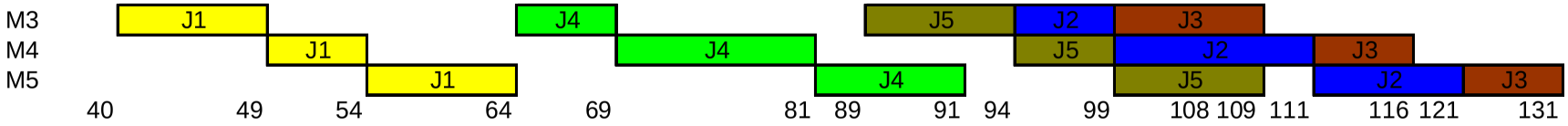
F_{avg}=570

{1-3-2-5-4}



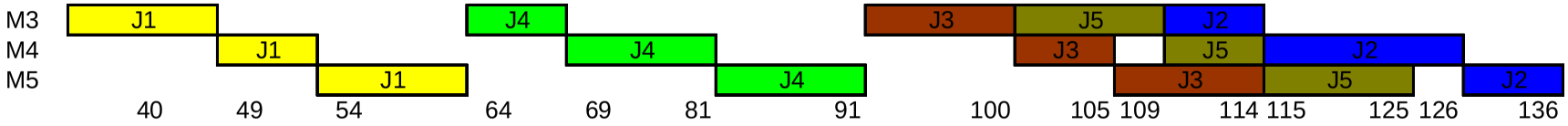
F_{avg}=590

{1-4-5-2-3}



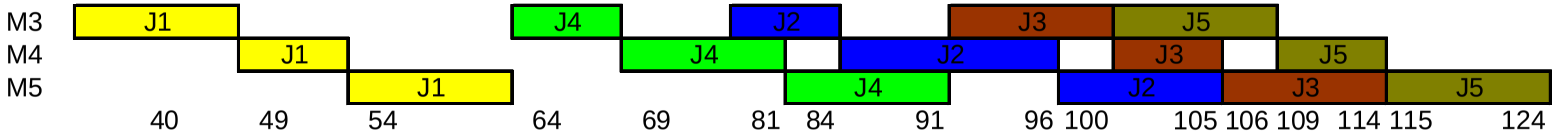
F_{avg}=542

{1-4-3-5-2}



F_{avg}=531

{1-4-2-3-5}



F_{avg}=500

Best sequence: {1-4-2-3-5}

Flow times from minicell 2:

P1-64

P2-106

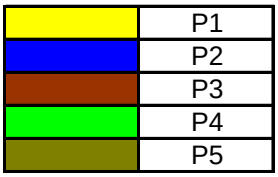
P3-115

P4-91

P5-124

Minicell 3(Machines: M3, M4, M5)

Jobs to be processed: {1, 3, 5}



Mean processing times

P1-8

P3-8

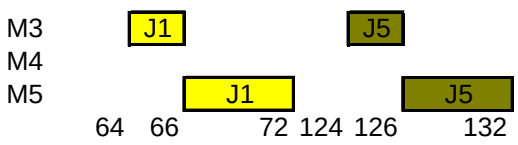
P5-8

Ascending order of jobs: P1<P5<P3

K=1, S= {1}

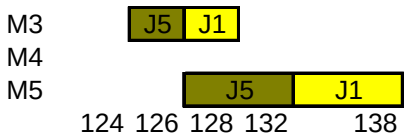
K=2, S= {1-5} {5-1}

{1-5}



F_{avg}=204

{5-1}

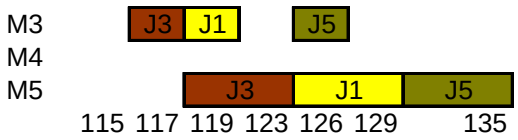


F_{avg}=270

Select {1-5}

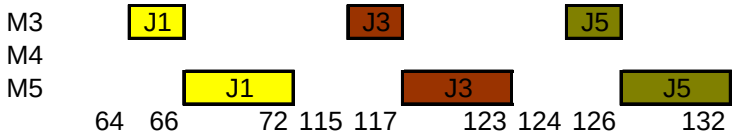
K=3, S= {3-1-5} {1-3-5} {1-5-3}

{3-1-5}



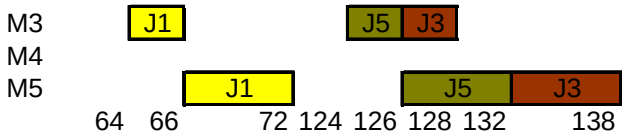
F_{avg}=387

{1-3-5}



F_{avg}=327

{1-5-3}

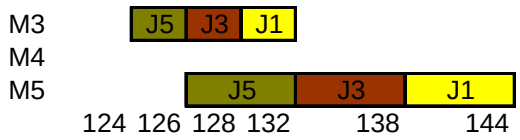


F_{avg}=342

Select {1-3-5}

Alternate sequences {3-1-5} {5-3-1} {1-5-3}

{5-3-1}



F_{avg}=414

Best sequence {1-3-5}

Flow times from minicell 3:

