MINIMIZING TOTAL TARDINESS AND CREW SIZE IN LABOR INTENSIVE CELLS USING MATHEMATICAL MODELS

A thesis presented to

the faculty of

the Russ College of Engineering and Technology of Ohio University

In partial fulfillment

of the requirements for the degree

Master of Science

Kuldip U. Kamat

June 2007

This thesis titled

MINIMIZING TOTAL TARDINESS AND CREW SIZE IN LABOR INTENSIVE CELLS USING MATHEMATICAL MODELS

by

KULDIP U. KAMAT

has been approved for

the Department of Industrial and Manufacturing Systems Engineering

and the Russ College of Engineering and Technology by

GÜRSEL A. SÜER

Professor of Industrial and Manufacturing Systems Engineering

Dennis Irwin

Dean, College of Engineering and Technology

Abstract

KAMAT, KULDIP U., M.S., June 2007, Industrial and Systems Engineering

MINIMIZING TOTAL TARDINESS AND CREW SIZE IN LABOR INTENSIVE CELLS USING MATHEMATICAL MODELS (145 pp)

Director of Thesis: GÜRSEL A. SÜER

This thesis attempts to provide solution to some of the real world issues in cellular manufacturing, namely manpower allocation and cell loading. Although significant work has been done in these fields, research to minimize the Total Tardiness (TT) within a multi – cell environment has been limited. The methodology adopted in this thesis is a two-step one. Firstly, manpower allocation for operations within a cell is determined using mathematical models. Traditional mathematical models for finding the number of operators to be assigned for an operation have been expanded to be able to determine the individual operators to be assigned for each operation. This step also includes variations of the mathematical model for sharing of operators between operations and sharing of operators with restrictions. The models with little or no restrictions have found to yield a higher production rate than the model with no operator sharing allowed. The same model has been extended to introduce operator skill levels as well.

The next step involves developing and testing mathematical models for cell loading. The performance measure examined in this phase is the Total Tardiness subject to total Crew Size restriction. The Total Tardiness is reduced by increasing the Crew Size. In addition, the crisp mathematical models have been adapted to solve a bi-objective problem for minimizing the Total Tardiness and the Crew Size using fuzzy sets. The results indicate that the fuzzy math models offer a good alternative in terms of accuracy to the optimal solution, as well as the computation time.

Approved: ____

GÜRSEL A. SÜER

Professor of Industrial and Manufacturing Systems Engineering

Acknowledgements

A number of people have touched my life so far – and I would like to dedicate this work to them. My parents for believing in me, my friends for their guidance and helping me stay positive.

I would like to thank Dr. Süer for his constant guidance throughout my Master's coursework and this thesis. His insights into manufacturing systems have helped me appreciate the complexities involved in real-world systems, which has shaped the way I think through a problem.

I would also like to thank the entire Industrial & Systems Engineering faculty, in particular Dr. Sormaz and Dr. Weckman; and also my peers at Ohio University for giving me so many opportunities to grow academically, and otherwise. I would also like to thank Dr. Adie from the Economics department at Ohio University for his time and valuable inputs to this work. I would like to mention the assistance my peers Emre and Jing provided me with programming the mathematical models.

Of course, as with everything in life – nothing significant can be accomplished without guidance from an inexplicable, yet a very special force called faith. It is what has kept me motivated and focused through this work.

Table of Contents

Page

List of Tables

List of Figures

CHAPTER I

INTRODUCTION

This chapter briefly explains the different manufacturing systems particularly Cellular Manufacturing (CM), issues and performance measures used in industrial settings. An overview of Production Rates, Manpower Allocation and Mathematical Modeling is also presented. An introduction to fuzzy mathematical modeling is also provided.

1.1 Manufacturing Systems

The term 'manufacturing' is used to describe either the fabrication or the assembly of a product from raw materials to finished goods. Several inputs such as consumer demand, machine, Crew Size, energy, investments and human resources go into the production of a good. In order for a firm to be profitable and stay competitive, the cost of these inputs should remain as low as possible.

Depending on the quantity and the complexity of these inputs, and more importantly on the type of product manufactured; a manufacturing system can be typically designed in four ways. The manufacturing layout could be a fixed layout, a product layout, a process layout or a cellular layout. Figure 1.1 demonstrates the differences between a product, process and a cellular layout.
Fixed Layouts are used to manufacture products which are very large and typically have different processes running concurrently on the product. The manufacturing of ships and airplanes are good examples of manufacturing that utilize fixed layouts.

Product layouts are designed for the manufacture of single products which require different processes. A typical product layout utilizes a layout where the stations are arranged in the sequence of operations involved in the production. Product layouts are characterized by high volume and unidirectional flow of products. Figure 1.1.a represents such a type of a layout. Production of steel and plastics are examples of products that utilize a product layout.

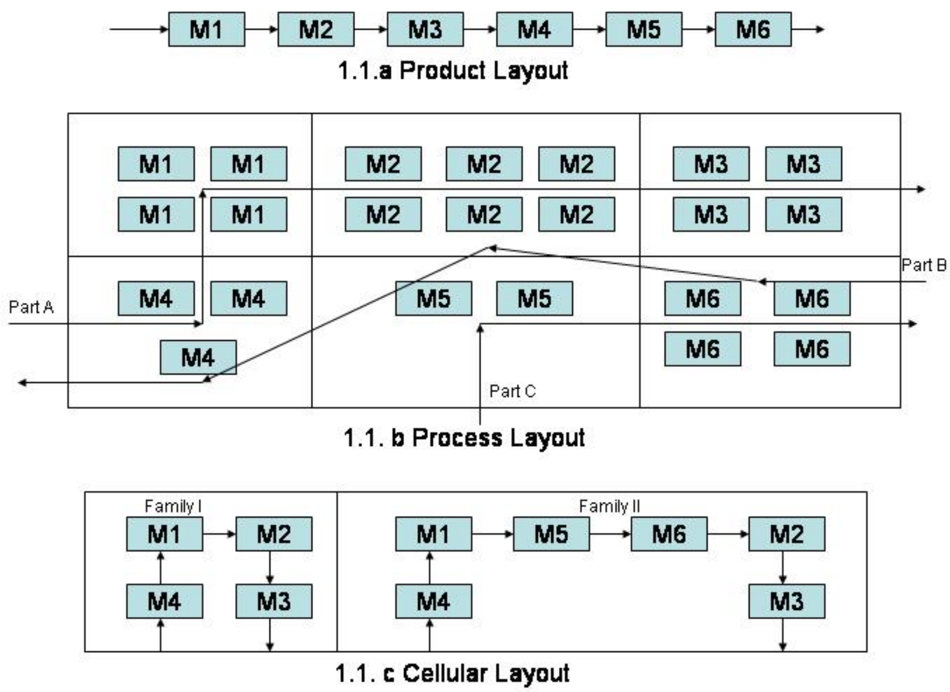

Figure 1.1: Flow of materials utilizing Product, Process & Cellular Layouts

Process layouts, on the other hand departmentalize the machines capable of performing similar functions as shown in the Figure 1.1.b. For example, part A passes through machines 4, 1, 2 and 3; part B uses machines 6, 2 and 4 and part 3 uses machines 5 and 6. Examples of products utilizing a process layout are automotive components such as gears which pass through various processes organized in different departments for milling, cutting, finishing etc. Process layouts are characterized by higher flexibility in terms of number of products, lower production volumes and high machine utilization.

Cellular layouts offer increased flexibility in terms of number of products by grouping similar parts (based on the processing requirements) in sufficient quantities. These parts are then processed on their own group of machines (cells) as depicted in Figure 1.1.c

Table 1.1 below shows the strengths & weaknesses of the layout types discussed above [1]

Table 1.1: Characteristics of Layout Types

Characteristics	**Product Layout**	**Process Layout**	**Cellular Layout**	**Fixed Layout**
Throughput time	Low	High	Low	Medium
Work in process	Low	High	Low	Medium
Skill level	Choice	High	Medium-High	Mixed
Product flexibility	Low	High	Medium-High	High
Demand flexibility	Medium	High	Medium	Medium
Machine utilization	High	Medium-Low	Medium-High	Medium
Worker utilization	High	High	High	Medium
Unit product cost	Low	High	Low	High

Depending on the production volumes and the product variety, products can be manufactured utilizing the product, process or cellular layouts as shown in Figure 1.2

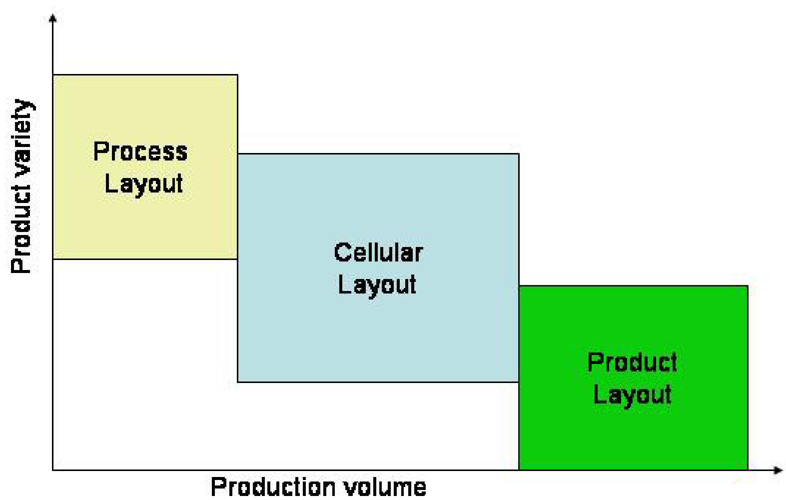

Figure 1.2 Comparison of different layouts

1.2 Cellular Manufacturing

Cellular Manufacturing (CM) offers a good balance between flexibility in terms of number of products and machine utilization. Depending on the processing complexity, manufacturing cells can either be machine-intensive or labor-intensive. Typically, more complex processes utilize machine-intensive cells and in such environments, the importance of machine allocation to cells is higher than the labor allocation. On the other hand, in labor-intensive cells, the machines are smaller and inexpensive; and require higher Crew Size (CS) levels. In labor-intensive environments, the availability of machines is assumed not to be a problem. Goods such as apparel manufacturing and jewelry manufacturing typically utilize labor intensive cells. Such cells utilize variable manpower levels in each cell based on the demand and the products assigned to be manufactured in each cell.

1.3 Cellular Control

Generally, Cellular Layouts require two steps to ensure a smooth production – these are Cell Loading and Manpower Allocation. Cell Loading is the process of allocating parts to cells as shown in Figure 1.3. In this figure, we see that jobs 1, 3, 4 and 5 are assigned to Cell I, jobs 2 and 6 assigned to Cell II, and jobs 7 and 8 assigned to Cell III. Manpower Allocation is determining the number of operators to be assigned to each operation in a cell for each product as shown in Figure 1.4.a. In this figure, we see that Cell I has six operators, Cell II has ten operators, and Cell III has eight operators. The total number of operators in the manufacturing unit is referred to as the Crew Size. The second step in manpower allocation is finding the operators required to perform each operation within a cell as shown in Figure 1.4.b. In this Figure 1.3, we see that two operators are assigned to operation I, one operator to operation II and three operators to operation III.

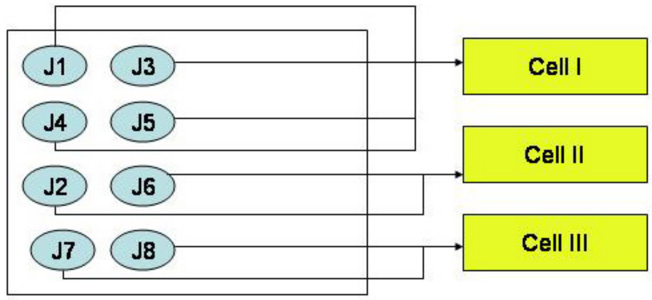

Figure 1.3: Cell Loading

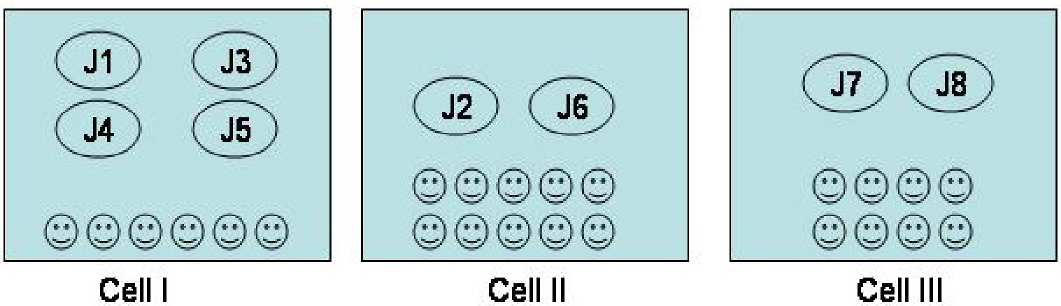

Figure 1.4.a: Manpower allocation to cells

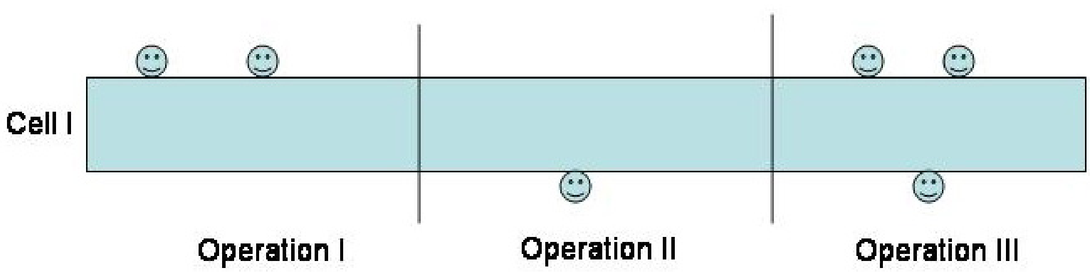

Figure 1.4.b: Operator allocation to operations

1.4 Cell Scheduling

Manufacturing Scheduling can be broadly defined as the allocation of jobs to machines in the cell with the objective of maximizing or minimizing (or in some cases a combination of the both) a performance measure under the given set of constraints; as well as determining the sequence of products to be manufactured within a cell. These performance measures can be total completion time, makespan, costs associated with manufacturing or tardiness.

1.5 Performance Measures

In this thesis, the performance measures considered are Total Tardiness and Crew Size. A job is considered tardy if it is completed beyond the due date ($c_i > d_i$); where c_i is the completion time of the job and d_i is the due date for the job. Tardiness for a job can take a value of zero or a positive value only, i.e. Ti = max $\{0, c_i > d_i\}$. The Total Tardiness (TT) is defined as the summation of tardiness for all jobs in the system. For example, if the completion times for three tardy jobs are 10, 15, and 17; and their respective due dates are 8, 15 and 16; then the tardiness values for these jobs are 2, 0 and 1 and the TT value is 3.

Crew Size Levels for each cell can be broadly defined as the number of operators assigned to the cell for the production of the parts or goods that have been assigned to the cell. Crew Size Levels fluctuate based on the type of processes, as well as the demand from one period to the next. In a manufacturing system, it is important to have an optimal value for Manpower Level as increased ML mean an increased labor cost to the unit (and hence for the product) whereas a very low ML would mean a larger number of tardy jobs as more operations would compete for limited resources.

1.6 Mathematical Modeling

The behavior of a real world system can be replicated using a set of mathematical equations and variables. These variables can take integer values or non-integer values depending on the constraints imposed.

Conceptually, all mathematical models have these variables defined as integer, real, binary or string values. The objectives of the model could be to minimize costs involved in production, minimize the number of tardy jobs, minimize makespan, maximize production rates, etc. Mathematical models are used to arrive at the optimal solution for this objective function for a given set of constraints. Real world constraints to a system could include a restriction on Crew Size levels for the system or a part of the system, limitation on resources like machines or capital, limitation on the number of different products that can be manufactured in a cell, etc.

Mathematical models can have a single objective function or more than one objective function depending on the user's goals. In this thesis, ILOG OPL Studio 3.7.1 is used to build and run these mathematical models. This software offers flexibility in terms of the number of variables and constraints that can be represented as well as quicker computation times compared to conventional solver programs like Excel.

1.7 Fuzzy Mathematical Modeling

Fuzzy modeling adopts a bi objective fuzzy math model to optimize more than one performance measure. In this work, the performance measures are Total Tardiness and Crew Size which are considered together in the fuzzy model.

The Figure 1.5.a shows the membership function for Total Tardiness which shows the Satisfaction Level against the TT. This suggests a linearly decreasing function of satisfaction level for TT values between 15 and 45. Similarly, Figure 1.5.b shows a membership function for CS where the satisfaction level is constant for all CS values lower than 40, but a linearly decreasing function for CS values between 40 and 50.

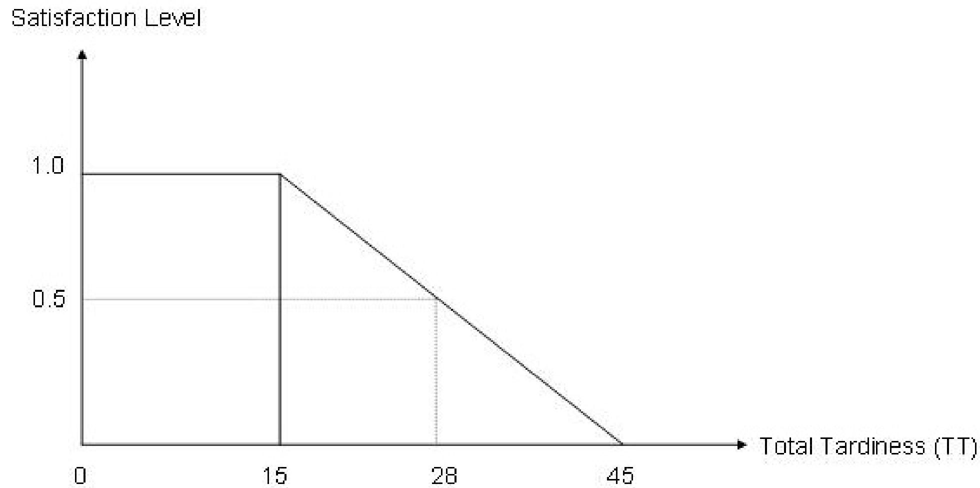

Figure 1.5.a: Membership function for Total Tardiness (TT)

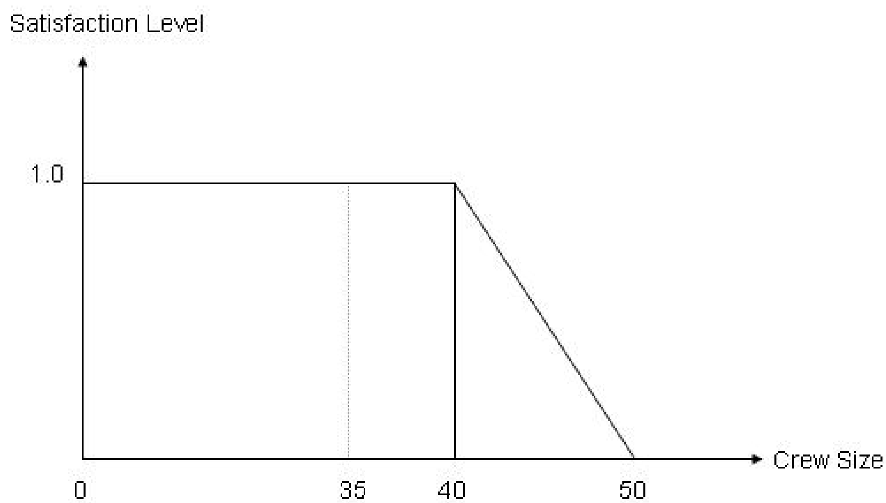

Figure 1.5.b: Membership function for Crew Size (CS)

1.8 Objectives of Thesis

This thesis utilizes concepts of optimal manpower allocation and cell loading introduced by Süer [2] to study the impact of correct manpower allocation under different scenarios on the production rates within different cells. Additionally cell loading with the objective of minimizing the Total Tardiness and the Crew Size is studied.

As a broad classification, this thesis uses two objectives – first to develop models to generate alternate cell configurations with or without operator sharing in labor intensive cells. Both these scenarios for sharing and non-sharing are examined.

The second objective function is to load the multiple cells to minimize the Total Tardiness (TT) with an objective to determine the crew sizes. For this, mathematical models were adapted minimize the Total Tardiness for a given crew size. In terms of usage, 'Manpower Level' is used in association with finding the number of operators within a cell; and 'Crew Size' is used in association with the total number of operators in the manufacturing unit. The goal is to find the sequence of jobs across multiple cells for a weekly schedule.

Lastly, fuzzy math models were adapted to the earlier cell loading models to optimize two performance measures, the TT and CS.

1.9 Justification

To the best knowledge of this author, this problem has not been studied before. Although research has been done on manpower allocation and cell loading in labor-intensive cells, limited research has been done focusing on the individual operator assignment (finding the operator to operation match as opposed to finding the total number of operators required for an operation); as well as on addressing the performance measures of Total

Tardiness and Crew Size together. These scenarios have been studied for different operator sharing rules between operations.

Real world systems rarely depend on the measurement of one performance measure as a metric of overall system performance. Generally, more than one performance measure is used to benchmark the system performance. For this, fuzzy math models were developed to study more than one performance measure.

1.10 Organization of the Thesis

- Chapter I gives an overview of the different types of manufacturing systems, and briefly explains some of the concepts of Cellular Manufacturing, mathematical modeling and the performance measures used and the objective of the thesis.

- Chapter II summarizes some of the previous work done in Cellular manufacturing systems, Cell Loading, Crew Size Levels and Fuzzy Aspiration levels.

- Chapter III discusses the problem in more detail by addressing the issues in manufacturing systems, and the mathematical models used to depict these systems for manpower allocation and fuzzy aspiration levels.

- Chapter IV provides more detail on the methodology which is a two step approach to solving the problem – Manpower Allocation models and the Cell Loading Models.

- Chapter V provides an analysis of the results

- Chapter VI provides concluding remarks and scope in terms of future work

CHAPTER II
LITERATURE REVIEW

Cellular Manufacturing has gained a lot of acceptance in several industries as it offers a good mix between higher flexibility and throughput. Several factors go in determining a good cellular layout such as demand which can be stochastic, product mix, operation times and machine capacities, setup and processing times for different products etc. [3]. This literature review mainly explores the research in the areas of manpower allocation, cell loading and fuzzy mathematical modeling.

The Total Tardiness problem has several real world applications in manufacturing, as well as other service industries. This problem is hard to solve as shown by Du and Leung [4]. Gupta and Chantaravaraparan [14] studied the group scheduling problem for minimizing the total tardiness on a single machine. The author proposes a mixed integer linear programming model for a single machine scheduling problem. Since the TT problem is NP hard, the authors also proposed heuristics, which were modifications of the NBR heuristic [15] and the PSK Heuristic [16]. Alidee and Rosa [18] suggested heuristic algorithms to solve this problem for a large number of jobs and machines. The objective is to minimize the Total Tardiness (TT) for a parallel machine problem. Both, weighted and unweighted tardiness are considered. The authors have utilized the Modified Due Date (MDD) algorithm proposed by Baker and Bertrand [5]. The algorithms proposed in this research are for the single machine total weighted tardiness (TWT) problem, single machine Total Tardiness (TT) problem, and for the parallel machine total weighted tardiness (TWT) problem. Another weighted approach for minimizing tardiness was studied by Kanet and Li [6]. A rule for Weighted Modified Due Date (WMDD) was studied given its practical relevance, and was compared against competing rules for weighted tardiness using simple queuing systems.

Another heuristic to solve a complex parallel machine scheduling problem for TT was studied by Bilge, Kirac, Kurtulan and Pekgun [7]. The problem is complex as the independent jobs have sequence dependent setups, non identical due dates and arrival times. The solution methodology proposed is Tabu Search (TS) which is an iterative search procedure used to find an optimal solution.

Cheing Hu studied the Total Tardiness problem with worker assignment for parallel machines [8] using heuristics for both, job scheduling and worker assignment. For job scheduling, the Shortest Processing Time (SPT), Earliest Due Date (EDD) and the Slack heuristics are used whereas the largest marginal contribution (LMC) procedure is used for worker assignment.

Süer and Bera [9] analyzed the simultaneous process of cell loading and cell size determination for labor intensive cells. This research proposed a two phase approach involving the generation of alternative cell sizes for different Crew Sizes, where optimal manpower allocation is performed. This is followed by developing various mathematical models for cell loading and cell size determination. The models determine both; the cell loading and the manpower size simultaneously. A scenario for loading six products into three cells using three alternative configurations was studied across multiple periods.

The Operator assignment problem has been studied by Wittrock [10] using the concept of operator "skill sets". These skill sets may be identical, non-identical or in some cases overlapping from one operator to another. The primary objective of this research was to maximize capacity of the system.

Süer, Saiz, Dagli and Gonzalez [11] evaluated several cell loading rules and their effects on performance measures in a labor intensive environment. The performance measures considered to evaluate the rules against each other were the number of tardy jobs (nT), maximum tardiness (T_{max}), the Total Tardiness (TT), Average Cell Utilization. Overall, eleven rules are grouped into six categories. A total of 48 combinations are created using these rules; and these rule combinations are compared against each other based on the performance measures. The conclusion of the study was that no single combination

performed well to all the performance measures. Also, when these performance measures were studied individually, it was seen that n_T, T_{max} and TT behaved similarly.
Du and Leung [9] studied the problem for Total Tardiness for a given set of jobs on a single machine. The set of jobs are independent and the objective function of the schedule is to minimize the Total Tardiness. The research proves that the problem is NP-hard. This is done by proving that the corresponding decision problem is NP complete.

A three phase hierarchical methodology for manpower allocation, cell loading and sequencing was proposed by Süer and Gomez [12]. The specified performance measures are optimized using linear programming and Evolutionary Programming [EP]. The methodology is split into three phases. The first phase is to generate alternate cell configurations where the production rates are maximized for a given manpower level. The second phase finds the optimal operator assignment and also deals with Cell Loading using an integer programming model. The third phase finds the sequence of jobs for the performance measure selected using Evolutionary Programming (EP). The proposed EP technique to be used is Genetic Algorithms (GA).

In a real world scenario, a balance between two or more performance measures is desirable when one performance measure is not sufficient to evaluate a cell loading or manpower allocation technique. Süer, Arikan and Babayigit [13] studied four different bi-objective mathematical models. These models considered setup times in a labor intensive cellular environment and focused on minimizing the total number of tardy jobs and minimizing the total manpower needed simultaneously. The work mentioned above was based on the fuzzy math models proposed by Werners [17]. This study is an offshoot of real world examples observed in medical devices and jewelry manufacturing where the numbers of operators often adjust to meet the varying demand. A balance between minimizing the number of tardy jobs (which is focused on meeting the customers' requirements) and minimizing the number of operators (which is focused on reducing costs for the manufacturing unit) is accomplished using fuzzy solution techniques. This study builds on models for common cell size, which implies equal workloads amongst cells; models for allowing lot-splitting between cells are studied as well. Here the partial completion of a job in a cell is considered acceptable. In the last model, partial completion within a cell is not allowed i.e. the job has to be entirely completed within a cell.

CHAPTER III
PROBLEM STATEMENT

This chapter provides an overview of the issues discussed in this thesis. Manpower level decisions are often complex in labor intensive units as there are a number of constraints such as labor costs, number of operators that can be assigned to an operation, etc.

Given these constraints, the primary objective for any labor intensive environment is then to maximize the production rates for each cell with constraints imposed on the number of operators within each cell. The allocation of operators to operations within a cell comprises of two steps – the total number of operators assigned to each operation, as well as the individuals assigned to work in each operation. This aspect of manpower allocation becomes especially important when decision making involves measuring operator performance.

The first phase involves assigning individual operators to operations within the cell. The Figure 3.1 illustrates an example of assignment of operators to individual operations.

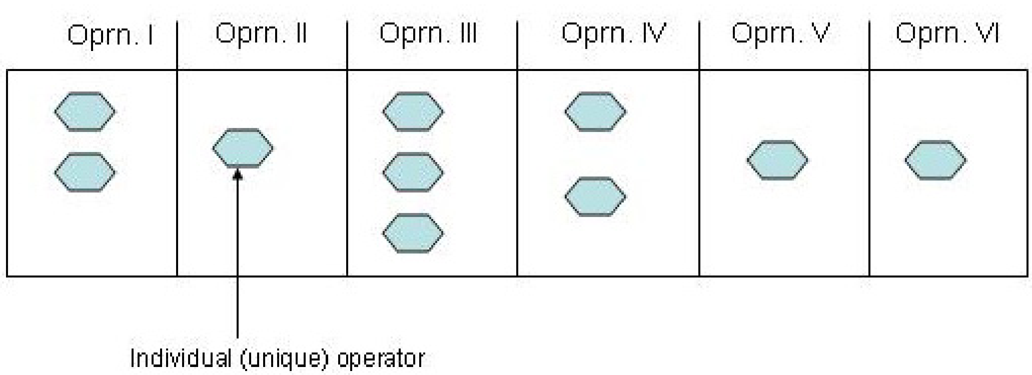

Figure 3.1: Example of Manpower Allocation across different operations within a cell

In the Figure 3.1 above, we see that Operation I has two Operators assigned, Operation II has one Operator assigned to it, etc.

Many manpower allocation issues across cells involve a changing crew size in each cell depending on the set of products being manufactured. The operator sharing is allowed between operations within a cell.

Many decisions need to be made in a Cellular Manufacturing environment such as the number of cells, Crew Size (CS), product allocation to cells, product sequence, number of operators to each operation, etc. The need for mathematical modeling in this thesis is based on the need to find the optimal solution in Cell Loading.

3.1 Operator Assignment

The first step in math modeling requires the assignment of individual operators to operations in each cell within the manufacturing unit. An assumption made in assigning operators to individual operations is that the operation times are known in advance.
The Operator assignment problem is studied for each cell configuration individually, and each operation within the cell is assigned a number of operators (>=1) depending on the operation times. The goal of this model is to maximize the Production Rate (PR) for a cell with a given number of total operators e.g. 10 operators assigned to a single cell across 5 operations with the operation times known in advance. This determination can be made for a variable manpower level e.g. 11 operators, 12 operators, 15 operators within a cell etc. for the given number of operations and the operation times.

Labor intensive cells could have complexities associated with the number of operators allowed within a cell due to space restrictions. In such cases, in addition to determining the number of operators assigned to each operation in a cell and determining the number of operations for each operator becomes beneficial.

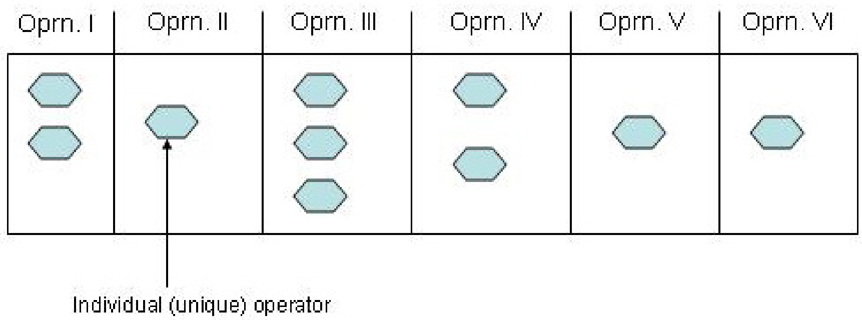

Figure 3.2: Operator assignment with no sharing allowed between operations

In the Figure 3.2, the operators are distributed between individual operations with no sharing allowed between operations. Such a scenario restricts any inter-operation movement of operators as they are assigned to perform only a particular operation. In the example, there are ten operators in the cell across six operations.

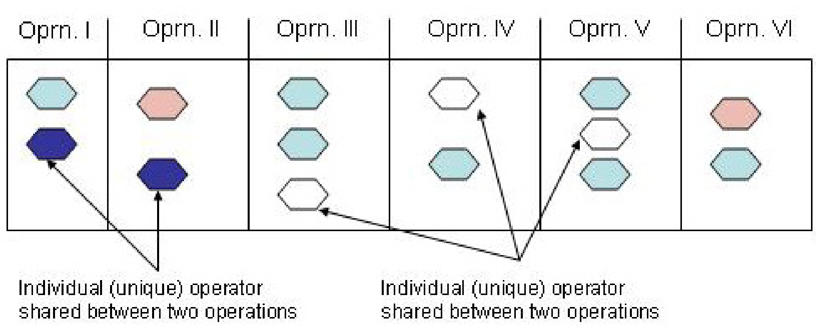

Figure 3.3: Operator assignment with sharing allowed between operations

Figure 3.3 shows the scenario where an operator is allowed to perform more than one operation within a cell. The example above shows three out of the ten operators performing more than one operation. Two of these three shared operators perform two operations each, while the third performs three operations within the cell.

By relaxing the conditions of the number of operations performed by a single operator, the Production Rate (PR) can show an improvement, as there is greater flexibility in assigning resources to an operation when it is needed e.g. operations with the largest processing time might require more manpower for a short period of time (for example when a part which has a higher processing time is being manufactured). The disadvantages of this scenario are that it could lead to too much inter-operation movement for an operator, and a worker could be assigned partially, for e.g. less than 7% (refer table 5.3) of the operator's available time which could introduce inefficiencies. Additionally, the constant movement of operators between operations would mean that the learning curve (ability of an operator to perform the same task more effectively with increased experience) for an operator would be diminished.

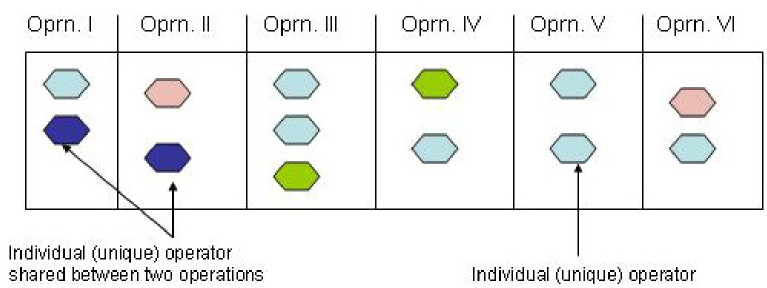

Figure 3.4: Operator sharing allowed with restrictions

Figure 3.4 shows the sharing of operators between operations with restrictions in terms of number of operations an individual operator can perform within a cell. This attempts to overcome some of the drawbacks of unrestricted sharing between operations. In the example illustrated, each operator is allowed to perform a maximum of only two operations within a cell. Such an operator for example, could spend about 70% of his/her time performing one operation, and the remainder 30% on another operation within the same cell.

The above mentioned examples (Figure 3.2, Figure 3.3 and Figure 3.4) can be applied to a number of Crew Sizes. The examples illustrate 10 operators within a cell; however this can be increased to a higher number or reduced to a lower number depending on the requirements.

Real world systems utilize this concept of operator sharing to increase Production Rates by assigning more operators to an operation when a product with several operations has one or two operations which has a high processing time.

3.2 Cell Loading

The next step (after the manpower allocation to individual cells) involves Cell Loading i.e. assigning individual products to cells in an effort to optimize a performance measure. This involves assigning the products to different cells and sequencing these products.

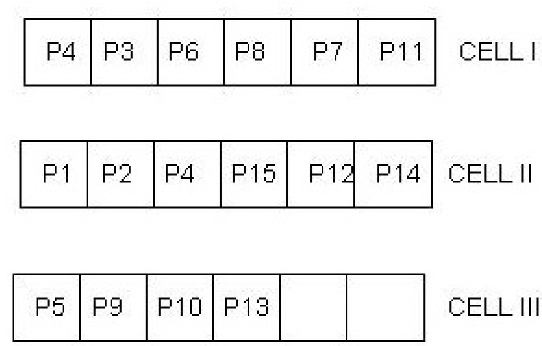

Figure 3.5: Loading the different products into multiple cells

The Figure 3.5 above shows the product assignment to different cells. This step takes into account all the products that need to be loaded across multiple cells. Sequencing is the process of ordering the products to be manufactured within a cell in the order they are to be produced. An assumption made in this thesis is that all the raw materials needed to manufacture the products are available when needed. The demand could be considered as a ‘build to order’ i.e. the weekly schedule is made for the products that are ordered by the customers weekly. The demand can also be considered as a net demand for the week i.e. some inventories may exist, and will be considered in computing the net demand.

For a manufacturing unit with multiple products, the customer demand drives the production, and thereby the performance measures used to track production. Several performance measures such as the throughput rate, total tardiness etc. are measured. For a unit with several products, the measure of the total number tardy jobs is important as customer satisfaction is tied closely to this metric. A job is considered tardy if the completion time is greater than its Due Date ($Ci > Di$). For example, if a job had a due date of 12 units, and it is completed at t = 15, then the tardiness for that job is 3 units. In a practical application, a product within a cell can have several jobs. If each of these jobs is tardy by a certain amount, then the Total Tardiness (TT) is increased. TT can be defined as the sum of tardiness of all jobs; so the TT for the system is the sum of the tardiness for all products in the system.

In a perfect sense, customers for any product would like to have no Total Tardiness (TT = 0). However, firms are often challenged by constraints such as the availability of resources like manpower, as well as space and budgetary restrictions. In such a scenario, cell loading would determine the value of tardiness of each job, and hence the Total Tardiness in the system. In a real world case, customers might be willing to purchase if the jobs are tardy by a small amount i.e. each product is tardy by a small amount rather than a few jobs not tardy, but the remainder tardy by a large amount.

3.3 Fuzzy Mathematical Modeling

Manufacturing firms face a challenge of meeting customer satisfaction levels which could be based on the Total Tardiness measure. The total manpower availability for manufacturing, often defined as the Crew Size is a constraint for production. Hence, a balance needs to be made between the two to achieve a maximum value of customer satisfaction while keeping the Crew Sizes low. Such a multi-criteria objective can be studied using fuzzy mathematical models.