P1-72

P2-106

P3-123

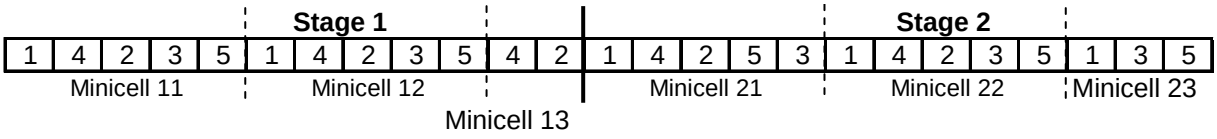
P4-91

P5-132

Total flow time = 542 units

Average flow time= 105 units

The final schedule given by FL heuristic is given below:

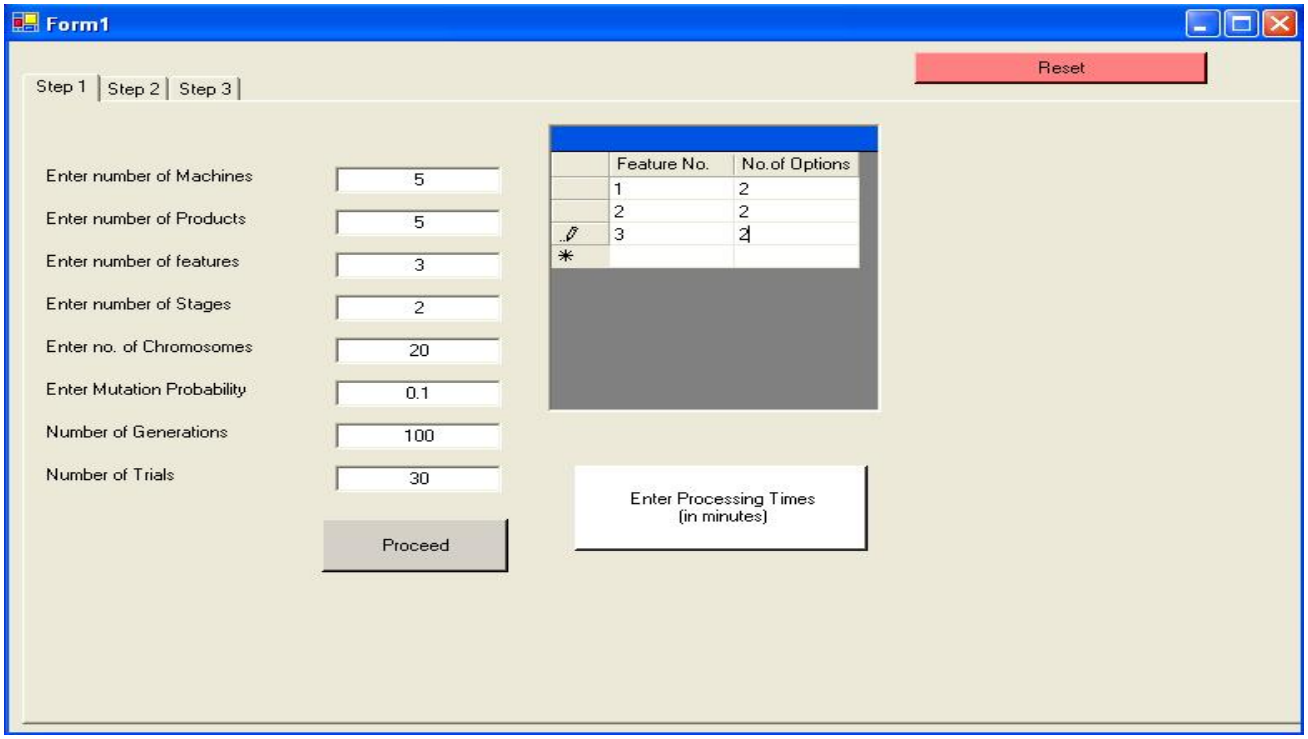


APPENDIX II

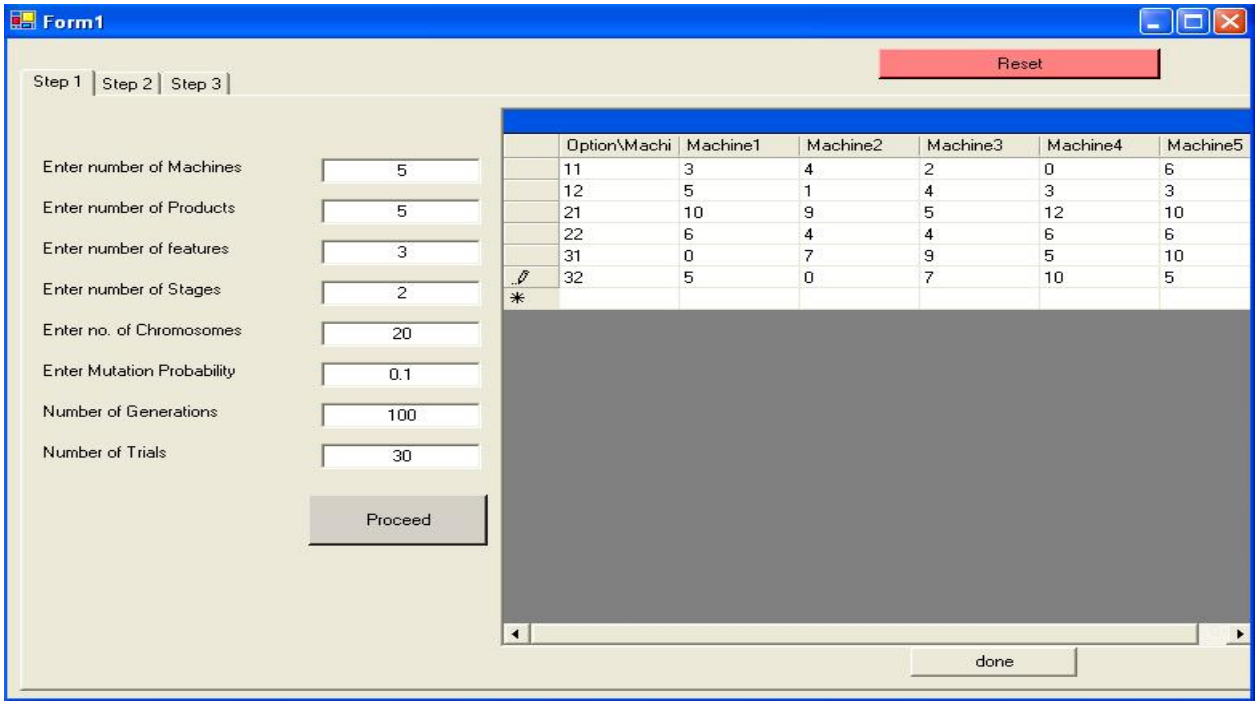