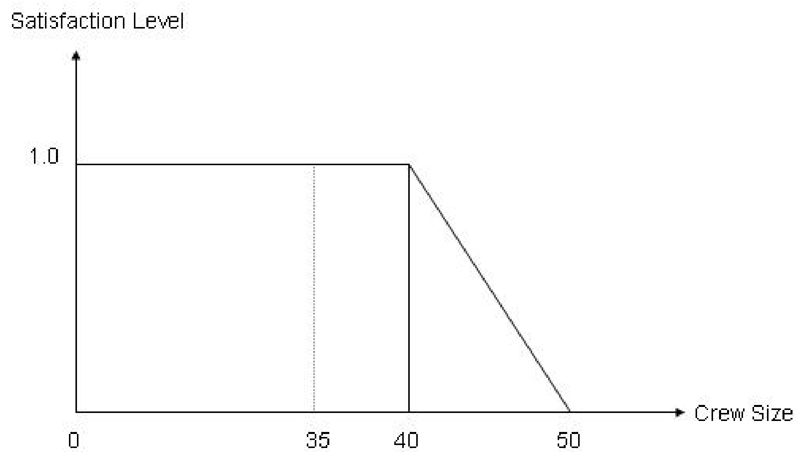

Figure 3.6: Function for the Satisfaction Level based on the Crew Size

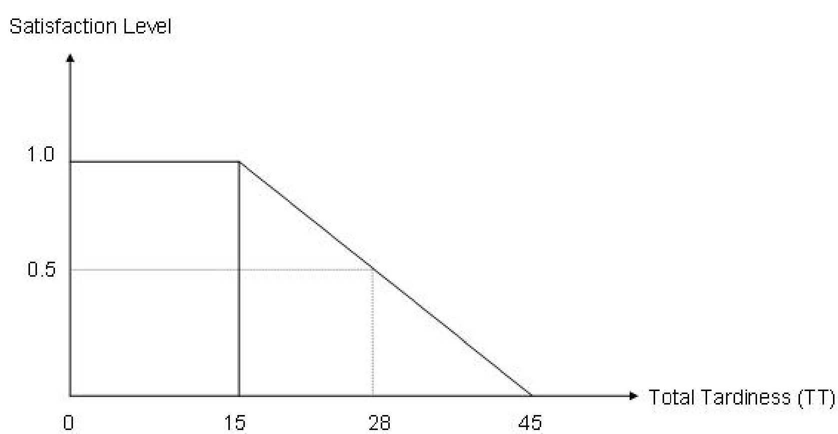

Figure 3.7: Function for the Satisfaction Level based on Total Tardiness

Figure 3.6 and Figure 3.7 show the satisfaction levels based on Crew Size and Total Tardiness respectively. Figure 3.6 shows that it is possible to achieve the maximum satisfaction level of 1.0 for a unit which has a Crew Size of 40 or less. A lower Crew Size if desirable to the firm as it means lower operating costs. A Crew Size of 50 is least desirable for the firm, and a number between 40 and 50 implies a decreasing linear function for satisfaction levels.

Figure 3.7 shows the satisfaction level for a set of customers with regards to the Total Tardiness (TT). A set of customer might be willing to purchase a products with a TT value of 15 or lower, but might be very dissatisfied (satisfaction level = 0) for a TT value of 45 or more. A TT value between 15 and 45 will have a satisfaction level between 0.0 and 1.0.

In the specific instances used in Figure 3.6 and Figure 3.7, the maximum satisfaction level is 0.5. This is a minimum of the satisfaction levels for the values corresponding to CS and TT. Assume that for the run for the fuzzy mathematical model, the CS = 35 and the TT = 28, and solutions provided by the model for satisfaction levels are 1.0 and 0.5 respectively; the overall satisfaction level is 0.5.

Fuzzy sets are used to strike a balance between two or more performance measures to maximize a satisfaction level for both the producer as well as the consumer.

CHAPTER IV
METHODOLOGY

In this chapter, the proposed methodology for finding the optimal Crew Size, cell loading and fuzzy math models for finding a balance between the crew size and the total tardiness are explained in detail.

The general methodology for the solution can be split into 3 phases as shown the Figure 4.1.1. The specific methodology for manpower allocation and cell loading in this thesis can be found in figure 4.1.2

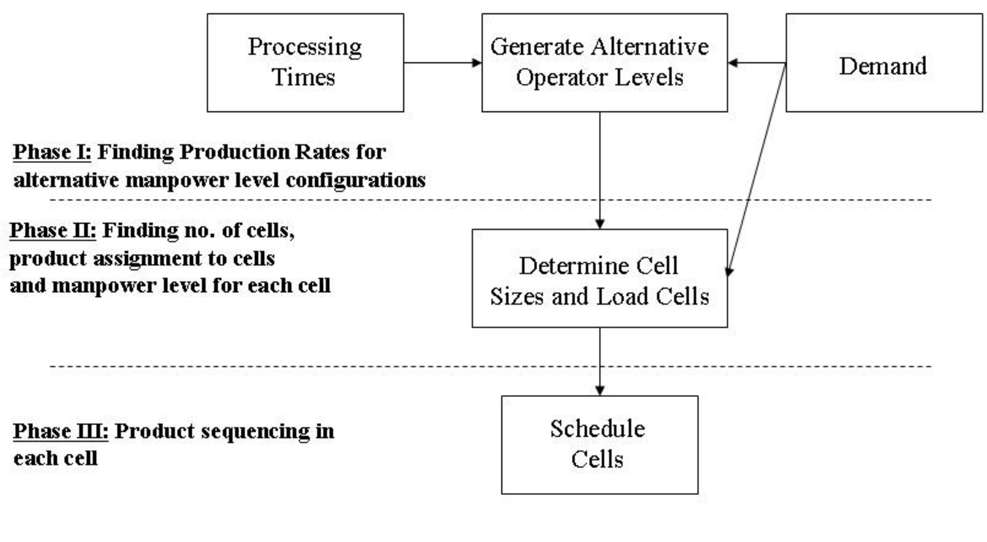

Figure 4.1.1: The general methodology in cell loading

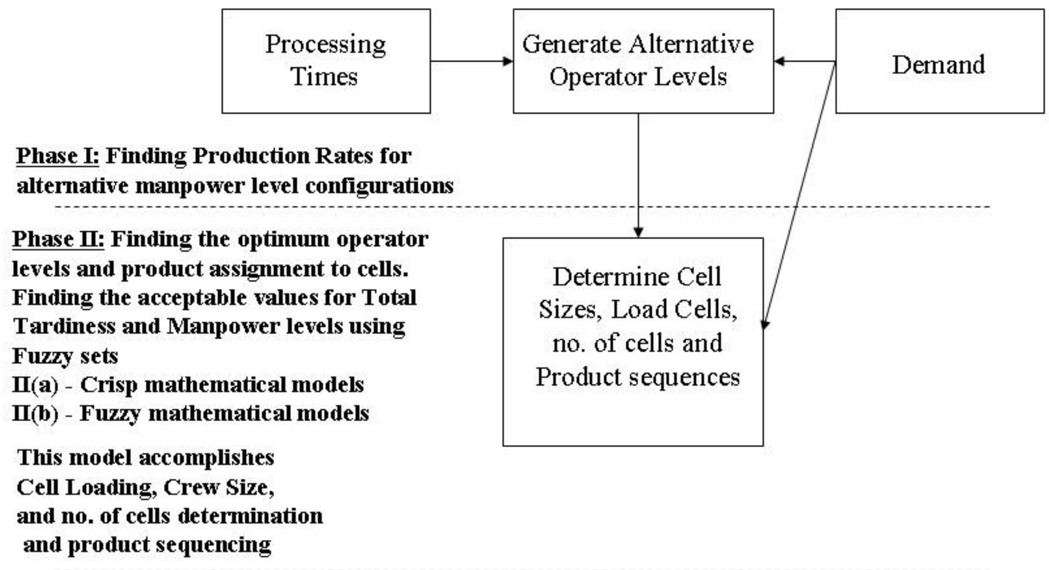

Figure 4.1.2: The methodology for cell loading used in this thesis

In this thesis, Phase II and Phase III are accomplished by the same math model simultaneously i.e. the Phase II model loads the products to the cell as well as sequences them within each cell.

Table 4.1: Tasks and objectives for each phase

Phases	Tasks	Objectives
Phase I	Finding Production Rates for alternative manpower level configurations	Find the individual assignments and the total number of operators required for each operation in the cell. Also determine the maximum Production Rates in each scenario for different manpower level configurations
Phase II (A) & Phase II (B) and Phase III	Finding the optimum operator Levels and product assignment to cells. Finding the acceptable values for Total Tardiness and Manpower levels using	Product sequencing in each cell for a weekly schedule

	Fuzzy sets II(a) - Crisp mathematical models II(b) - Fuzzy mathematical models	

4.1 Phase I: Finding the Optimal Manpower Levels

This objective of this phase is to find the optimal manpower assignments for every operation of a product for a given manpower level. This is done using a mixed integer programming model. The inputs to this model are the processing times for each operation, and the total number of operators for all operations i.e. manpower level. The outcome of this model is the Production Rate (which is maximized for the given inputs) and the individual operator assignment for each operation.

In the model formulated below, the Equation (4.1) is the objective function for the production rate for each product which is maximized. Equation (4.2) provides the constraint for achieving the maximum production rate for each product i.e. determining which operator should be assigned to each operation to achieve the production rate R. These constraints are the processing times for each operation and the operator assignment. Equation (4.3) provides the constraint that each operator is assigned once to at least one operation within the cell.

In section 4.1.1, y_{kj} takes on integer values which mean that the operator assignment to ever operation is an integer. In a practical sense this implies that an operator is assigned to only operation. In 4.1.2, y_{kj} takes on real values which means that the the operators can be assigned to more than one operation.

In 4.1.3, two additional constraints are introduced. Equation (4.4) dictates that one operator can perform a maximum of two operations; whereas (4.5) is used to link the real values of w_{kj} to the real values of y_{kj}.

4.1.1.: Mathematical Model for operator assignment (No sharing allowed):

Objective Function:

Max Z=R (4.1)

Subject to:

$$\sum_{k=1}^{n}[(a)*(y_{kj}/t_j)]-R \geq 0 \qquad j = 1, 2, 3 \ldots s \qquad (4.2)$$

$$\sum_{j=1}^{s} y_{kj} \leq 1 \qquad k = 1, 2, 3 \ldots n \qquad (4.3)$$

y_{kj} >= 0 and integer for all k and j

R >= 0 and real

Notation:

R	production rate
y_{kj}	= 1, if operator k is assigned to operation j
	= 0, otherwise
t_j	processing time of operation j
s	number of operations in the cell
k	operator index
j	operation index
n	number of operators
a	minutes per unit time.

In this thesis production rate was computed on an hourly basis, therefore (a) was taken as 60 minutes.

This model is run for each manpower level and product independently.

4.1.2.: Mathematical Model for operator assignment (Sharing allowed without restrictions):

Objective Function:

Max Z=R (4.1)

Subject to:

$$\sum_{k=1}^{n}[(a)*(y_{kj}/t_j)]-R \geq 0 \qquad j = 1, 2, 3\ldots s \qquad (4.2)$$

$$\sum_{j=1}^{s} y_{kj} \leq 1 \qquad k = 1, 2, 3\ldots n \qquad (4.3)$$

y_{kj}>= 0 and real for all k and j
R>= 0 and real

Notation:

R	production rate
y_{kj}	= non zero, if operator *k* is assigned to operation *j*
	= 0, otherwise
t_j	processing time of operation *j*
s	number of operations in the cell
k	operator index
j	operation index
n	number of operators
a	minutes per unit time.

In this thesis production rate was computed on an hourly basis, therefore (a) was taken as 60 minutes.

This model is run for each manpower level and product independently.

4.1.3.: Mathematical Model for operator assignment (Sharing allowed with restrictions on the number of operations a single operator is allowed to perform):

Objective Function:

Max Z=R (4.1)

Subject to:

$$\sum_{k=1}^{n}[(a)*(y_{kj}/t_j)]-R\geq 0 \qquad j = 1, 2, 3\ldots s \qquad (4.2)$$

$$\sum_{j=1}^{s} y_{kj} \leq 1 \qquad k = 1, 2, 3\ldots n \qquad (4.3)$$

$$\sum_{j=1}^{s} w_{kj} \leq u \qquad k = 1, 2, 3\ldots n \qquad (4.4.)$$

$$w_{kj} - y_{kj} \geq 0 \qquad k = 1, 2, 3\ldots n \qquad (4.5)$$

j = 1, 2, 3…s

y_{kj} >= 0 and real for all k and j (4.6)

w_{kj}>= 0 and binary

R>= 0 and positive

Notation:

R production rate

y_{kj} = non zero, if operator k is assigned to operation j

= 0, otherwise

w_{kj} = 1, if $y_{kj} > 0$

= 0, otherwise

t_j processing time of operation j

s number of operations in the cell

k	operator index
j	operation index
n	number of operators
u	maximum number of operations an operator is allowed to perform
a	minutes per unit time.

In this thesis production rate was computed on an hourly basis, therefore (a) was taken as 60 minutes.

This model is run for each manpower level and product independently

```
// Defining Z as the value of R where R is the production rate
var float+z in 0..maxint;
var float+R in 0..maxint;

maximize z //objective function to find the maximum production rate

subject to

{

z = R;
(60/0.45)* (y11+ y21+ y31+ y41+ y51+ y61+ y71+ y81+ y91+ y101)- R >=0;   // Finding maxi
(60/0.38)* (y12+ y22+ y32+ y42+ y52+ y62+ y72+ y82+ y92+ y102)- R >=0;
(60/0.33)* (y13+ y23+ y33+ y43+ y53+ y63+ y73+ y83+ y93+ y103)- R >=0;
(60/0.82)* (y14+ y24+ y34+ y44+ y54+ y64+ y74+ y84+ y94+ y104)- R >=0;
(60/0.39)* (y15+ y25+ y35+ y45+ y55+ y65+ y75+ y85+ y95+ y105)- R >=0;
(60/0.71)* (y16+ y26+ y36+ y46+ y56+ y66+ y76+ y86+ y96+ y106)- R >=0;


y11+ y12+ y13+ y14+ y15+ y16= 1; // Operator can be assigned to only one Operation
y21+ y22+ y23+ y24+ y25+ y26 = 1; // Remember as "Operator to Operation"
y31+ y32+ y33+ y34+ y35+ y36 = 1;
y41+ y42+ y43+ y44+ y45+ y46 = 1;
y51+ y52+ y53+ y54+ y55+ y56 = 1;
y61+ y62+ y63+ y64+ y65+ y66 = 1;
y71+ y72+ y73+ y74+ y75+ y76 = 1;
y81+ y82+ y83+ y84+ y85+ y86 = 1;
y91+ y92+ y93+ y94+ y95+ y96 = 1;
y101+ y102+ y103+ y104+ y105+ y106 = 1;
```

Figure 4.2: Mathematical model example for OPL

4.2 Phase II (A): Cell Loading to minimize Total Tardiness (TT)

Integer programming is used to build the model for cell loading. The performance measure used to evaluate the performance is Total Tardiness (TT) for the given total manpower level i.e. the total number of operators available in the manufacturing unit.

Equation (4.7) is the objective function for minimizing the Total Tardiness (TT) for all the products in the system. Equations (4.8) states that each job must be assigned to one position i.e. product assignment constraint, and Equation (4.9) states that each position should have only one job i.e. sequence assignment constraint.

Equation (4.10) is the completion time for the first job in the sequence, whereas equation (4.11) is the completion time for the subsequent jobs in the sequences. Similarly Equation (4.12) and (4.13) are the tardiness values for the first job and the subsequent jobs in the sequence. Equation (4.14) dictates that each cell must have only one configuration level.

<u>Mathematical Model</u>:

Objective Function:

$$MinTT = \sum_{j=1}^{s}\sum_{k=1}^{q}\sum_{m=1}^{p} T_{jkm} \qquad (4.7)$$

Subject to:

$$\sum_{j=1}^{s}\sum_{k=1}^{q}\sum_{m=1}^{p} X_{ijkm} = 1 \qquad i = 1, 2, 3\ldots n \qquad (4.8)$$

$$\sum_{i=1}^{n} X_{ijkm} \leq y_{km} \qquad j = 1, 2, 3\ldots s \qquad (4.9)$$

k = 1, 2, 3...q
m = 1, 2, 3...p

$$C_{1km} = \sum_{i=1}^{n} [P_{im} * X_{i1km}] \qquad k = 1, 2, 3\ldots q \qquad (4.10)$$

m = 1, 2, 3...p

$$C_{jkm} = C_{j-1,km} + \sum_{i=1}^{n} P_{im} * X_{ijkm} \qquad j = 2, 3, 4...s \qquad (4.11)$$

k = 1, 2, 3...q

m =1, 2, 3...p

$$C_{jkm} - [\sum_{i=1}^{n} d_i * X_{ijkm}] \le T_{jkm} \qquad j = 1, 2, 3...s \qquad (4.12)$$

k = 1, 2, 3...q

m = 1, 2, 3...p

$$\sum_{m=1}^{p} y_{km} \le 1 \qquad k = 1, 2, 3…q \qquad (4.13)$$

$$\sum_{m=1}^{p} \sum_{k=1}^{q} b_{km} * y_{km} \le W \qquad (4.14)$$

where X_{ijkm} *(0, 1)*

and C_{jkm}, $T_{jkm} \geq 0$

Notation:

T_{jkm} Tardiness value for the job in j^{th} sequence in cell k with configuration m)

X_{ijkm} assignment of the job i to the j^{th} sequence in cell k with configuration m

C_{jkm} completion time of the job in the j^{th} sequence in cell k with configuration m

P_i processing time for job i

d_i due date for job i

q number of cells

b_{km} manpower level for configuration m in cell k

p	number of manpower level configurations
s	number of operations
W	Total Crew Size

The OPL code in the appendix shows the cell loading model for 6 products in 3 cells for 2 manpower levels.

4.3 Phase II (B): Finding acceptable levels of Total Tardiness and Crew Size based on fuzzy intervals

The next step is to determine the acceptable levels of Total Tardiness and Crew Size for a given range. This is done by assigning an acceptable range for both; the Crew and the Total Tardiness (TT).

Figure 4.3 and Figure 4.4 give a visual indicator for these two values for ML and TT.

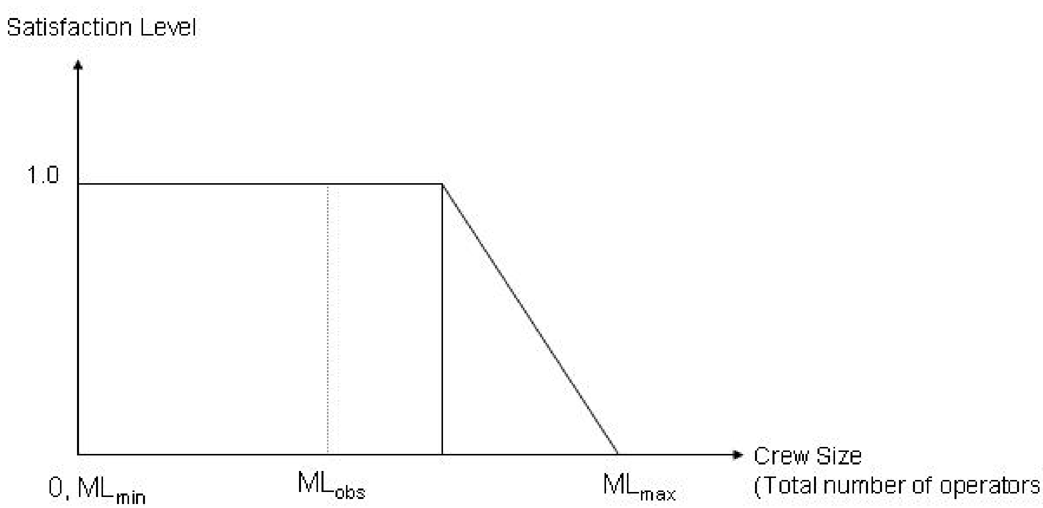

Figure 4.3: Crew Size computation

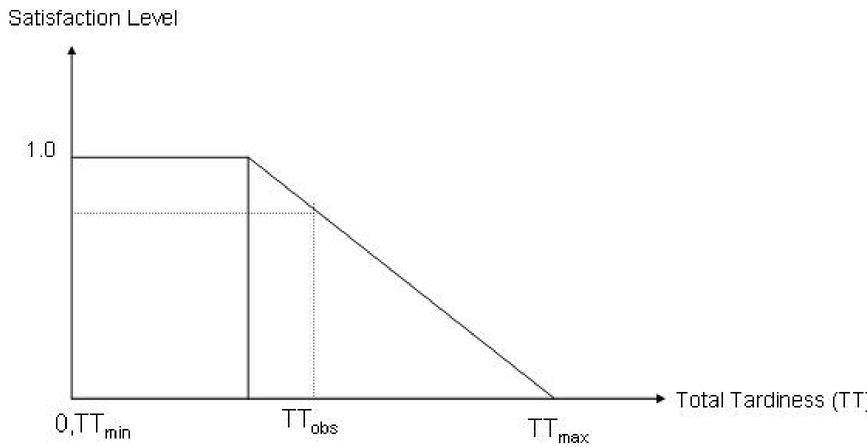

Figure 4.4: Total Tardiness computation

The maximum value of Satisfaction Level for both these solutions i.e. observed values is defined as the minimum of both these values

Satisfaction Level = Min $\{ML_{obs}, TT_{obs}\}$ (4.15)

Where,

ML_{obs} observed value of Manpower Levels in the Cell Loading model for a given acceptable TT

TT_{obs} observed value of Total Tardiness for the given Crew Size (total ML)

In this thesis the minimum (min) operator is chosen to compute the satisfaction level, and is the minimum of the λ values for TT and CS.

4.4 Addition operators for λ, $\lambda_1 + \lambda_2$ and $\lambda + \lambda_1 + \lambda_2$

The fuzzy operators used in the math model are λ, $\lambda_1 + \lambda_2$ and $\lambda + \lambda_1 + \lambda_2$. These are added to the Cell Loading models in Phase II as float values. The manpower level is added as an integer value. In the Cell loading models, the manpower level is an input to the model for e.g. a total crew size of 35, 40 etc, where as in the fuzzy math models, the manpower level is an output. The results for TT and CS obtained from the fuzzy math model are then compared against the optimal solution for the given CS in the Cell Loading model. The addition to the OPL code is as follows –

```
dvar int + manpower in 0..maxint;
dvar float+ lambda in 0..1;
dvar float+ lambda1 in 0..1;
dvar float+ lambda2 in 0..1;
```

By definition, the lambda1 and lambda2 were computed for the various ranges in the TT and CS intervals as follows -

lambda1 = $(TT_{max} – TT)/ (TT_{max} – TT_{min})$

lambda2 = $(CS_{max} – CS)/ (CS_{max} – CS_{min})$

The objective functions are based on the addition operator used and can be defined as one of the following in the OPL code –

1. maximize lambda; (4.16)
2. maximize lambda1 + lambda2; (4.17)
3. maximize lambda lambda1 + lambda2; (4.18)

The detailed OPL code can be found in the appendix.

CHAPTER V
RESULTS

The experiments focus on manpower allocations in a cell, cell loading to minimize the Total Tardiness and applying fuzzy mathematical models to determine the best levels for Total Tardiness and Manpower Level in a multi-cell manufacturing environment.

5.1 PHASE I: Manpower Allocation

Phase I experimentation deals with operator assignment for maximizing the production rates. The experiments are run for a single period for a given number of operations and total number of operators available for a single cell. Each product is run through the mathematical model independently, with the goal of determining the number of operators required per operation as well as the individual operator assignment(s) for each operation. This model is developed to understand the effects of production rates by varying the manpower levels as well as to understand the allocation (percentage of time that an operator can perform an operation) for each operator in the models I (B) and I (C). These models can expanded to account for the operator skill sets as well.

For the experimentation, a total of six operations are considered per cell. Each operation is assumed to have a fixed operation time which is known in advance. The operation times are modeled using a uniform distribution for each product, i.e. each product has a different set operation times for all six operations within the cell. The table 5.1 below illustrates the processing times for ten products (as a sample) for each operation within a cell. For example, Product 1 (P1) has unit operating times of 0.45, 0.38, 0.33, 0.82, 0.39 and 0.71 minutes for operations 1 through 6, respectively.

Table 5.1: Operation times for individual products (15 products)

PRODUCTS/ OPERATIONS	Op1	Op2	Op3	Op4	Op5	Op6
P1	0.45	0.38	0.33	0.82	0.39	0.71
P2	0.19	0.51	0.41	0.60	0.39	0.70
P3	0.48	0.63	0.18	0.52	0.29	0.23
P4	0.30	0.54	0.44	0.76	0.45	0.29
P5	0.37	0.65	0.30	0.37	0.33	0.70
P6	0.12	0.23	0.41	0.41	0.26	0.75
P7	0.50	0.58	0.44	0.76	0.25	0.26
P8	0.44	0.57	0.29	0.81	0.43	0.77
P9	0.30	0.48	0.21	0.71	0.42	0.31
P10	0.13	0.77	0.49	0.54	0.36	0.54
P11	0.43	0.21	0.46	0.96	0.26	0.36
P12	0.11	0.64	0.34	0.45	0.38	0.69
P13	0.49	0.26	0.45	0.45	0.42	0.86
P14	0.25	0.33	0.27	1.00	0.35	0.62
P15	0.33	0.34	0.35	0.64	0.27	0.69

Manpower allocation can be done in several ways depending on the flexibility desired. Each of the scenarios is explained in detail in the following sub sections -

5.1.1. No operator sharing allowed

The experiments are run for manpower levels varying from 10 operators per cell to 20 operators in a cell. Each experimental run is completed for a single manpower level e.g. 10 operators, then 11 operators in a cell, etc. The objective is to maximize the production rates, with constraints on the number of operators. These experiments are run for each product for a total of 15 products. In addition, the operators are not allowed to perform more than one operation.

The OPL run for this model (Phase I, Proposed Model I A) shows the mathematical model for 10 operators in a cell for the objective of maximizing the production rates. This model can be found in the appendix.

RESULTS FOR MODEL: Phase I (a), ML = 10, Product 1

Table 5.2: Results for Phase I, Model I (A), Product 1

R = 153.85 units/hr (max. production rate)					
Oprn I	Oprn II	Oprn III	Oprn IV	Oprn V	Oprn VI
1,7	6	8	4, 9, 10	2	3, 5

The results show that each operator is assigned to at exactly one operation. Operation I has operators 1 and 7; Operation II has operator 6; Operation II has operator 8 etc. Since the skill levels are similar, the model can potentially find alternative solutions e.g. Operation I can have operators 2 and 8 instead of 1 and 7. In other words, the number of operators is more important than their identity.

5.1.2. Operator sharing allowed without restrictions

In these experiments, the manpower levels are varied from 10 to 20 as in the previous case. The notable difference here being that operators are allowed to perform more than one operation within the same cell. The OPL code for this model can be found in the appendix. The table 5.3 shows the results for the run made for ten operators in the cell. Again, a total of 15 products were run for the different manpower levels.

RESULTS FOR MODEL: Phase I (b), ML = 10, Product 1

Table 5.3: Results for Phase I (B), Product 1

R = 194.805 (max. production rate)					
Oprn I	Oprn II	Oprn III	Oprn IV	Oprn V	Oprn VI
8 (0.46)	4 (0.36)	1 (0.07)	5 (0.12)	3 (0.62)	1 (0.93)
9 (1.0)	5 (0.88)	10 (1.0)	6 (1.0)	4 (0.64)	2 (1.0)
			7 (1.0)		3 (0.38)
			8 (0.54)		

The results show that an operator can perform more than one operation within a cell. For example operator 1 is allocated for 7% of the production time for Operation III and 93% of the time for operation VI. In this case too, the number of operators and percentage of time an individual operator spends on a particular operation is more important than their identity.

Table 5.4: Operator utilization for all ten operators (product 1)

Op 1	Op 2	Op 3	Op 4	Op 5	Op 6	Op 7	Op 8	Op 9	Op 10
0.93	1.0	0.38	0.64	0.88	1.0	1.0	0.54	1.0	1.0
0.07		0.62	0.36	0.12			0.46		

5.1.3. Operator sharing allowed with restrictions on number of operations

In these experiments, the manpower levels are varied from 10 to 20 as in the previous case. The notable difference here being that operators are allowed to perform more than one operation within the same cell. However, the maximum number of operations that can be performed by an operator is 2. Again, a total of 15 products are run for the different manpower levels. In this scenario, it is observed that the Production Rate (PR) stays the same as the Operator sharing without restrictions. For this reasons, 15 products were run in the experimentation. An observation is that certain runs in scenario I (b) result in an operator being assigned to 3 operations, whereas the run for the same ML in phase I (c) results in an operator assigned to 2 operation (the production rates remain the same in both scenarios). This can be attributed to the fact that the model finds a better fit for operator assignments in scenario I (c) i.e. better utilization of operators. The OPL code for the run made for a ten operators in the cell can be found in the appendix. Table 5.5 shows the results for this run, and table 5.6 shows the operator utilization. . In this case too, the number of operators and percentage of time an individual operator spends on a particular operation is more important than their identity.

RESULTS FOR MODEL: Phase I (c), ML = 10, Product 1

Table 5.5: Results for Phase I (c), Product 1

R = 194.805 units/ hour (max. production rate)					
Oprn I	Oprn II	Oprn III	Oprn IV	Oprn V	Oprn VI
3 (1.0)	1 (0.66)	2 (0.08)	4 (0.97)	1 (0.34)	7 (1.0)
4 (0.03)	6 (0.57)	5 (1.0)	9 (0.69)	2 (0.92)	8 (1.0)
6 (0.43)			10 (1.0)		9 (0.31)

Table 5.6: Operator utilization for all ten operators (product 1)

Op 1	Op 2	Op 3	Op 4	Op 5	Op 6	Op 7	Op 8	Op 9	Op 10
0.66	0.08	1.0	0.03	1.0	0.57	1.0	1.0	0.69	1.0
0.34	0.92		0.97		0.43				

5.1.4. No Operator sharing allowed with skill sets

This set of experiments introduce the concept of skill sets. A total of four skills levels were used for experimentation. These skill levels have been computed using the operation times used in 5.1.1, 5.1.2 and 5.1.3 with each skill level having its own set of operation times for each operation.

Table 5.7: Skill levels with operation times.

	Op1	Op2	Op3	Op4	Op5	Op6
skill level 1 (Best)	0.35	0.30	0.27	0.66	0.31	0.57
skill level 2	0.4	0.34	0.3	0.74	0.35	0.64
skill level 3	0.5	0.42	0.36	0.9	0.43	0.78
skill level 4 (Worst)	0.55	0.46	0.39	0.98	0.47	0.85

Each operator is randomly assigned a skill level (between 1 and 4) in a way that skill levels 2 and 3 have a 40% probability of each being chosen and level 1 and 4 have a 10% probability of being chosen.

Table 5.8: Operation times for different operators

PRODUCTS/ OPERATIONS	Skill Level	Op1	Op2	Op3	Op4	Op5	Op6
P1 (average)	**Average**	0.45	0.38	0.33	0.82	0.39	0.71
Operator 1	3	0.5	0.42	0.36	0.9	0.43	0.78
Operator 2	2	0.4	0.34	0.3	0.74	0.35	0.64
Operator 3	2	0.4	0.34	0.3	0.74	0.35	0.64
Operator 4	1	0.35	0.30	0.27	0.66	0.31	0.57
Operator 5	3	0.5	0.42	0.36	0.9	0.43	0.78
Operator 6	3	0.5	0.42	0.36	0.9	0.43	0.78
Operator 7	3	0.5	0.42	0.36	0.9	0.43	0.78
Operator 8	2	0.4	0.34	0.3	0.74	0.35	0.64
Operator 9	2	0.4	0.34	0.3	0.74	0.35	0.64
Operator 10	3	0.5	0.42	0.36	0.9	0.43	0.78
Operator 11	3	0.5	0.42	0.36	0.9	0.43	0.78
Operator 12	2	0.4	0.34	0.3	0.74	0.35	0.64
Operator 13	3	0.5	0.42	0.36	0.9	0.43	0.78
Operator 14	4	0.55	0.46	0.39	0.98	0.47	0.85

A total of five runs were made to compare the production rates for various manpower levels with varied skills and uniform skills. The results of the experiments can be found in table 5.9

Table 5.9: Comparison of Phase I (a) and I (d) results

	PR (varied skill Levels)	PR (uniform skill Levels)
ML = 10	176.47	153.85
ML = 11	187.50	157.89
ML = 12	200.00	169.01
ML = 13	222.22	181.83
ML = 14	253.07	219.51

5.1.5 RESULTS FOR PHASE I

Table 5.10 shows the results for the maximum production rates for scenarios I (a), I (b) and I (c) for a total of 10 operators in a cell. Phase I (A) dictates that no sharing is allowed for operators between operations, Phase I (B) allows sharing of operators between operations whereas Phase I (c) allows sharing of operators between operations.

In terms of maximum production rates under these scenarios, theoretically I (B) > I (C) > I (A). However, in the experimental runs, it has been observed that the production rates observed in I (C) equal to those observed in I (B).

Table 5.10: Maximum Production rates for scenario I (a), I (b) and I (c)

	Phase 1(A): No sharing	Phase 1(B): Sharing without restrictions	Phase 1(C): Sharing with restrictions (2 operations per operator)
Product 1	153.85	194.81	194.81
Product 2	153.85	214.29	214.29
Product 3	206.90	257.51	257.51
Product 4	157.89	215.83	215.83
Product 5	171.43	220.59	220.59
Product 6	230.77	275.23	275.23
Product 7	157.89	215.05	215.05
Product 8	139.53	181.27	181.27
Product 9	193.55	246.91	246.91
Product 10	155.84	212.01	212.01
Product 11	166.67	223.88	223.88
Product 12	173.91	229.89	229.89
Product 13	139.53	204.78	204.78
Product 14	180.00	212.77	212.77
Product 15	176.47	229.01	229.01

The detailed results for Phase I are inserted in the appendix.

5.2 PHASE II: Cell Loading

These set of experiments utilize the results for the production rates obtained from Phase I. Cell Loading also utilizes inputs for the due dates for each product, as well as well as the demand. All products are loaded to the three cells to determine the Total Tardiness (TT). Table 5.8 shows the setup (inputs) needed for the phase 2 experimentation for a total crew size (CS) of 30. The Due Dates are computed using random numbers, such that a product will have DD equaling 8,16,24,32 or 40. These due dates are considered as the end days for a week for a weekly schedule i.e. DD = 8 would mean that the product is due at the end of day 1, DD = 16 would mean that the product is due at the end of day 2 etc. Demand is computed using a uniform distribution between 1,000 and 5,000 for each product. This demand is determined such that 3 cells will be sufficient. The total processing time for a product (or the time a product stays in a cell) is computed by dividing the Demand by the Production Rate. The table 5.11 below shows the calculations using the maximum of Production Times obtained by the Phase I experiments for ML = 10 in each cell (these are observed in Phase I(b) and I (c), both of which have identical Production Rates). These setups are repeated for ML from 11 through 20. The objective is to study the effect of Total Tardiness with varying crew sizes.

Table 5.11: Setup for Phase II experiments

	Max Prod Rate (per hour)	Due Dates (hours)	Demand (no. of units)	Processing Time (in-cell time)
Product 1	194.81	16	3007	15.43
Product 2	214.29	40	4085	19.06
Product 3	257.51	24	2787	10.82
Product 4	215.83	16	2631	12.19
Product 5	220.59	24	2541	11.52
Product 6	275.23	16	1740	6.32
Product 7	215.05	32	2731	12.70
Product 8	181.27	32	1520	8.39
Product 9	246.91	24	2818	11.41
Product 10	212.01	32	1971	9.30
Product 11	223.88	40	4598	20.54
Product 12	229.89	32	2113	9.19
Product 13	204.78	8	3888	18.99
Product 14	212.77	16	4664	21.92
Product 15	229.01	16	1723	7.53

5.2.1: Experiment I – 15 products, 3 cells, 5 configurations (ML = 10, 11, 12, 13, 14)

In this set of experiments, a total of 13 runs are made by varying the Total Crew size in the interval of 30 to 42. The chosen ML for this experimentation is 10, 11, 12, 13 and 14 per cell. Different combinations of these will result in the overall Crew Size for example a combination of 13, 13 and 14 will yield a Crew Size of 40, which would mean that all 3 cells will be utilized. Table 5.12 shows the results for this set of experiments by varying the CS from 30 to 42

Table 5.12: Results for Experiment I, Phase II

Run	Crew Size	TT Phase I (B): sharing allowed
1	30	166.57
2	31	151.04
3	32	138.05
4	33	127.29
5	34	117.10
6	35	105.17
7	36	92.64
8	37	82.05
9	38	74.63
10	39	66.13
11	40	59.42
12	41	53.96
13	42	49.91

5.2.2: Experiment II – 15 products, 3 cells, 5 configurations (ML = 13, 14, 15, 16, 17)

In this set of experiments, a total of 7 runs are made by varying the Total Crew size in the interval of 30 to 40. The chosen ML for this experimentation is 13, 14, 15, 16 and 17 per cell. Different combinations of these will result in the overall Crew Size for example a combination of 13, 13 and 14 will yield a Crew Size of 40, which would mean that all 3 cells will be utilized. Conversely, a Crew size of 31 will utilize 2 cells, either with a 14-17 combination or a 15-16 combination for ML in each cell. The behavior of Cell Loading is examined in special cases to view cell utilization i.e. if all 3 cells are used, or of only 2 of the 3 available cells are used based on the total manpower level. The following model shows the results as a Gantt chart for the OPL code for a Total Crew size of 30. The actual model is in the appendix. Figure 5.1 shows the Gantt chart for ML configurations of 10,11,12,13 and 14 operators; whereas Figure 5.2 shows the similar experimentation for number of products (15), number of cells (3) and the number of configurations (5), with the operator levels options set as 13, 14, 15, 16, 17.

In figure 5.1 all 3 cells are utilized, and the tardiness of individual products represented on top of each product block in the Gantt chart. 10 operators are assigned to each of the 3 cells. The Total Tardiness is 166.57.