The MFGA Software Interface

The graphical interface used to gather the information required for the MFGA are presented here.

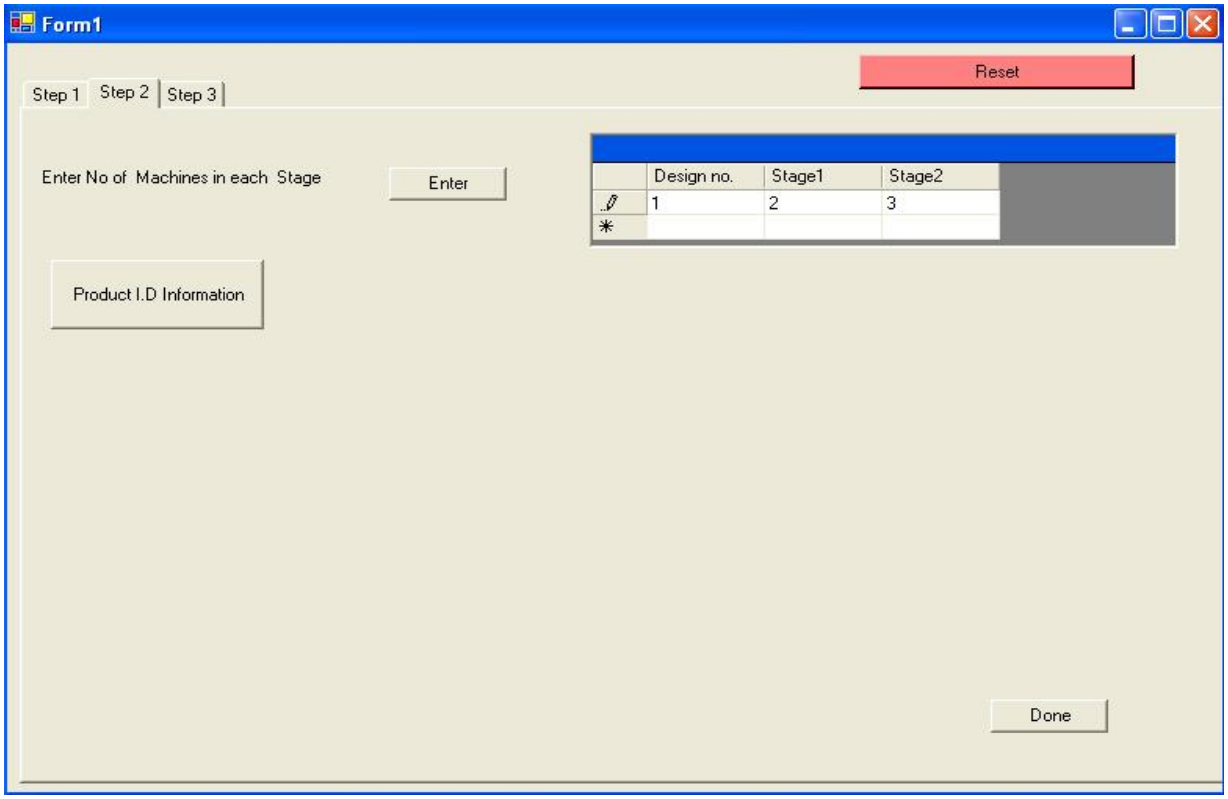
The following figure represents the graphical interface for input of initial parameters.