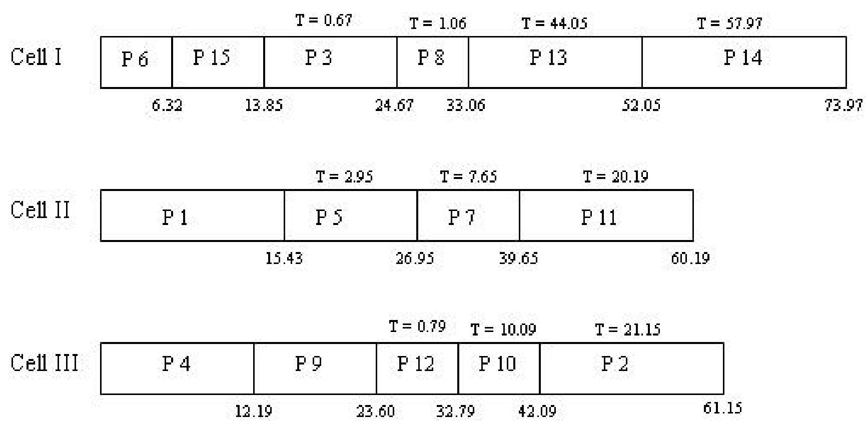

Figure 5.1: Gantt chart for 15-3-5 (Manpower level options 10, 11, 12, 13, 14)

The figure 5.2 shows that only Cell II and Cell III are utilized with 14 and 16 operators in the two cells respectively. The Total Tardiness is lower at 142.41.

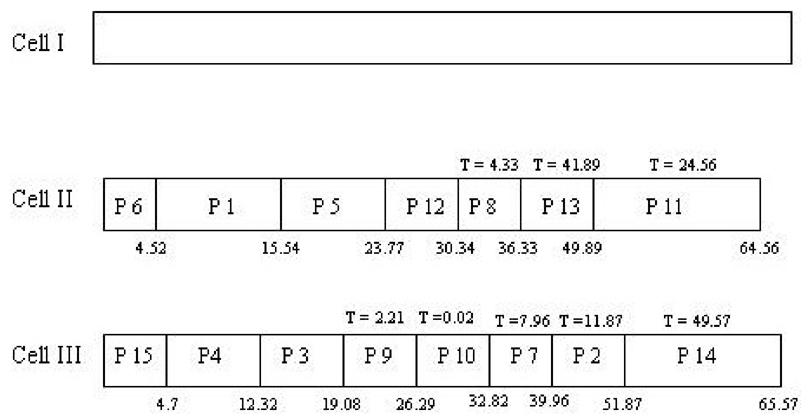

Figure 5.2: Gantt chart for 15-3-5 (Manpower level options 13, 14, 15, 16, 17)

The results to measure the TT for these runs are summarized in Table 5.13. As expected, the TT value will be lower by increasing the Crew Size from 30 to 40. Figure 5.3 shows the trends for the values of TT observed. The seven runs for the different CS were selected in such a way that the possible combinations of operators in each cell could total to the CS in the runs. For example a run with CS was avoided, as no combination of operators across two or three cells could sum upto 35.

Table 5.13: Results of Experiment II

Run	Crew Size	Total Tardiness (TT)
1	30	142.41
2	31	131.64
3	32	120.99
4	33	108.99
5	34	98.23
6	39	69.53
7	40	61.83

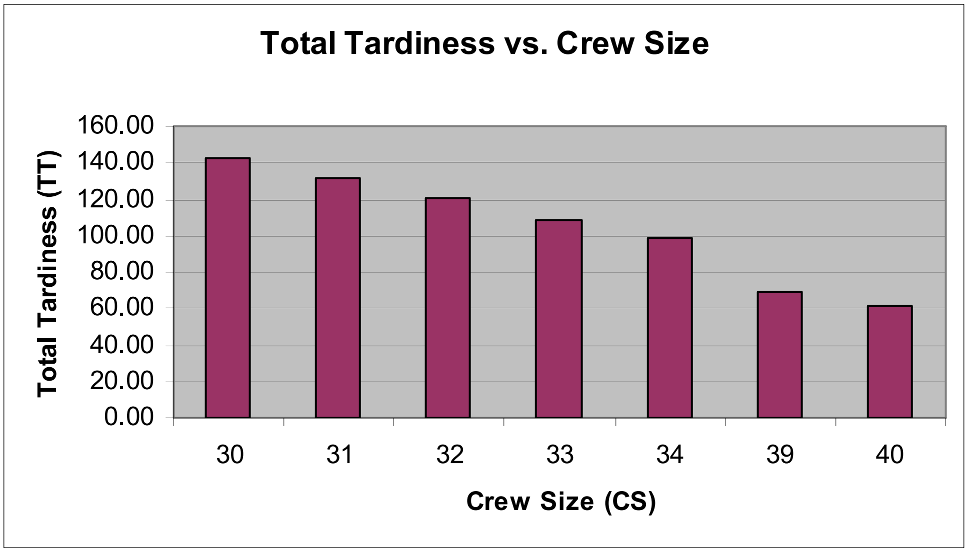

Figure 5.3: Total Tardiness vs. Crew Size for Experiment II

5.2.3: Experiment III – 15 products, 3 cells, 5 configurations (ML = 16, 17, 18, 19, 20)

In this set of experiments, a total of 6 runs are made by varying the Total Crew size in the interval of 35 to 40 with all intervals considered. The chosen ML for this experimentation is 16, 17, 18, 19 and 20 per cell. The TT is measured by varying the CS. These runs show that the TT is reduced for the same CS when compared to the runs made in Experiment II. The run for CS = 39 in Experiments II and III show a lower TT when the ML are varied between 16 and 20. Table 5.14 shows the variation of TT and the different Crew sizes. The last run (CS = 40) is a partial solution which was observed after the model had been running for over 25 hours (the experimentation aborted due to insufficient memory). Figure 5.4 shows the trend for TT against the CS.

Table 5.14: Results for Experiment III

Run	Crew Size	Total Tardiness (TT)
1	35	85.94
2	36	75.42
3	37	67.36
4	38	59.85
5	39	54.50
6	40	48.90

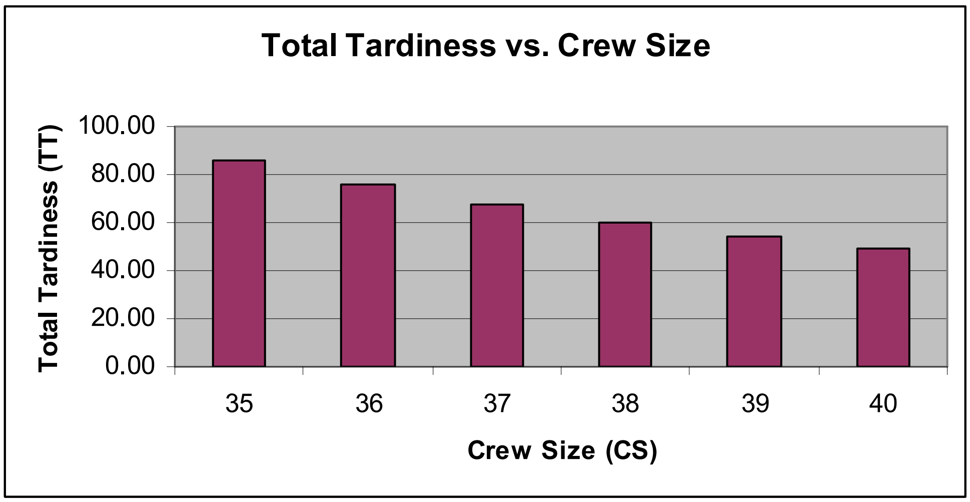

Figure 5.4: Total Tardiness vs. Crew Size for Experiment III

5.2.4: Experiment IV – 15 products, 3 cells, 10 configurations (ML = 10, 11, 12, 13, 14, 15, 16, 17, 18, 19)

For Experiment IV, two runs were made for a CS of 39 and 40. The Manpower level options were 10 i.e. each cell could have varying ML from 10 through 20.

Table 5.15: Results for Experiment IV

Run	Crew Size	Total Tardiness (TT)
1	39	64.74
2	40	58.27

The value of TT observed for CS = 40 is a partial solution which was observed after the model had been running for over 25 hours (the experimentation aborted due to insufficient memory).

5.2.5: Experiment V – 15 products, 3 cells, 10 configurations (ML = 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)

The setup for the two runs in Experiment V is very similar to Experiment IV, except for the fact that the ML configurations are between 11 and 20. The results for these two runs are summarized in the table below.

Table 5.16: Results for Experiment V

Run	**Crew Size**	**Total Tardiness (TT)**
1	39	54.50
2	40	48.90

Table 5.17 summarizes the results for all five sets of experiments (Experiments I to V) for Cell Loading for CS = 39 and 40. The results with (*) denote partial solutions which were aborted after the experiment had been running for over 25 hours.

Table 5.17: Summary of experiments for Phase II (for CS = 39 and 40)

Experiments/ CS	**I**	**II**	**III**	**IV**	**V**
39	66.13	69.53	54.50	64.74	54.50
40	59.42	61.83	48.9*	58.27*	48.90

5.3 Comparing results for Total Tardiness (sharing vs. non-sharing of operators)

The Total Tardiness for given demand and due dates would then depend on the Processing Times (PT) for each product. The PT in turn depends on the Production Rates. From Table 5.15 we see that the maximum production rates for I (B) i.e. sharing allowed between operators is higher than the Production Rates observed in I (A) i.e. no sharing allowed for individual operators between operations. Thus, it is a good comparison to compare the TT for both these scenarios.

The experimental conditions are 15 products, 3 cells and 5 configurations (ML = 10, 11, 12, 13, 14). Table 5.18 shows the performance for TT for the varying Crew Sized for the set of experiments. The figure 5.5 shows the trends for TT as the CS is increased for both options.

Table 5.18: Comparison for Total Tardiness based on operator sharing options

Run	**Crew Size**	**TT Phase I (B): sharing allowed**	**TT Phase I (I A): No sharing allowed**
1	30	166.57	273.49
2	31	151.04	268.38
3	32	138.05	222.06
4	33	127.29	207.09
5	34	117.10	187.00
6	35	105.17	170.16
7	36	92.64	162.15
8	37	82.05	147.30
9	38	74.63	135.61
10	39	66.13	129.51
11	40	59.42	124.69
12	41	53.96	124.69
13	42	49.91	120.33

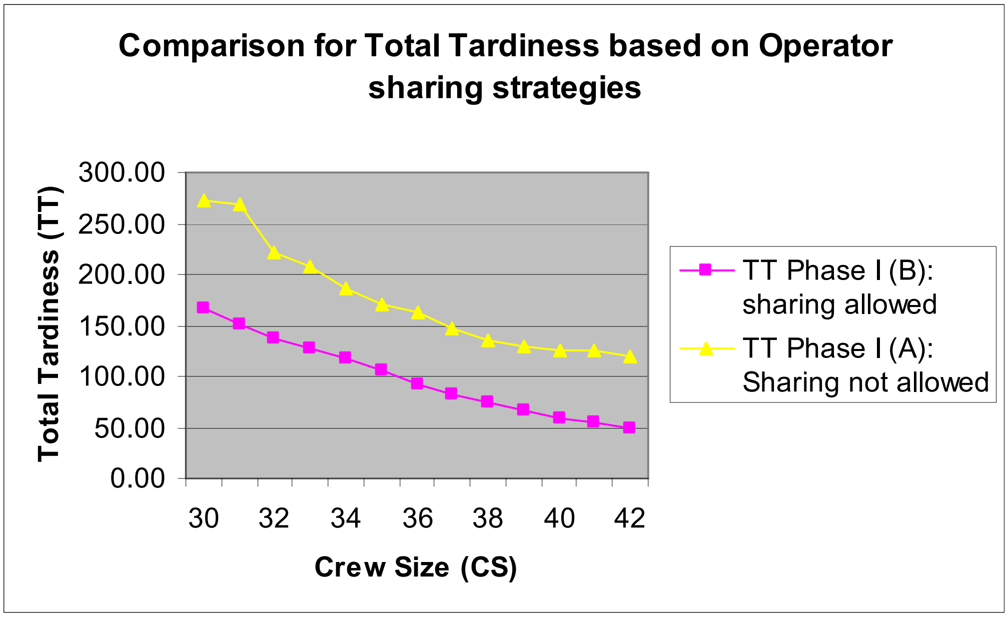

Figure 5.5: Comparison for Total Tardiness based on Operator sharing strategies

5.4 Fuzzy Mathematical Modeling

This set of experiments introduces fuzzy modeling which is a technique used to maximize the Satisfaction Level. The Satisfaction Level is dependent on Total Tardiness (lower TT values being more preferable) and the Crew Size (lower crew sizes being preferable).

The experimentation is done using the addition operator which suits the linear integer programming in OPL. The crew sizes and the Total Tardiness (TT) values were varied across all the experimentation sets. These results were then compared to the optimal results obtained by running the individual crew size models in Phase II i.e. Cell Loading models.

Overall the fuzzy intervals were defined in the ranges of 30 to 42 for manpower levels and 49.01 to 166.57 for Total Tardiness corresponding to the maximum and minimum values for TT (for the given CS) in Cell Loading. Within this range, five intervals for the TT and four ranges for CS were defined as shown in tables 5.17 and 5.18. The overall range was split at the quarter, mid and three fourths of the TT range for experimentation. A cost can be attributed to the Total Tardiness as well as the Crew Sizes in the models. The Total Cost would be the sum of the Total Tardiness cost as the Crew Size cost. In a real world scenario, the tardiness costs (in terms of lost sales, dissatisfied customers) as well as shortage costs are hard to determine.

- In terms of the mathematical model, the addition to the OPL code was in defining manpower, lambda (λ) values and the objective function, where $\lambda = \min(\lambda_1, \lambda_2)$.

The addition operator can be used in three ways –

- λ
- $\lambda_1 + \lambda_2$
- $\lambda + \lambda_1 + \lambda_2$
- Based on computation times, the second ($\lambda_1 + \lambda_2$) and the third ($\lambda + \lambda_1 + \lambda_2$) cases were most effective (least computation times). The first case (λ) took over 25 hours for a partial (best known) solution.

The table 5.16, table 5.17 and table 5.18 summarize the results for the 3 runs for Fuzzy modeling. The intervals for TT are established by considering the TT values corresponding to CS = 30 and 42, and varying them within those TT intervals (The ranges were defined using the differences between these TT values, and defining acceptable levels of TT for different sub sections of this range) results are equal to the results seen in Experiment I for the TT values observed for the scenario where operator sharing is allowed.

Table 5.19: λ Experimentation results

CREW SIZE	lambda 1	lambda 2	total (obj) function	TT	ML
30 TO 42	0.526	0.583	0.526	105.170	35
CS = 49.91 to 166.57					

Table 5.20: $\lambda_1 + \lambda_2$ Experimentation results

CREW SIZE	lambda 1	lambda 2	total (obj) function	TT	ML	abs (λ1 - λ2)
30 TO 42						
(i) 49.91 to 166.57	0.725	0.417	1.141	82.050	37	0.308
(ii) 83.24 to 133.24	1.000	0.583	1.583	108.240	37*	0.417
(iii) 108.24 to 166.57	1.000	0.583	1.583	108.240	35*	0.417
(iv) 49.91 to 108.24	0.583	0.005	0.589	105.170	35	0.578
(v) 49.91 to 79.08	0.503	0.083	0.587	53.960	41	0.420
CREW SIZE	**lambda 1**	**lambda 2**	**total (obj) function**	**TT**	**ML**	**abs (λ1 - λ2)**
30 TO 36						
(i) 49.91 to 166.57	0.000	1.000	1.000	166.570	30	1.000
(ii) 83.24 to 133.24	0.812	0.000	0.812	92.640	30	0.812
(iii) 108.24 to 166.57	0.848	0.333	1.814	117.100	34	0.515
(iv) 49.91 to 108.24	0.267	0.000	0.267	92.640	36	0.267
(v) 49.91 to 79.08	0.102	0.167	0.269	82.050	37	0.065
CREW SIZE	**lambda 1**	**lambda 2**	**total (obj) function**	**TT**	**ML**	**abs (λ1 - λ2)**
33 to 39						
(i) 49.91 to 166.57	0.337	1.000	1.337	127.290	33	0.663
(ii) 83.24 to 133.24	0.333	1.000	1.333	83.240	37*	0.667
(iii) 108.24 to 166.57	0.848	0.833	1.681	117.100	34	0.015
(iv) 49.91 to 108.24	0.449	0.333	0.782	82.050	37	0.116
(v) 49.91 to 79.08	0.444	0.000	0.444	66.130	39	0.444

CREW SIZE	lambda 1	lambda 2	total (obj) function	TT	ML	abs (λ1 - λ2)
30 TO 39						
(i) 49.91 to 166.57	0.244	0.778	1.022	138.050	32	0.533
(ii) 83.24 to 133.24	1.000	0.222	1.222	83.240	37*	0.778
(iii) 108.24 to 166.57	0.989	0.444	1.444	108.850	35*	0.545
(iv) 49.91 to 108.24	0.722	0.000	0.722	66.130	39	0.722
(v) 49.91 to 79.08	0.443	0.000	0.444	66.130	39	0.443
				Max difference		1.000
				Min Difference		0.015
				Avg. difference		0.486

Table 5.21: $\lambda + \lambda_1 + \lambda_2$ Experimentation results

CREW SIZE	lambda	lambda 1	lambda 2	total (obj) function	TT	ML	abs (λ1 - λ2)
30 TO 42							
(i) 49.91 to 166.57	0.526	0.526	0.583	1.636	105.170	35	0.057
(ii) 83.24 to 133.24	0.417	1.000	0.417	1.833	83.240	37*	0.583
(iii) 108.24 to 166.57	0.667	0.848	0.667	2.181	117.100	34	0.181
(iv) 49.91 to 108.24	0.417	0.449	0.417	1.282	82.050	37	0.032
(v) 49.91 to 79.08	0.083	0.862	0.083	1.028	53.960	41	0.778
CREW SIZE	**lambda**	**lambda 1**	**lambda 2**	**total (obj) function**	**TT**	**ML**	**abs (λ1 - λ2)**
30 TO 36							
(i) 49.91 to 166.57	0.334	0.337	0.500	1.173	127.290	33	0.163
(ii) 83.24 to 133.24	0.323	0.323	0.333	0.979	117.100	34	0.011
(iii) 108.24 to 166.57	0.500	0.673	0.500	1.673	127.290	33	0.173
(iv) 49.91 to 108.24	0.053	0.053	0.167	0.272	105.170	35	0.114
(v) 49.91 to 79.08	0.167	0.102	0.167	0.435	82.050	37	0.065
CREW SIZE	**lambda**	**lambda 1**	**lambda 2**	**total (obj) function**	**TT**	**ML**	**abs (λ1 - λ2)**
33 to 39							
(i) 49.91 to 166.57	0.562	0.562	0.667	1.719	105.170	35	0.104
(ii) 83.24 to 133.24	0.500	0.812	0.500	1.812	92.640	36	0.312
(iii) 108.24 to 166.57	0.833	0.848	0.833	2.515	117.100	34	0.015
(iv) 49.91 to 108.24	0.333	0.449	0.333	1.116	82.050	37	0.116
(v) 49.91 to 79.08	0.153	0.153	0.167	0.472	74.630	38	0.014
CREW SIZE	**lambda**	**lambda 1**	**lambda 2**	**total (obj) function**	**TT**	**ML**	**abs (λ1 - λ2)**
30 TO 39							
(i) 49.91 to 166.57	0.444	0.526	0.444	1.415	105.170	35	0.082
(ii) 83.24 to 133.24	0.333	0.812	0.333	1.479	92.640	36	0.479
(iii) 108.24 to 166.57	0.667	0.573	0.667	2.007	127.290	33	0.093
(iv) 49.91 to 108.24	0.222	0.449	0.222	0.893	82.050	37	0.227
(v) 49.91 to 79.08	0.000	0.444	0.000	0.444	66.130	39	0.444

Max difference	0.778
Min difference	0.00

Comparing the tables 5.17 and table 5.18; we see that $\lambda + \lambda_1 + \lambda_2$ provides the optimal solution on more runs (19 runs out of 20) as compared to $\lambda_1 + \lambda_2$ (15 times out of 20). The instances where the results differ from the optimal solution are represented by '*' in the ML field to suggest that these differ from the optimal solution. The absolute difference in the λ_1 and λ_2 is computed for all the runs in both cases and the average difference is lower in the case of $\lambda + \lambda_1 + \lambda_2$ which would suggest that the λ_1 and λ_2 values are closer which implies that the model is more motivated to find the optimal solution.