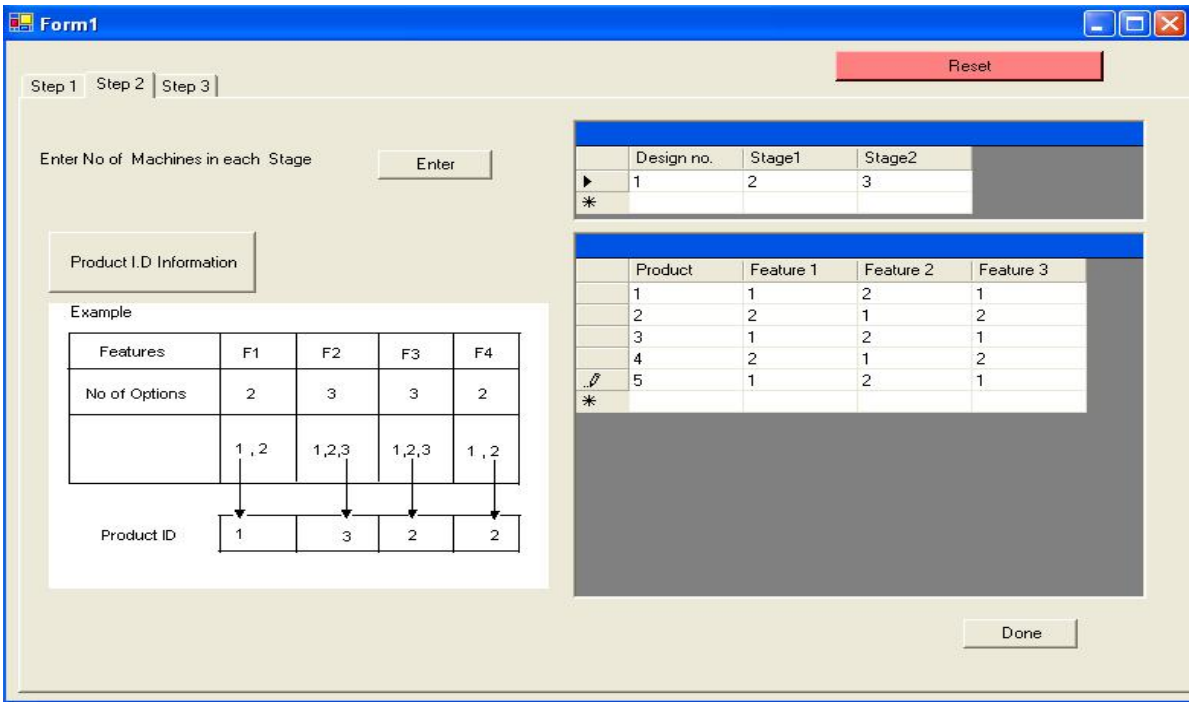
The following figure represents the graphical interface for entering processing times:



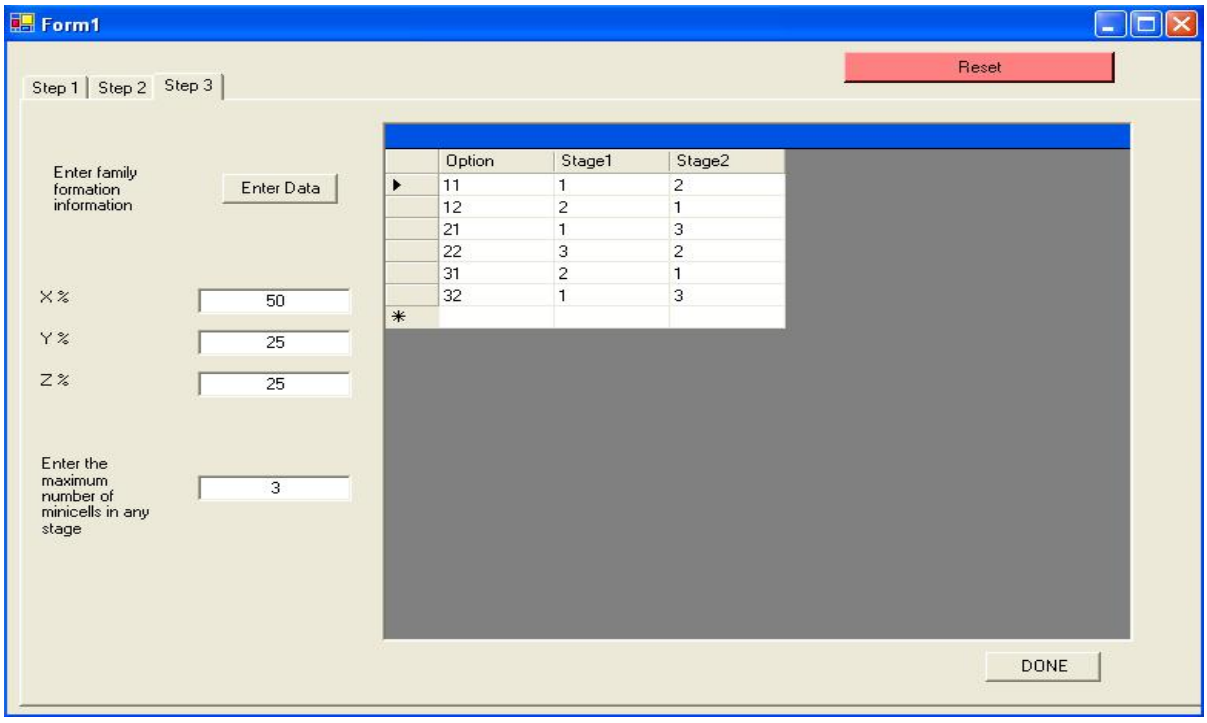
The following figure represents the graphical interface to obtain the information regarding the assignment of machines to stages:



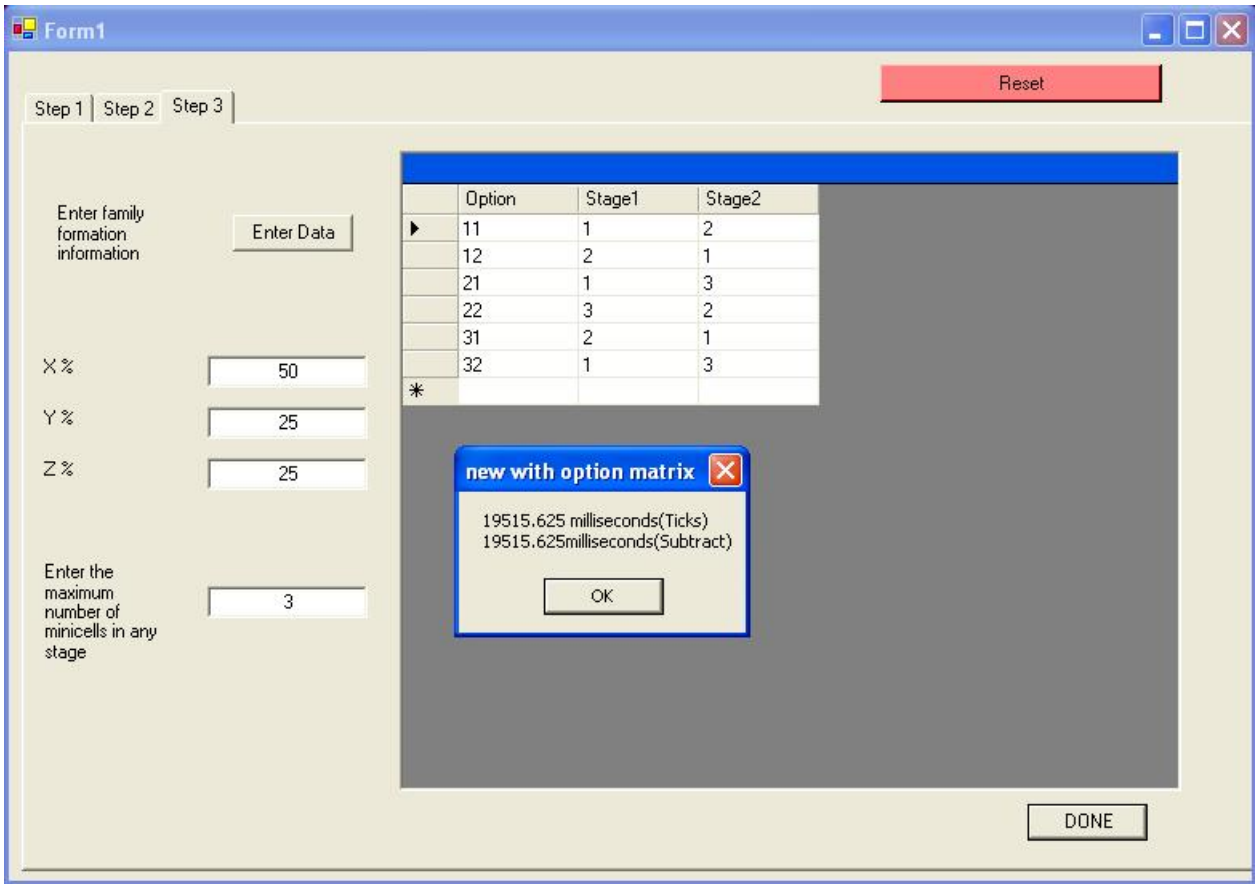
The following figure represents the graphical interface used to obtain the product structure (product I.D) for each product:



The following figure represents the graphical interface for the minicell assignment and the strategy selection:



The following figure represents the graphical interface for the end of the MFGA



REFERENCES

1. Reisman, A., Kumar, A., Motwani, J., (1997), "Flowshop Scheduling/Sequencing Research: A Statistical Review of the Literature, 1952-1994", *IEEE transactions on Engineering Management*, Vol.44, No. 3, August 1997, pp. 316-329.
2. Garey, M.R., and Johnson, D.S., (1976), "Strong NP-Completeness Results: Motivation, Examples, and Implications", *J. Assoc. Comp. Mach.*, 25 (1978), pp. 499-508.
3. Garey, M.R., Johnson, D.S., and Sethi, R., (1976), "The complexity of flow shop and job shop scheduling", *Mathematics of Operations Research*, 1, pp. 117-129.
4. Pine II, B.J., (1993), "Mass Customization: The New Frontier in Business Competition", Harvard Business School Press, Boston, MA.
5. Pine II, B.J., Victor, and Boynton, A.C., (1993), "Making Mass Customization Work", *Harvard Business Review*, September – October, 1993, pp. 108-116.
6. Piller, F.T., and Stotko, C.M., (2002), "Mass Customization: Four Approaches to Deliver Customized Products and Services with Mass Production Efficiency", *Proceedings to the 2002 IEEE International Engineering Management Conference*, Managing Technology for the New Economy, 18 - 20 August 2002, St. John's College, Cambridge, UK, pp. 773 – 778.
7. Badurdeen, F. F., and Masel, D. T., and Thuramalla, S., (2006), "A Modular Minicell Configuration for Mass Customization Manufacturing", Group Technology & Cellular Manufacturing International Conference, July 2-5, Grouingen, Netherlands.
8. Duray, R., (1997), "Mass customization Configurations: An empirical investigation of manufacturing practices of customization", Dissertation, The Ohio State University.
9. Gen, M. and Cheng, R., (1999), "Genetic Algorithms and Engineering Optimization", John Wiley and Sons, New York.
10. www.ie.bilkent.edu.tr/~lors/ie572/burhaneddin.pdf, accessed on 10/01/2006.
11. Garey, M., and Johnson, D., (1979), "Computers and Intractability: A Guide to the Theory of NP-Completeness", W.H. Freeman, 1979.
12. Blum, C., and Roli, A., (2003) "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison", ACM Computing Surveys, Vol. 35, No. 3, pp. 268–308.

13. http://www.1000ventures.com/business_guide/lean_production_main.html , accessed on 10/01/2006.
14. Rajendran, C., and Ziegler, H. 1997, "An efficient heuristic for scheduling in a flowshop to minimize total weighted flow time of jobs", *European Journal of Operational Research*, 103, pp. 129-138.
15. http://www.managingchange.com/masscust/overview.htm, accessed on 10/01/2006.
16. Davis, S. M., (1987), "Future Perfect", Addison-Wesley publishing company.
17. Badurdeen, F.F., (2005), "Minicell Configuration for Mass Customization Manufacturing", PhD Dissertation, Ohio University, Athens, Ohio, USA.
18. Campbell, H.G., Dudek, R.A., and Smith, M.L., (1970), "A Heuristic Algorithm for the n Job m Machine Sequencing Problem", *Management Science*, Vol. 16 Issue 10, pp. 630-637.
19. Johnson, S.M., (1954), "Optimal Two and Three-stage Production Schedules with Setup Time Included", *Naval Research Logistics Quarterly*, 1, pp. 61-68.
20. Rajendran, C., and Ziegler, H., (1997), "An efficient heuristic for scheduling in a flowshop to minimize total weighted flow time of jobs", *European Journal of Operational Research*, 103, pp. 129-138.
21. Framinan, J.M and Leisten R, (2003), "An efficient heuristic for flow time minimization in permutation flow shops", *OMEGA*, 31, pp. 311-317.
22. http://personal.ashland.edu/~rjacobs/m503jit.html , accessed on 08/28/2006.
23. Teresko, J., (1994), "Mass Customization or Mass Confusion", *Industry Week,* 243 (12), pp. 45-48.
24. Drolet, J, Abdulnour, G and Rheault, M, (1996), "The cellular manufacturing evolution", *Computers & Industrial Engineering*, v 31, n 1-2, pp. 139-42.
25. Badurdeen, F.F., Class notes, ME 512: Manufacturing Systems, Fall 2005 semester, University of Kentucky, USA.
26. Kotha, S., (1996), "From mass production to mass customization: The case of the National Industrial Bicycle Company of Japan", *European Management Journal*, Vol.14, No.5, pp. 442-450.
27. Feitzinger, E., Lee, H. L., (1997), "Mass Customization at Hewlett-Packard: The Power of Postponement", *Harvard Business Review*, Jan/Feb97, Vol. 75, Issue 1, pp.116-121.

28. Kusiak, A., and Heragu, S.S., (1987), “Group Technology”, *Computers in Industry,* Volume 9, Issue 2, October 1987, pp. 83-91.
29. Greene, T.J., and Sadowski, R.P., (1983), "Cellular Manufacturing Control”, *Journal of Manufacturing Systems*, Vol. 2, No. 2, pp. 137-146.
30. MCLean, C.R., Bloom, H.M., and Hopp, T.H., (1982), "The Virtual Manufacturing Cell", *Proceedings of Fourth IFAC/IFIP Conference on Information Control Problems in Manufacturing Technology*, Gaithersburg, MD, October 1982.
31. Martin, R., Jocelyn, D., and Georges, A., (1995), “Physically Reconfigurable Virtual Cells: A Dynamic Model for a Highly Dynamic Environment”, *Computers and Industrial Engineering*, Vol. 29, Nos 1-4, pp. 221-225.
32. Vakharia, A.J., Moiley, J.P., and Huang, Y., (1999), “Evaluating Virtual Cells and Multistage Flow Shops: An Analytical Approach”, *The International Journal of Flexible Manufacturing Systems*, 11, pp. 291–314.
33. Suresh, N.C., and Slomp, J., (2005), “Performance comparison of virtual cellular manufacturing with functional and cellular layouts in DRC settings”, *International Journal of Production Research*, Vol. 43, No. 5, pp. 945–979.
34. Montreuil, B., and Lefrancios, P., (1996), “Organizing Factories as Responsibility Networks”, in Progress in Materials Handling Research 1996, ed. Graves, et al., Material Handling Institute, pp.375-411.
35. www.mel.nist.gov/msidlibrary/doc/flexms.pdf, Guixiu Qiao, Roberto Lu, and Charles McLean, “Flexible Manufacturing System for Mass Customization Manufacturing”, accessed 09/05/2006
36. http://www.people.hbs.edu/dupton/papers/organic/WorkingPaper.html#REF62249, David M. Upton, “A Flexible Structure For Computer-Controlled Manufacturing Systems”, accessed 09/05/2006.
37. Framinan, J.M., Usano, R.R, and Leisten R, (2005), “Comparison of heuristics for flow time minimization in permutation flow shops”, *Computers & Operations Research*, Volume 32, Number 5, pp. 1237–1254.
38. Pugazhendhi, S, Thiagarajan, S.; Rajendran, C.; and Anantharaman, N., (2004), “Relative performance evaluation of permutation and non-permutation schedules in flow line-based

manufacturing systems with flow time objective", *International Journal of Advanced Manufacturing Technology*, June 2004, Vol. 23 Issue 11/12, pp. 820-830.

39. Süer G.A., Vazquez R., and Santos J., "Evolutionary programming for minimizing the average flow time in the presence of non-zero ready times", *Computers & Industrial Engineering,* volume 45, pp. 331-344.
40. Allahverdi, A., Aldowaisan, T., (2002), "New heuristic to minimize total completion time in m-machine flow shops", *International Journal of Production Economics*, volume 77, pp. 71-83.
41. Cochran, J.K., Horng, S.M., and Fowler, J.W., (2003), "A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines", *Computers & Operations Research*, Volume 30, Issue 7, pp. 1087-1102.
42. Amin, A., Ammar, R., and Rajasekaran, S., (2005), "Maximizing Reliability While Scheduling Real-Time Task-Graphs on a Cluster of Computers", *10th IEEE Symposium on Computers and Communications (ISCC'05),* pp. 1001-1006.
43. Zegordi, S.H., Itoh, K., and Enkawa, T., 1995, "Minimizing makespan for flow-shop scheduling by combining simulated annealing with sequencing knowledge", *European Journal of Operational Research*, 85, pp. 515-531.
44. Framinan, J.M., Gupta, J.N.D., and Leisten, R., (2004), "A review and classification of heuristics for permutation flow-shop scheduling with makespan objective", *Journal of the Operational Research Society,* December 2004, Volume 55, Number 12, pp. 1243-1255 .
45. Nawaz, M., Enscore, E.E., and Ham, I., (1983), "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem", *OMEGA*, 11, pp. 91-95.
46. Gupta, J.N.D., Neppalli, V.R., and Werner, F., (2001), "Minimizing Total Flow Time in a Two-Machine Flowshop Problem with Minimum Makespan", *International Journal of Production Economics,* 69, pp. 323 - 338.
47. Kravchenko, S.A., Werner F., (2001), "A heuristic algorithm for minimizing mean flow time with unit setups", *Information Processing Letters*, Volume 79, Issue 6, pp. 291 – 296.
48. Azizoglu, M., Kirca, O., (1999), "On the minimization of total weighted flow time with identical and uniform parallel machines", *European Journal of Operational Research,* 113, pp. 91-100.