CHAPTER VI
CONCLUSIONS AND FUTURE WORK

6.1 CONCLUSIONS

The first phase in this thesis attempts to allocate operators to different operations within a cell, with a focus on individual operator assignment. Three different sharing strategies were examined by measuring the production rates under each scenario (no operator sharing allowed, sharing allowed without restrictions, sharing allowed with restrictions). The last scenario for sharing with restriction dictated that no operator could perform more than 2 operations within a cell. Under these sharing scenarios, it was observed that the production rates were maximum for the scenario where operator sharing was allowed. Interesting enough, the sharing restrictions did not affect the production rates (the PR remained the same for sharing and sharing with restrictions; both outperformed the scenario where no operator sharing was allowed). In some cases, for the same PR, an operator that performed 3 operations within a cell (sharing without restrictions), performed only 2 operations (sharing with restrictions). This can be attributed to the model finding a better fit and utilizing the operators more effectively.

The second phase for cell loading examined the relationship between the Total Tardiness and the Crew Size for all cells. As expected, the Total Tardiness decreased as the Crew Size was increased. Depending on the crew size selected the crisp mathematical model returned the total Tardiness, as well as the configuration (number of operators in each cell). For equal crew sizes, different configurations yielded different levels of TT, which implies that the configuration is an important issue in cell scheduling along with the Crew Size.

An extension of the second phase included fuzzy mathematical model. Different addition operators were examined to test the solution (for Total Tardiness and Crew Size) by comparing the output against the crisp mathematical model for cell loading. The $\lambda + \lambda 1 +$

$\lambda 2$ operator outperformed the $\lambda 1 + \lambda 2$ operator in terms of the number of times it found the optimal solution for a number of runs. The experimental runs were varied to include a range of acceptable Total Tardiness values). The fuzzy math model approach helps a manufacturing unit to determine a weekly schedule for a number of products in a multi cell environment pretty effectively in terms of computation time; as well as the need to run the model only once.

6.2 Future Work

This work can be extended to include the following issues encountered in a manufacturing environment –

- Examining restrictions in terms of the minimum allowable time an operator can be allowed to perform an operation. In a real world scenario, this would mean that the operator can have a better learning curve as he/ she would not need to spend a short time on an operation, but focus on an operation for a more extended time
- Examining restrictions in terms of the distance (number of stations between successive operations) an operator can travel in a cell.
- Examining individual operator skills and operator preference in manpower allocation.
- Examining the maximum number of products that can be scheduled weekly and the number of configuration levels possible.
- In this work, the addition operators were used in the fuzzy math model. In the future, other operators can be studied to understand their effects on fuzzy math modeling for a cell loading problem.
- Cost analysis for Total Tardiness and Crew Sizes.

REFERENCES

[1] Askin, R.G. and Skandrige C.R., Modeling and analysis of manufacturing systems, 1993.

[2] Süer G.A., “Optimal operator assignment and cell loading in labor intensive manufacturing cells”, Computers and Industrial Engineering, vol 31, no 1/2, pp. 155-158, 1996.

[3] Drolet, J., Abdulnor, G. and Rhealt, M, The Cellular Manufacturing Revolution. Computers and Industrial Engineering, 31 (1/2), 139- 142, 1996.

[4] Du, J., Leung J., “Minimizing Total Tardiness on one machine is NP-hard”, Mathematics of Operation Research, vol 15, no 3, August 1990.

[5] Baker KR and Bertrand JW. A Dynamic Priority Rule for Scheduling against Due-dates. Journal of Operational Management; 3: 37-42., 1982.

[6] Kanet, J and Li X., “A weighted modified due date rule for sequencing to minimize weighted tardiness”, Journal of scheduling, vol 7, 261- 276, 2004.

[7] Bilge U., Kirac, F., Kurtulan M., Pekgun P., “A tabu search algorithm for parallel machine total tardiness problem”, Computers and Operations Research, vol 31, 397- 414, 2002.

[8] Hu, P., “Further study of minimizing total tardiness for the worker assignment scheduling problem in the identical parallel- machine models”, International Journal of Advanced Manufacturing Technology, vol 29, 65- 169, 2006.

[9] Süer, G.A., and Bera, I.S., "Multi period cell loading and cell size determination", Computers and Industrial Engineering, vol 35, no 1- 2, 85- 88, 1998.

[10] Wittrock, R.J., "Operator assignment and the parametric preflow algorithm", Management Science, vol 38, no 9, September 1992.

[11] Süer, G.A., Saiz, M. and Gonzalez W., "Evaluation of manufacturing cell loading rules for independent cells", International Journal of Production Research,vol 37, no 15, 3445- 3468, 1999.

[12] Süer, G.A., Gomez A.M., "A Hierarchical Hybrid Approach to Cell Loading, Manpower Allocation and Job Sequencing in Cellular Manufacturing", 2002.

[13] Süer, G.A., Arikan, F., Babayigit C., "Effects of different fuzzy operators on fuzzy bi- objective cell loading problem in labor intensive manufacturing cells", 35th International Conference on Computers and Industrial Engineering, 1825- 1830.

[14] Gupta, N.D and Chantaravaraparan, S., "Single machine group scheduling with family setups to minimize total tardiness", International Journal of Production Research, 1-16, 2007.

[15] Holsenback, J.E. and Russell, R.M., "A heuristic algorithm for sequencing on one machine to minimize the total tardiness", vol 43, Journal of Operations Research Society, 53- 62, 1992.

[16] Panwalkar, S.S., Smith, M.L., and Kolamas, C.P., A heuristic for the single machine tardiness problem, European Journal of Operations Research, 70, 304- 310, 1993.

[17] Werners, B.M., "Aggregation models in mathematical programming", Mathematical models for decision support, Springer, Berlin, 295- 305, 1988.

[18] Bahram Alidaee and Duane Rosa, "Scheduling parallel machines to minimize total weighted and unweighted tardiness", Computers & Operations Research, Vol 24, No.8, 775 – 788, 1997.

APPENDIX

Experiment: Phase 1, Proposed Model 1A: No sharing operators between operations

```
// Manpower Allocation OPL model (Single product, 6 operations to be completed
// by a maximum of n operators)
// Defining the Operations performed by the Operator i.e. Operator 1, Operator 2 etc.
// y (i,j) is the Operator i assigned to Operation j
```

// 10 products, 6 operations, 10 operators (Product 1)

```
// Operation 1
dvar int+ y11 in 0..maxint;
dvar int+ y21 in 0..maxint;
dvar int+ y31 in 0..maxint;
dvar int+ y41 in 0..maxint;
dvar int+ y51 in 0..maxint;
dvar int+ y61 in 0..maxint;
dvar int+ y71 in 0..maxint;
dvar int+ y81 in 0..maxint;
dvar int+ y91 in 0..maxint;
dvar int+ y101 in 0..maxint;

// Operation 2
dvar int+ y12 in 0..maxint;
dvar int+ y22 in 0..maxint;
dvar int+ y32 in 0..maxint;
dvar int+ y42 in 0..maxint;
dvar int+ y52 in 0..maxint;
dvar int+ y62 in 0..maxint;
dvar int+ y72 in 0..maxint;
```

```
dvar int+ y82 in 0..maxint;
dvar int+ y92 in 0..maxint;
dvar int+ y102 in 0..maxint;

// Operation 3
dvar int+ y13 in 0..maxint;
dvar int+ y23 in 0..maxint;
dvar int+ y33 in 0..maxint;
dvar int+ y43 in 0..maxint;
dvar int+ y53 in 0..maxint;
dvar int+ y63 in 0..maxint;
dvar int+ y73 in 0..maxint;
dvar int+ y83 in 0..maxint;
dvar int+ y93 in 0..maxint;
dvar int+ y103 in 0..maxint;

// Operation 4
dvar int+ y14 in 0..maxint;
dvar int+ y24 in 0..maxint;
dvar int+ y34 in 0..maxint;
dvar int+ y44 in 0..maxint;
dvar int+ y54 in 0..maxint;
dvar int+ y64 in 0..maxint;
dvar int+ y74 in 0..maxint;
dvar int+ y84 in 0..maxint;
dvar int+ y94 in 0..maxint;
dvar int+ y104 in 0..maxint;

// Operation 5
dvar int+ y15 in 0..maxint;
```

```
dvar int+ y25 in 0..maxint;
dvar int+ y35 in 0..maxint;
dvar int+ y45 in 0..maxint;
dvar int+ y55 in 0..maxint;
dvar int+ y65 in 0..maxint;
dvar int+ y75 in 0..maxint;
dvar int+ y85 in 0..maxint;
dvar int+ y95 in 0..maxint;
dvar int+ y105 in 0..maxint;

// Operation 6
dvar int+ y16 in 0..maxint;
dvar int+ y26 in 0..maxint;
dvar int+ y36 in 0..maxint;
dvar int+ y46 in 0..maxint;
dvar int+ y56 in 0..maxint;
dvar int+ y66 in 0..maxint;
dvar int+ y76 in 0..maxint;
dvar int+ y86 in 0..maxint;
dvar int+ y96 in 0..maxint;
dvar int+ y106 in 0..maxint;


// Defining Z as the value of R where R is the production rate
dvar float+ z in 0..maxint;
dvar float+ R in 0..maxint;

maximize z; //objective function to find the maximum production rate

subject to
```

```
{
z == R;
(60/0.45)* (y11+ y21+ y31+ y41+ y51+ y61+ y71+ y81+ y91+ y101)- R >= 0;
(60/0.38)* (y12+ y22+ y32+ y42+ y52+ y62+ y72+ y82+ y92+ y102)- R >= 0;
(60/0.33)* (y13+ y23+ y33+ y43+ y53+ y63+ y73+ y83+ y93+ y103)- R >= 0;
(60/0.82)* (y14+ y24+ y34+ y44+ y54+ y64+ y74+ y84+ y94+ y104)- R >= 0;
(60/0.39)* (y15+ y25+ y35+ y45+ y55+ y65+ y75+ y85+ y95+ y105)- R >= 0;
(60/0.71)* (y16+ y26+ y36+ y46+ y56+ y66+ y76+ y86+ y96+ y106)- R >= 0;


y11+ y12+ y13+ y14+ y15+ y16 == 1; // Operator can be assigned to only one Operation
y21+ y22+ y23+ y24+ y25+ y26 == 1; // Remember as "Operator to Operation"
y31+ y32+ y33+ y34+ y35+ y36 == 1;
y41+ y42+ y43+ y44+ y45+ y46 == 1;
y51+ y52+ y53+ y54+ y55+ y56 == 1;
y61+ y62+ y63+ y64+ y65+ y66 == 1;
y71+ y72+ y73+ y74+ y75+ y76 == 1;
y81+ y82+ y83+ y84+ y85+ y86 == 1;
y91+ y92+ y93+ y94+ y95+ y96 == 1;
y101+ y102+ y103+ y104+ y105+ y106 == 1;

};
```

Example Phase 1, Proposed Model 1B: Sharing operators between operations allowed without any restrictions

```
// Manpower Allocation OPL model (Single product, 6 operations to be completed
// by a maximum of n operators)
// Defining the Operations performed by the Operator i.e. Operator 1, Operator 2 etc.
// y (i,j) is the Operator i assigned to Operation j
```

// 10 products, 6 operations, 10 operators (Product 1)

```
dvar float+ y11 in 0..maxint;
dvar float+ y21 in 0..maxint;
dvar float+ y31 in 0..maxint;
dvar float+ y41 in 0..maxint;
dvar float+ y51 in 0..maxint;
dvar float+ y61 in 0..maxint;
dvar float+ y71 in 0..maxint;
dvar float+ y81 in 0..maxint;
dvar float+ y91 in 0..maxint;
dvar float+ y101 in 0..maxint;

// Operation 2
dvar float+ y12 in 0..maxint;
dvar float+ y22 in 0..maxint;
dvar float+ y32 in 0..maxint;
dvar float+ y42 in 0..maxint;
dvar float+ y52 in 0..maxint;
dvar float+ y62 in 0..maxint;
dvar float+ y72 in 0..maxint;
dvar float+ y82 in 0..maxint;
```

```
dvar float+ y92 in 0..maxint;
dvar float+ y102 in 0..maxint;

// Operation 3
dvar float+ y13 in 0..maxint;
dvar float+ y23 in 0..maxint;
dvar float+ y33 in 0..maxint;
dvar float+ y43 in 0..maxint;
dvar float+ y53 in 0..maxint;
dvar float+ y63 in 0..maxint;
dvar float+ y73 in 0..maxint;
dvar float+ y83 in 0..maxint;
dvar float+ y93 in 0..maxint;
dvar float+ y103 in 0..maxint;

// Operation 4
dvar float+ y14 in 0..maxint;
dvar float+ y24 in 0..maxint;
dvar float+ y34 in 0..maxint;
dvar float+ y44 in 0..maxint;
dvar float+ y54 in 0..maxint;
dvar float+ y64 in 0..maxint;
dvar float+ y74 in 0..maxint;
dvar float+ y84 in 0..maxint;
dvar float+ y94 in 0..maxint;
dvar float+ y104 in 0..maxint;

// Operation 5
dvar float+ y15 in 0..maxint;
dvar float+ y25 in 0..maxint;
```

```
dvar float+ y35 in 0..maxint;
dvar float+ y45 in 0..maxint;
dvar float+ y55 in 0..maxint;
dvar float+ y65 in 0..maxint;
dvar float+ y75 in 0..maxint;
dvar float+ y85 in 0..maxint;
dvar float+ y95 in 0..maxint;
dvar float+ y105 in 0..maxint;

// Operation 6
dvar float+ y16 in 0..maxint;
dvar float+ y26 in 0..maxint;
dvar float+ y36 in 0..maxint;
dvar float+ y46 in 0..maxint;
dvar float+ y56 in 0..maxint;
dvar float+ y66 in 0..maxint;
dvar float+ y76 in 0..maxint;
dvar float+ y86 in 0..maxint;
dvar float+ y96 in 0..maxint;
dvar float+ y106 in 0..maxint;


// Defining Z as the value of R where R is the production rate
dvar float+ z in 0..maxint;
dvar float+ R in 0..maxint;

maximize z; //objective function to find the maximum production rate

subject to
{
```

```
z == R;
(60/0.45)* (y11+ y21+ y31+ y41+ y51+ y61+ y71+ y81+ y91+ y101)- R >=0;
(60/0.38)* (y12+ y22+ y32+ y42+ y52+ y62+ y72+ y82+ y92+ y102)- R >=0;
(60/0.33)* (y13+ y23+ y33+ y43+ y53+ y63+ y73+ y83+ y93+ y103)- R >=0;
(60/0.82)* (y14+ y24+ y34+ y44+ y54+ y64+ y74+ y84+ y94+ y104)- R >=0;
(60/0.39)* (y15+ y25+ y35+ y45+ y55+ y65+ y75+ y85+ y95+ y105)- R >=0;
(60/0.71)* (y16+ y26+ y36+ y46+ y56+ y66+ y76+ y86+ y96+ y106)- R >=0;

y11+ y12+ y13+ y14+ y15+ y16 == 1; // Operator can be assigned to only one Operation
y21+ y22+ y23+ y24+ y25+ y26 == 1; // Remember as "Operator to Operation"
y31+ y32+ y33+ y34+ y35+ y36 == 1;
y41+ y42+ y43+ y44+ y45+ y46 == 1;
y51+ y52+ y53+ y54+ y55+ y56 == 1;
y61+ y62+ y63+ y64+ y65+ y66 == 1;
y71+ y72+ y73+ y74+ y75+ y76 == 1;
y81+ y82+ y83+ y84+ y85+ y86 == 1;
y91+ y92+ y93+ y94+ y95+ y96 == 1;
y101+ y102+ y103+ y104+ y105+ y106 == 1;

};
```

Example Phase 1, Proposed Model C: sharing (with restrictions) allowed between Operations for a single Operator

```
// Manpower Allocation OPL model (Single product, 6 operations to be completed
// by a maximum of n operators)
// Defining the Operations performed by the Operator i.e. Operator 1, Operator 2 etc.
// y (i,j) is the Operator i assigned to Operation
```

// 10 products, 6 operations, 10 operators (Product 1)

```
// Operation 1
dvar float+ y11 in 0..maxint;
dvar float+ y21 in 0..maxint;
dvar float+ y31 in 0..maxint;
dvar float+ y41 in 0..maxint;
dvar float+ y51 in 0..maxint;
dvar float+ y61 in 0..maxint;
dvar float+ y71 in 0..maxint;
dvar float+ y81 in 0..maxint;
dvar float+ y91 in 0..maxint;
dvar float+ y101 in 0..maxint;

// Operation 2
dvar float+ y12 in 0..maxint;
dvar float+ y22 in 0..maxint;
dvar float+ y32 in 0..maxint;
dvar float+ y42 in 0..maxint;
dvar float+ y52 in 0..maxint;
dvar float+ y62 in 0..maxint;
dvar float+ y72 in 0..maxint;
```

```
dvar float+ y82 in 0..maxint;
dvar float+ y92 in 0..maxint;
dvar float+ y102 in 0..maxint;

// Operation 3
dvar float+ y13 in 0..maxint;
dvar float+ y23 in 0..maxint;
dvar float+ y33 in 0..maxint;
dvar float+ y43 in 0..maxint;
dvar float+ y53 in 0..maxint;
dvar float+ y63 in 0..maxint;
dvar float+ y73 in 0..maxint;
dvar float+ y83 in 0..maxint;
dvar float+ y93 in 0..maxint;
dvar float+ y103 in 0..maxint;

// Operation 4
dvar float+ y14 in 0..maxint;
dvar float+ y24 in 0..maxint;
dvar float+ y34 in 0..maxint;
dvar float+ y44 in 0..maxint;
dvar float+ y54 in 0..maxint;
dvar float+ y64 in 0..maxint;
dvar float+ y74 in 0..maxint;
dvar float+ y84 in 0..maxint;
dvar float+ y94 in 0..maxint;
dvar float+ y104 in 0..maxint;

// Operation 5
dvar float+ y15 in 0..maxint;
```

```
dvar float+ y25 in 0..maxint;
dvar float+ y35 in 0..maxint;
dvar float+ y45 in 0..maxint;
dvar float+ y55 in 0..maxint;
dvar float+ y65 in 0..maxint;
dvar float+ y75 in 0..maxint;
dvar float+ y85 in 0..maxint;
dvar float+ y95 in 0..maxint;
dvar float+ y105 in 0..maxint;

// Operation 6
dvar float+ y16 in 0..maxint;
dvar float+ y26 in 0..maxint;
dvar float+ y36 in 0..maxint;
dvar float+ y46 in 0..maxint;
dvar float+ y56 in 0..maxint;
dvar float+ y66 in 0..maxint;
dvar float+ y76 in 0..maxint;
dvar float+ y86 in 0..maxint;
dvar float+ y96 in 0..maxint;
dvar float+ y106 in 0..maxint;


// Defining w as the integer value for a real, positive value of y(i,j)
dvar int+ w11 in 0..maxint;
dvar int+ w12 in 0..maxint;
dvar int+ w13 in 0..maxint;
dvar int+ w14 in 0..maxint;
dvar int+ w15 in 0..maxint;
dvar int+ w16 in 0..maxint;
```

```
// Operator 2, constraint as to how many Operations can be performed by this particular Operator
dvar int+ w21 in 0..maxint;
dvar int+ w22 in 0..maxint;
dvar int+ w23 in 0..maxint;
dvar int+ w24 in 0..maxint;
dvar int+ w25 in 0..maxint;
dvar int+ w26 in 0..maxint;

// Operator 3, constraint as to how many Operations can be performed by this particular Operator
dvar int+ w31 in 0..maxint;
dvar int+ w32 in 0..maxint;
dvar int+ w33 in 0..maxint;
dvar int+ w34 in 0..maxint;
dvar int+ w35 in 0..maxint;
dvar int+ w36 in 0..maxint;

// Operator 4, constraint as to how many Operations can be performed by this particular Operator
dvar int+ w41 in 0..maxint;
dvar int+ w42 in 0..maxint;
dvar int+ w43 in 0..maxint;
dvar int+ w44 in 0..maxint;
dvar int+ w45 in 0..maxint;
dvar int+ w46 in 0..maxint;

// Operator 5, constraint as to how many Operations can be performed by this particular Operator
dvar int+ w51 in 0..maxint;
```

```
dvar int+ w52 in 0..maxint;
dvar int+ w53 in 0..maxint;
dvar int+ w54 in 0..maxint;
dvar int+ w55 in 0..maxint;
dvar int+ w56 in 0..maxint;

// Operator 6, constraint as to how many Operations can be performed by this particular
Operator
dvar int+ w61 in 0..maxint;
dvar int+ w62 in 0..maxint;
dvar int+ w63 in 0..maxint;
dvar int+ w64 in 0..maxint;
dvar int+ w65 in 0..maxint;
dvar int+ w66 in 0..maxint;

// Operator 7, constraint as to how many Operations can be performed by this particular
Operator
dvar int+ w71 in 0..maxint;
dvar int+ w72 in 0..maxint;
dvar int+ w73 in 0..maxint;
dvar int+ w74 in 0..maxint;
dvar int+ w75 in 0..maxint;
dvar int+ w76 in 0..maxint;

// Operator 8, constraint as to how many Operations can be performed by this particular
Operator
dvar int+ w81 in 0..maxint;
dvar int+ w82 in 0..maxint;
dvar int+ w83 in 0..maxint;
dvar int+ w84 in 0..maxint;
```

```
dvar int+ w85 in 0..maxint;
dvar int+ w86 in 0..maxint;

// Operator 9, constraint as to how many Operations can be performed by this particular
Operator
dvar int+ w91 in 0..maxint;
dvar int+ w92 in 0..maxint;
dvar int+ w93 in 0..maxint;
dvar int+ w94 in 0..maxint;
dvar int+ w95 in 0..maxint;
dvar int+ w96 in 0..maxint;

// Operator 10, constraint as to how many Operations can be performed by this particular
Operator
dvar int+ w101 in 0..maxint;
dvar int+ w102 in 0..maxint;
dvar int+ w103 in 0..maxint;
dvar int+ w104 in 0..maxint;
dvar int+ w105 in 0..maxint;
dvar int+ w106 in 0..maxint;


// Defining z as the value of R where R is the production rate
dvar float+ z in 0..maxint;
dvar float+ R in 0..maxint;


maximize z; //objective function to find the maximum production rate

subject to
```

```
{

// Finding maximum Production Rate for the process for product 1

z == R;
(60/0.45)* (y11+ y21+ y31+ y41+ y51+ y61+ y71+ y81+ y91+ y101)- R >=0;
(60/0.38)* (y12+ y22+ y32+ y42+ y52+ y62+ y72+ y82+ y92+ y102)- R >=0;
(60/0.33)* (y13+ y23+ y33+ y43+ y53+ y63+ y73+ y83+ y93+ y103)- R >=0;
(60/0.82)* (y14+ y24+ y34+ y44+ y54+ y64+ y74+ y84+ y94+ y104)- R >=0;
(60/0.39)* (y15+ y25+ y35+ y45+ y55+ y65+ y75+ y85+ y95+ y105)- R >=0;
(60/0.71)* (y16+ y26+ y36+ y46+ y56+ y66+ y76+ y86+ y96+ y106)- R >=0;

y11+ y12+ y13+ y14+ y15+ y16 == 1; // Operator can be assigned to only one Operation
y21+ y22+ y23+ y24+ y25+ y26 == 1; // Remember as "Operator to Operation"
y31+ y32+ y33+ y34+ y35+ y36 == 1;
y41+ y42+ y43+ y44+ y45+ y46 == 1;
y51+ y52+ y53+ y54+ y55+ y56 == 1;
y61+ y62+ y63+ y64+ y65+ y66 == 1;
y71+ y72+ y73+ y74+ y75+ y76 == 1;
y81+ y82+ y83+ y84+ y85+ y86 == 1;
y91+ y92+ y93+ y94+ y95+ y96 == 1;
y101+ y102+ y103+ y104+ y105+ y106 == 1;


w11- y11 >= 0; // Linking integer value of w to the real value of y
w12- y12 >= 0; // for each instance of Operator - Operation assignment
w13- y13 >= 0;
w14- y14 >= 0;
w15- y15 >= 0;
w16- y16 >= 0;
```

w21- y21 >= 0;
w22- y22 >= 0;
w23- y23 >= 0;
w24- y24 >= 0;
w25- y25 >= 0;
w26- y26 >= 0;

w31- y31 >= 0;
w32- y32 >= 0;
w33- y33 >= 0;
w34- y34 >= 0;
w35- y35 >= 0;
w36- y36 >= 0;

w41- y41 >= 0;
w42- y42 >= 0;
w43- y43 >= 0;
w44- y44 >= 0;
w45- y45 >= 0;
w46- y46 >= 0;

w51- y51 >= 0;
w52- y52 >= 0;
w53- y53 >= 0;
w54- y54 >= 0;
w55- y55 >= 0;
w56- y56 >= 0;

w61- y61 >= 0;
w62- y62 >= 0;

```
w63- y63 >= 0;
w64- y64 >= 0;
w65- y65 >= 0;
w66- y66 >= 0;

w71- y71 >= 0;
w72- y72 >= 0;
w73- y73 >= 0;
w74- y74 >= 0;
w75- y75 >= 0;
w76- y76 >= 0;

w81- y81 >= 0;
w82- y82 >= 0;
w83- y83 >= 0;
w84- y84 >= 0;
w85- y85 >= 0;
w86- y86 >= 0;

w81- y81 >= 0;
w82- y82 >= 0;
w83- y83 >= 0;
w84- y84 >= 0;
w85- y85 >= 0;
w86- y86 >= 0;

w91- y91 >= 0;
w92- y92 >= 0;
w93- y93 >= 0;
w94- y94 >= 0;
```

```
w95- y95 >= 0;
w96- y96 >= 0;

w101- y101 >= 0;
w102- y102 >= 0;
w103- y103 >= 0;
w104- y104 >= 0;
w105- y105 >= 0;
w106- y106 >= 0;


w11+ w12+ w13+ w14+ w15+ w16 <=2; //constraints as to how many Operations an
Operator can be assigned to each operation
w21+ w22+ w23+ w24+ w25+ w26 <=2;
w31+ w32+ w33+ w34+ w35+ w36 <=2;
w41+ w42+ w43+ w44+ w45+ w46 <=2;
w51+ w52+ w53+ w54+ w55+ w56 <=2;
w61+ w62+ w63+ w64+ w65+ w66 <=2;
w71+ w72+ w73+ w74+ w75+ w76 <=2;
w81+ w82+ w83+ w84+ w85+ w86 <=2;
w91+ w92+ w93+ w94+ w95+ w96 <=2;
w101+ w102+ w103+ w104+ w105+ w106 <=2;
};
```

DETAILED REULTS FOR PHASE 1 (PRODUCTION RATES)

no. operations	6		
no. operators	10		
	Production Rates (per hour)		
	Phase 1(A): No sharing	Phase 1(B): Sharing without restrictions	Phase 1 (C): Sharing with restrictions
Product 1	153.85	194.81	194.81
Product 2	153.85	214.29	214.29
Product 3	206.90	257.51	257.51
Product 4	157.89	215.83	215.83
Product 5	171.43	220.59	220.59
Product 6	230.77	275.23	275.23
Product 7	157.89	215.05	215.05
Product 8	139.53	181.27	181.27
Product 9	193.55	246.91	246.91
Product 10	155.84	212.01	212.01
Product 11	166.67	223.88	223.88
Product 12	173.91	229.89	229.89
Product 13	139.53	204.78	204.78
Product 14	180.00	212.77	212.77
Product 15	176.47	229.01	229.01

no. operations	6		
no. operators	11		
	Production Rates (per hour)		
	Phase 1(A)	Phase 1(B)	Phase I (C)
Product 1	157.89	214.29	214.29
Product 2	171.43	235.71	235.71
Product 3	230.77	283.26	283.26
Product 4	200.00	237.41	237.41
Product 5	181.82	242.65	242.65
Product 6	240.00	302.75	302.75
Product 7	206.90	236.56	236.56
Product 8	148.15	199.40	199.40
Product 9	200.00	271.60	271.60
Product 10	166.67	233.22	233.22
Product 11	187.50	246.27	246.27
Product 12	176.47	252.87	252.87
Product 13	142.86	225.26	225.26
Product 14	181.82	234.04	234.04
Product 15	181.82	251.91	251.91

no. operations	6		
no. operators	12		
	Production Rates (per hour)		
	Phase 1(A)	Phase 1(B)	Phase I (C)
Product 1	169.01	233.76	233.76
Product 2	200.00	257.14	257.14
Product 3	250.00	309.01	309.01
Product 4	206.90	258.99	258.99
Product 5	184.62	264.71	264.71
Product 6	260.87	330.28	330.28
Product 7	230.77	258.06	258.06
Product 8	155.84	217.52	217.52
Product 9	250.00	296.30	296.30
Product 10	222.22	254.42	254.42
Product 11	230.77	268.66	268.66
Product 12	187.50	275.86	275.86
Product 13	209.30	245.73	245.73
Product 14	193.55	255.32	255.32
Product 15	187.50	274.81	274.81

no. operations	6		
no. operators	13		
	Production Rates (per hour)		
	Phase 1(A)	Phase 1(B)	Phase I (C)
Product 1	181.83	253.25	253.25
Product 2	235.29	278.57	278.57
Product 3	260.87	334.76	334.76
Product 4	222.22	280.58	280.58
Product 5	200.00	286.76	286.76
Product 6	292.68	357.80	357.80
Product 7	236.84	279.57	279.57
Product 8	206.90	235.65	235.65
Product 9	253.52	320.99	320.99
Product 10	222.22	275.62	275.62
Product 11	250.00	291.04	291.04
Product 12	260.87	298.85	298.85
Product 13	230.77	266.21	266.21
Product 14	222.22	276.60	276.60
Product 15	222.22	297.71	297.71

no. operations	6		
no. operators	14		
	Production Rates (per hour)		
	Phase 1(A)	Phase 1(B)	Phase 2
Product 1	219.51	272.73	272.73
Product 2	257.14	300.00	300.00
Product 3	285.71	360.52	360.52
Product 4	236.84	302.16	302.16
Product 5	257.14	308.82	308.82
Product 6	292.68	385.32	385.32
Product 7	240.00	301.08	301.08
Product 8	210.53	253.78	253.78
Product 9	285.71	345.68	345.68
Product 10	233.77	296.82	296.82
Product 11	260.87	313.43	313.43
Product 12	266.67	321.84	321.84
Product 13	244.90	286.69	286.69
Product 14	240.00	297.87	297.87
Product 15	260.87	320.61	320.61

no. operations	6		
no. operators	15		
	Production Rates (per hour)		
	Phase 1(A)	Phase 1(B)	Phase I (C)
Product 1	253.52	292.21	292.21
Product 2	292.68	321.43	321.43
Product 3	333.33	386.27	386.27
Product 4	266.67	323.74	323.74
Product 5	276.92	330.88	330.88
Product 6	320.00	412.84	412.84
Product 7	240.00	322.58	322.58
Product 8	222.22	271.90	271.90
Product 9	285.71	370.37	370.37
Product 10	244.90	318.02	318.02
Product 11	279.07	335.82	335.82
Product 12	281.25	344.83	344.83
Product 13	266.67	307.17	307.17
Product 14	240.00	319.15	319.15
Product 15	281.25	343.51	343.51

no. operations	6		
no. operators	16		
	Production Rates (per hour)		
	Phase 1(A)	Phase 1(B)	Phase I (C)
Product 1	266.67	311.69	311.69
Product 2	300.00	342.86	342.86
Product 3	346.15	412.02	412.02
Product 4	272.73	345.32	345.32
Product 5	324.32	352.94	352.94
Product 6	400.00	440.37	440.37
Product 7	272.73	344.09	344.09
Product 8	233.77	290.03	290.03
Product 9	338.03	395.06	395.06
Product 10	311.69	339.22	339.22
Product 11	285.71	358.21	358.21
Product 12	315.79	367.82	367.82
Product 13	266.67	327.65	327.65
Product 14	290.32	340.43	340.43
Product 15	342.86	366.41	366.41

no. operations	6		
no. operators	17		
	Production Rates (per hour)		
	Phase 1(A)	Phase 1(B)	Phase I (C)
Product 1	292.68	331.17	331.17
Product 2	307.69	364.29	364.29
Product 3	375.00	437.77	437.77
Product 4	315.79	366.91	366.91
Product 5	324.32	375.00	375.00
Product 6	439.02	467.89	467.89
Product 7	310.35	365.59	365.59
Product 8	272.73	308.16	308.16
Product 9	375.00	419.75	419.75
Product 10	333.33	360.42	360.42
Product 11	312.50	380.60	380.60
Product 12	347.83	390.80	390.80
Product 13	279.07	348.12	348.12
Product 14	300.00	361.70	361.70
Product 15	347.83	389.31	389.31

no. operations	6		
no. operators	18		
	Production Rates (per hour)		
	Phase 1(A)	Phase 1(B)	Phase I (C)
Product 1	307.69	350.65	350.65
Product 2	315.79	385.71	385.71
Product 3	380.95	463.52	463.52
Product 4	333.33	388.49	388.49
Product 5	342.86	397.06	397.06
Product 6	439.02	495.41	495.41
Product 7	315.79	387.10	387.10
Product 8	279.07	326.28	326.28
Product 9	387.10	444.44	444.44
Product 10	333.33	381.63	381.63
Product 11	333.33	402.99	402.99
Product 12	352.94	413.79	413.79
Product 13	285.71	368.60	368.60
Product 14	342.86	382.98	382.98
Product 15	352.94	412.21	412.21

no. operations	6		
no. operators	19		
	Production Rates (per hour)		
	Phase 1(A)	Phase 1(B)	Phase I (C)
Product 1	315.79	370.13	370.13
Product 2	342.86	407.14	407.14
Product 3	413.79	489.27	489.27
Product 4	394.74	410.07	410.07
Product 5	363.64	419.12	419.12
Product 6	461.54	522.94	522.94
Product 7	360.00	408.60	408.60
Product 8	296.30	344.41	344.41
Product 9	400.00	344.41	344.41
Product 10	333.33	402.83	402.83
Product 11	375.00	425.37	425.37
Product 12	375.00	436.78	436.78
Product 13	348.84	389.08	389.08
Product 14	360.00	404.26	404.26
Product 15	363.64	435.11	435.11

no. operations	6		
no. operators	20		
	Production Rates (per hour)		
	Phase 1(A)	Phase 1(B)	Phase I (C)
Product 1	338.03	389.61	389.61
Product 2	352.94	428.57	428.57
Product 3	461.54	515.02	515.02
Product 4	400.00	431.65	431.65
Product 5	369.23	441.18	441.18
Product 6	480.00	550.46	550.46
Product 7	394.74	430.11	430.11
Product 8	311.69	362.54	362.54
Product 9	422.54	493.83	493.83
Product 10	367.35	424.03	424.03
Product 11	391.30	447.76	447.76
Product 12	400.00	459.77	459.77
Product 13	367.35	409.56	409.56
Product 14	363.64	425.53	425.53
Product 15	375.00	458.02	458.02