49. Woo, D.S., Yim, H.S., (1998) "A heuristic algorithm for mean flow time objective in flowshop scheduling." *Computers and Operations Research,* 25, pp. 175–82.
50. Liu, J., Reeves, C.R., (2001), "Constructive and composite heuristic solutions to the P//ΣCi scheduling problem", *European Journal of Operational Research,* 132, pp. 439–452.
51. Kirkpatrick, S., Gelatt, C.D., and Vecchi, M. P., (1983), "Optimization by simulated annealing", *Science*, 220 no 4598, pp. 671-680.
52. http://www.cs.sandia.gov/opt/survey/ts.html, accessed on 09/12/2006.
53. http://citeseer.ist.psu.edu/cache/papers/cs/25783/http:zSzzSzwww.docs.uu.sezSzasteczSzReportszSzReportszSz9712.pdf/altenbernd97slack.pdf, accessed on 09/12/2006.
54. http://www.dei.unipd.it/~fisch/ricop/tabu_search_glover_laguna.pdf#search=%22tabu%20search%20pdf%22, accessed on 09/12/2006.
55. Süer G.A., Badurdeen**,** F.F, and Dissanayake, N., (2006), "Capacitated Lot Sizing by Using Multi-Chromosome Crossover Strategy", *International Journal of General Systems*, forthcoming.
56. Gilmore, J.H. and Pine II, B.J., (1997), "Four Faces of Mass Customization", *Harvard Business Review,* Jan-Feb 1997, pp. 91-101.
57. Baldwin, C.Y., and Clark, K.B., (1997), "Managing in an age of modularity", *Harvard Business Review,* Sep/Oct97, Vol. 75, Issue 5, pp. 84–93.
58. http://www.infotech.com/MR/Industry%20Center/Manufacturing/Governance/Strip%20the%20Technology%20Fat%20from%20Lean%20Manufacturing.aspx, accessed on 10/01/2006.
59. Ruiz-Torres, A.J. , Enscore, E.E., and Barton, R.R., (1997), "Simulated annealing heuristics for the average flow-time and the number of tardy jobs bi-criteria identical parallel machine problem**",** *Computers & Industrial Engineering*, v 33, n 1-2, Oct. 1997, pp. 257-260.
60. Low, C., Yeh, J.Y., and Huang, Kai-I, (2004), "A robust simulated annealing heuristic for flow shop scheduling problems", *International Journal of Advanced Manufacturing Technology*, v 23, n 9-10, pp. 762-767.

61. Low, C., (2005), “Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines**”,** *Computers and Operations Research,* Volume 32 , Issue 8, pp. 2013 – 2025.
62. Yuedong, X., Yajiie, T., and Sannomiya, N., (2003), “Three-stage tabu search for solving large-scale flow shop scheduling problems**”,** *Transactions of the Institute of Electrical Engineers of Japan, Part C*, v 123-C, n 3, March 2003, pp. 601-608.
63. Grabowski, J., and Wodecki, M., (2004), “A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion”, *Computers & Operations Research*, v 31, n 11, Sept. 2004, pp. 1891-1909.
64. Chen, C.L., Usher, J.M., and Palanimuthu, N., (1998), “Tabu search based heuristic for a flexible flow line with minimum flow time criterion”, *International Journal of Industrial Engineering: Theory Applications and Practice*, v 5, n 2, Jun, 1998, pp. 157-168.
65. Shu, W.L., Wu, C.C., and Lai, W.C., (2004), “Design the Hardware of Genetic Algorithm for TSP and MSA”, Knowledge-Based Intelligent Information and Engineering Systems, Springer Berlin / Heidelberg, pp. 1260-1267.
66. Fogel, D.B., (1988), “An Evolutionary Approach to the Traveling Salesman Problems”, *Biological Cybernetics*, volume 60, pp. 139-144.
67. Oliver, I.M., Smith, D.J., and Holland, J.R.C., (1987), “A study of permutation crossover operators on the TSP”, *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference*, Lawrence Erlbaum, Hillsdale, New Jersey, pp. 224-230.
68. http://www.nd.com/genetic/crossover.html, accessed on 10/05/2006.
69. Neubauer, A., (1997), “The circular schema theorem for genetic algorithms and two-point crossover”, *Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications,* pp. 209-214.

VITA

The author was born on Nov 10, 1982 at Hyderabad, Andhra Pradesh, India. He received his bachelor of engineering degree in Mechanical engineering from Osmania University, Hyderabad, India in 2004. He held the position of teaching assistant in Department of Mechanical Engineering, University of Kentucky (August 2005- May 2006).

Phanindra Kumar Chadalavada