Cell Loading (Basic Model) - 8 products, 3 cells and 2 options for Manpower Levels

```
// Cell Loading Models (Crew Size)
// Defining the variables in the order of product, sequence (in cell) and the Cell number, options for Manpower levels
// Overall: 8 products, 3 cells and 2 options for Manpower Levels

// Product 1, all 8 sequences, 3 cells, and 2 options
var int+ X1111 in 0..maxint;
var int+ X1112 in 0..maxint;

var int+ X1121 in 0..maxint;
var int+ X1122 in 0..maxint;

var int+ X1131 in 0..maxint;
var int+ X1132 in 0..maxint;

var int+ X1211 in 0..maxint;
var int+ X1212 in 0..maxint;

var int+ X1221 in 0..maxint;
var int+ X1222 in 0..maxint;

var int+ X1231 in 0..maxint;
var int+ X1232 in 0..maxint;

var int+ X1311 in 0..maxint;
var int+ X1312 in 0..maxint;

var int+ X1321 in 0..maxint;
var int+ X1322 in 0..maxint;

var int+ X1331 in 0..maxint;
var int+ X1332 in 0..maxint;

var int+ X1411 in 0..maxint;
var int+ X1412 in 0..maxint;

var int+ X1421 in 0..maxint;
var int+ X1422 in 0..maxint;

var int+ X1431 in 0..maxint;
var int+ X1432 in 0..maxint;
```

```
var int+ X1511 in 0..maxint;
var int+ X1512 in 0..maxint;

var int+ X1521 in 0..maxint;
var int+ X1522 in 0..maxint;


var int+ X1531 in 0..maxint;
var int+ X1532 in 0..maxint;


var int+ X1611 in 0..maxint;
var int+ X1612 in 0..maxint;

var int+ X1621 in 0..maxint;
var int+ X1622 in 0..maxint;

var int+ X1631 in 0..maxint;
var int+ X1632 in 0..maxint;

var int+ X1711 in 0..maxint;
var int+ X1712 in 0..maxint;

var int+ X1721 in 0..maxint;
var int+ X1722 in 0..maxint;

var int+ X1731 in 0..maxint;
var int+ X1732 in 0..maxint;

var int+ X1811 in 0..maxint;
var int+ X1812 in 0..maxint;

var int+ X1821 in 0..maxint;
var int+ X1822 in 0..maxint;

var int+ X1831 in 0..maxint;
var int+ X1832 in 0..maxint;


// Product 2, all 8 sequences, 3 cells, and 2 options

var int+ X2111 in 0..maxint;
var int+ X2112 in 0..maxint;
```

```
var int+ X2121 in 0..maxint;
var int+ X2122 in 0..maxint;

var int+ X2131 in 0..maxint;
var int+ X2132 in 0..maxint;

var int+ X2211 in 0..maxint;
var int+ X2212 in 0..maxint;

var int+ X2221 in 0..maxint;
var int+ X2222 in 0..maxint;

var int+ X2231 in 0..maxint;
var int+ X2232 in 0..maxint;

var int+ X2311 in 0..maxint;
var int+ X2312 in 0..maxint;

var int+ X2321 in 0..maxint;
var int+ X2322 in 0..maxint;

var int+ X2331 in 0..maxint;
var int+ X2332 in 0..maxint;

var int+ X2411 in 0..maxint;
var int+ X2412 in 0..maxint;

var int+ X2421 in 0..maxint;
var int+ X2422 in 0..maxint;

var int+ X2431 in 0..maxint;
var int+ X2432 in 0..maxint;

var int+ X2511 in 0..maxint;
var int+ X2512 in 0..maxint;

var int+ X2521 in 0..maxint;
var int+ X2522 in 0..maxint;


var int+ X2531 in 0..maxint;
var int+ X2532 in 0..maxint;
```

```
var int+ X2611 in 0..maxint;
var int+ X2612 in 0..maxint;

var int+ X2621 in 0..maxint;
var int+ X2622 in 0..maxint;

var int+ X2631 in 0..maxint;
var int+ X2632 in 0..maxint;

var int+ X2711 in 0..maxint;
var int+ X2712 in 0..maxint;

var int+ X2721 in 0..maxint;
var int+ X2722 in 0..maxint;

var int+ X2731 in 0..maxint;
var int+ X2732 in 0..maxint;

var int+ X2811 in 0..maxint;
var int+ X2812 in 0..maxint;

var int+ X2821 in 0..maxint;
var int+ X2822 in 0..maxint;

var int+ X2831 in 0..maxint;
var int+ X2832 in 0..maxint;

// Product 3, all 8 sequences, 3 cells, and 2 options

var int+ X3111 in 0..maxint;
var int+ X3112 in 0..maxint;

var int+ X3121 in 0..maxint;
var int+ X3122 in 0..maxint;

var int+ X3131 in 0..maxint;
var int+ X3132 in 0..maxint;

var int+ X3211 in 0..maxint;
var int+ X3212 in 0..maxint;

var int+ X3221 in 0..maxint;
var int+ X3222 in 0..maxint;
```

```
var int+ X3231 in 0..maxint;
var int+ X3232 in 0..maxint;

var int+ X3311 in 0..maxint;
var int+ X3312 in 0..maxint;

var int+ X3321 in 0..maxint;
var int+ X3322 in 0..maxint;

var int+ X3331 in 0..maxint;
var int+ X3332 in 0..maxint;

var int+ X3411 in 0..maxint;
var int+ X3412 in 0..maxint;

var int+ X3421 in 0..maxint;
var int+ X3422 in 0..maxint;

var int+ X3431 in 0..maxint;
var int+ X3432 in 0..maxint;

var int+ X3511 in 0..maxint;
var int+ X3512 in 0..maxint;

var int+ X3521 in 0..maxint;
var int+ X3522 in 0..maxint;

var int+ X3531 in 0..maxint;
var int+ X3532 in 0..maxint;

var int+ X3611 in 0..maxint;
var int+ X3612 in 0..maxint;

var int+ X3621 in 0..maxint;
var int+ X3622 in 0..maxint;

var int+ X3631 in 0..maxint;
var int+ X3632 in 0..maxint;
var int+ X3711 in 0..maxint;
var int+ X3712 in 0..maxint;

var int+ X3721 in 0..maxint;
var int+ X3722 in 0..maxint;
```

```
var int+ X3731 in 0..maxint;
var int+ X3732 in 0..maxint;

var int+ X3811 in 0..maxint;
var int+ X3812 in 0..maxint;

var int+ X3821 in 0..maxint;
var int+ X3822 in 0..maxint;

var int+ X3831 in 0..maxint;
var int+ X3832 in 0..maxint;

// Product 4, all 8 sequences, 3 cells, and 2 options

var int+ X4111 in 0..maxint;
var int+ X4112 in 0..maxint;

var int+ X4121 in 0..maxint;
var int+ X4122 in 0..maxint;

var int+ X4131 in 0..maxint;
var int+ X4132 in 0..maxint;

var int+ X4211 in 0..maxint;
var int+ X4212 in 0..maxint;

var int+ X4221 in 0..maxint;
var int+ X4222 in 0..maxint;

var int+ X4231 in 0..maxint;
var int+ X4232 in 0..maxint;

var int+ X4311 in 0..maxint;
var int+ X4312 in 0..maxint;

var int+ X4321 in 0..maxint;
var int+ X4322 in 0..maxint;

var int+ X4331 in 0..maxint;
var int+ X4332 in 0..maxint;

var int+ X4411 in 0..maxint;
var int+ X4412 in 0..maxint;
```

```
var int+ X4421 in 0..maxint;
var int+ X4422 in 0..maxint;

var int+ X4431 in 0..maxint;
var int+ X4432 in 0..maxint;

var int+ X4511 in 0..maxint;
var int+ X4512 in 0..maxint;

var int+ X4521 in 0..maxint;
var int+ X4522 in 0..maxint;


var int+ X4531 in 0..maxint;
var int+ X4532 in 0..maxint;

var int+ X4611 in 0..maxint;
var int+ X4612 in 0..maxint;

var int+ X4621 in 0..maxint;
var int+ X4622 in 0..maxint;

var int+ X4631 in 0..maxint;
var int+ X4632 in 0..maxint;

var int+ X4711 in 0..maxint;
var int+ X4712 in 0..maxint;

var int+ X4721 in 0..maxint;
var int+ X4722 in 0..maxint;

var int+ X4731 in 0..maxint;
var int+ X4732 in 0..maxint;

var int+ X4811 in 0..maxint;
var int+ X4812 in 0..maxint;

var int+ X4821 in 0..maxint;
var int+ X4822 in 0..maxint;

var int+ X4831 in 0..maxint;
var int+ X4832 in 0..maxint;
```

```
// Product 5, all 8 sequences, 3 cells, and 2 options

var int+ X5111 in 0..maxint;
var int+ X5112 in 0..maxint;

var int+ X5121 in 0..maxint;
var int+ X5122 in 0..maxint;

var int+ X5131 in 0..maxint;
var int+ X5132 in 0..maxint;

var int+ X5211 in 0..maxint;
var int+ X5212 in 0..maxint;

var int+ X5221 in 0..maxint;
var int+ X5222 in 0..maxint;

var int+ X5231 in 0..maxint;
var int+ X5232 in 0..maxint;

var int+ X5311 in 0..maxint;
var int+ X5312 in 0..maxint;

var int+ X5321 in 0..maxint;
var int+ X5322 in 0..maxint;

var int+ X5331 in 0..maxint;
var int+ X5332 in 0..maxint;

var int+ X5411 in 0..maxint;
var int+ X5412 in 0..maxint;

var int+ X5421 in 0..maxint;
var int+ X5422 in 0..maxint;

var int+ X5431 in 0..maxint;
var int+ X5432 in 0..maxint;

var int+ X5511 in 0..maxint;
var int+ X5512 in 0..maxint;

var int+ X5521 in 0..maxint;
var int+ X5522 in 0..maxint;
```

```
var int+ X5531 in 0..maxint;
var int+ X5532 in 0..maxint;


var int+ X5611 in 0..maxint;
var int+ X5612 in 0..maxint;

var int+ X5621 in 0..maxint;
var int+ X5622 in 0..maxint;

var int+ X5631 in 0..maxint;
var int+ X5632 in 0..maxint;

var int+ X5711 in 0..maxint;
var int+ X5712 in 0..maxint;

var int+ X5721 in 0..maxint;
var int+ X5722 in 0..maxint;

var int+ X5731 in 0..maxint;
var int+ X5732 in 0..maxint;

var int+ X5811 in 0..maxint;
var int+ X5812 in 0..maxint;

var int+ X5821 in 0..maxint;
var int+ X5822 in 0..maxint;

var int+ X5831 in 0..maxint;
var int+ X5832 in 0..maxint;

// Product 6, all 8 sequences, 3 cells, and 2 options

var int+ X6111 in 0..maxint;
var int+ X6112 in 0..maxint;

var int+ X6121 in 0..maxint;
var int+ X6122 in 0..maxint;

var int+ X6131 in 0..maxint;
var int+ X6132 in 0..maxint;

var int+ X6211 in 0..maxint;
```

```
var int+ X6212 in 0..maxint;

var int+ X6221 in 0..maxint;
var int+ X6222 in 0..maxint;

var int+ X6231 in 0..maxint;
var int+ X6232 in 0..maxint;

var int+ X6311 in 0..maxint;
var int+ X6312 in 0..maxint;

var int+ X6321 in 0..maxint;
var int+ X6322 in 0..maxint;

var int+ X6331 in 0..maxint;
var int+ X6332 in 0..maxint;

var int+ X6411 in 0..maxint;
var int+ X6412 in 0..maxint;

var int+ X6421 in 0..maxint;
var int+ X6422 in 0..maxint;

var int+ X6431 in 0..maxint;
var int+ X6432 in 0..maxint;

var int+ X6511 in 0..maxint;
var int+ X6512 in 0..maxint;

var int+ X6521 in 0..maxint;
var int+ X6522 in 0..maxint;


var int+ X6531 in 0..maxint;
var int+ X6532 in 0..maxint;


var int+ X6611 in 0..maxint;
var int+ X6612 in 0..maxint;

var int+ X6621 in 0..maxint;
var int+ X6622 in 0..maxint;

var int+ X6631 in 0..maxint;
```

```
var int+ X6632 in 0..maxint;

var int+ X6711 in 0..maxint;
var int+ X6712 in 0..maxint;

var int+ X6721 in 0..maxint;
var int+ X6722 in 0..maxint;

var int+ X6731 in 0..maxint;
var int+ X6732 in 0..maxint;

var int+ X6811 in 0..maxint;
var int+ X6812 in 0..maxint;

var int+ X6821 in 0..maxint;
var int+ X6822 in 0..maxint;

var int+ X6831 in 0..maxint;
var int+ X6832 in 0..maxint;

// Product 7, all 8 sequences, 3 cells, and 2 options

var int+ X7111 in 0..maxint;
var int+ X7112 in 0..maxint;

var int+ X7121 in 0..maxint;
var int+ X7122 in 0..maxint;

var int+ X7131 in 0..maxint;
var int+ X7132 in 0..maxint;

var int+ X7211 in 0..maxint;
var int+ X7212 in 0..maxint;

var int+ X7221 in 0..maxint;
var int+ X7222 in 0..maxint;

var int+ X7231 in 0..maxint;
var int+ X7232 in 0..maxint;

var int+ X7311 in 0..maxint;
var int+ X7312 in 0..maxint;

var int+ X7321 in 0..maxint;
```

```
var int+ X7322 in 0..maxint;
var int+ X7331 in 0..maxint;
var int+ X7332 in 0..maxint;

var int+ X7411 in 0..maxint;
var int+ X7412 in 0..maxint;

var int+ X7421 in 0..maxint;
var int+ X7422 in 0..maxint;

var int+ X7431 in 0..maxint;
var int+ X7432 in 0..maxint;

var int+ X7511 in 0..maxint;
var int+ X7512 in 0..maxint;

var int+ X7521 in 0..maxint;
var int+ X7522 in 0..maxint;


var int+ X7531 in 0..maxint;
var int+ X7532 in 0..maxint;

var int+ X7611 in 0..maxint;
var int+ X7612 in 0..maxint;

var int+ X7621 in 0..maxint;
var int+ X7622 in 0..maxint;

var int+ X7631 in 0..maxint;
var int+ X7632 in 0..maxint;

var int+ X7711 in 0..maxint;
var int+ X7712 in 0..maxint;

var int+ X7721 in 0..maxint;
var int+ X7722 in 0..maxint;

var int+ X7731 in 0..maxint;
var int+ X7732 in 0..maxint;

var int+ X7811 in 0..maxint;
var int+ X7812 in 0..maxint;
```

```
var int+ X7821 in 0..maxint;
var int+ X7822 in 0..maxint;

var int+ X7831 in 0..maxint;
var int+ X7832 in 0..maxint;

// Product 8, all 8 sequences, 3 cells, and 2 options

var int+ X8111 in 0..maxint;
var int+ X8112 in 0..maxint;

var int+ X8121 in 0..maxint;
var int+ X8122 in 0..maxint;

var int+ X8131 in 0..maxint;
var int+ X8132 in 0..maxint;

var int+ X8211 in 0..maxint;
var int+ X8212 in 0..maxint;

var int+ X8221 in 0..maxint;
var int+ X8222 in 0..maxint;

var int+ X8231 in 0..maxint;
var int+ X8232 in 0..maxint;

var int+ X8311 in 0..maxint;
var int+ X8312 in 0..maxint;

var int+ X8321 in 0..maxint;
var int+ X8322 in 0..maxint;

var int+ X8331 in 0..maxint;
var int+ X8332 in 0..maxint;

var int+ X8411 in 0..maxint;
var int+ X8412 in 0..maxint;

var int+ X8421 in 0..maxint;
var int+ X8422 in 0..maxint;

var int+ X8431 in 0..maxint;
var int+ X8432 in 0..maxint;
```

```
var int+ X8511 in 0..maxint;
var int+ X8512 in 0..maxint;

var int+ X8521 in 0..maxint;
var int+ X8522 in 0..maxint;


var int+ X8531 in 0..maxint;
var int+ X8532 in 0..maxint;

var int+ X8611 in 0..maxint;
var int+ X8612 in 0..maxint;

var int+ X8621 in 0..maxint;
var int+ X8622 in 0..maxint;

var int+ X8631 in 0..maxint;
var int+ X8632 in 0..maxint;

var int+ X8711 in 0..maxint;
var int+ X8712 in 0..maxint;

var int+ X8721 in 0..maxint;
var int+ X8722 in 0..maxint;

var int+ X8731 in 0..maxint;
var int+ X8732 in 0..maxint;

var int+ X8811 in 0..maxint;
var int+ X8812 in 0..maxint;

var int+ X8821 in 0..maxint;
var int+ X8822 in 0..maxint;

var int+ X8831 in 0..maxint;
var int+ X8832 in 0..maxint;

// defining y(i,j) as cell and configuration option 1 or 2
//cell number & option

var int+ y11 in 0..maxint;
var int+ y12 in 0..maxint;

var int+ y21 in 0..maxint;
```

```
var int+ y22 in 0..maxint;

var int+ y31 in 0..maxint;
var int+ y32 in 0..maxint;

// Decision variables for Completion Times, and Total Tardiness
// sequence, cell number and option

var int+ C111 in 0..maxint;
var int+ C112 in 0..maxint;

var int+ C121 in 0..maxint;
var int+ C122 in 0..maxint;

var int+ C131 in 0..maxint;
var int+ C132 in 0..maxint;

var int+ C211 in 0..maxint;
var int+ C212 in 0..maxint;

var int+ C221 in 0..maxint;
var int+ C222 in 0..maxint;

var int+ C231 in 0..maxint;
var int+ C232 in 0..maxint;

var int+ C311 in 0..maxint;
var int+ C312 in 0..maxint;

var int+ C321 in 0..maxint;
var int+ C322 in 0..maxint;

var int+ C331 in 0..maxint;
var int+ C332 in 0..maxint;

var int+ C411 in 0..maxint;
var int+ C412 in 0..maxint;

var int+ C421 in 0..maxint;
var int+ C422 in 0..maxint;

var int+ C431 in 0..maxint;
var int+ C432 in 0..maxint;
```

```
var int+ C511 in 0..maxint;
var int+ C512 in 0..maxint;

var int+ C521 in 0..maxint;
var int+ C522 in 0..maxint;

var int+ C531 in 0..maxint;
var int+ C532 in 0..maxint;

var int+ C611 in 0..maxint;
var int+ C612 in 0..maxint;

var int+ C621 in 0..maxint;
var int+ C622 in 0..maxint;

var int+ C631 in 0..maxint;
var int+ C632 in 0..maxint;

var int+ C711 in 0..maxint;
var int+ C712 in 0..maxint;

var int+ C721 in 0..maxint;
var int+ C722 in 0..maxint;

var int+ C731 in 0..maxint;
var int+ C732 in 0..maxint;

var int+ C811 in 0..maxint;
var int+ C812 in 0..maxint;

var int+ C821 in 0..maxint;
var int+ C822 in 0..maxint;

var int+ C831 in 0..maxint;
var int+ C832 in 0..maxint;

var int+ T111 in 0..maxint;
var int+ T112 in 0..maxint;

var int+ T121 in 0..maxint;
var int+ T122 in 0..maxint;

var int+ T131 in 0..maxint;
var int+ T132 in 0..maxint;
```

```
var int+ T211 in 0..maxint;
var int+ T212 in 0..maxint;

var int+ T221 in 0..maxint;
var int+ T222 in 0..maxint;

var int+ T231 in 0..maxint;
var int+ T232 in 0..maxint;

var int+ T311 in 0..maxint;
var int+ T312 in 0..maxint;

var int+ T321 in 0..maxint;
var int+ T322 in 0..maxint;

var int+ T331 in 0..maxint;
var int+ T332 in 0..maxint;

var int+ T411 in 0..maxint;
var int+ T412 in 0..maxint;

var int+ T421 in 0..maxint;
var int+ T422 in 0..maxint;

var int+ T431 in 0..maxint;
var int+ T432 in 0..maxint;

var int+ T511 in 0..maxint;
var int+ T512 in 0..maxint;

var int+ T521 in 0..maxint;
var int+ T522 in 0..maxint;

var int+ T531 in 0..maxint;
var int+ T532 in 0..maxint;

var int+ T611 in 0..maxint;
var int+ T612 in 0..maxint;

var int+ T621 in 0..maxint;
var int+ T622 in 0..maxint;

var int+ T631 in 0..maxint;
```

```
var int+ T632 in 0..maxint;

var int+ T711 in 0..maxint;
var int+ T712 in 0..maxint;

var int+ T721 in 0..maxint;
var int+ T722 in 0..maxint;

var int+ T731 in 0..maxint;
var int+ T732 in 0..maxint;

var int+ T811 in 0..maxint;
var int+ T812 in 0..maxint;

var int+ T821 in 0..maxint;
var int+ T822 in 0..maxint;

var int+ T831 in 0..maxint;
var int+ T832 in 0..maxint;


var float+ TT in 0..maxint;

minimize TT

subject to


{

TT =
T111+T121+T131+T211+T221+T231+T311+T321+T331+T411+T421+T431+T511+T5
21+T531+T611+T621+T631+
T112+T122+T132+T212+T222+T232+T312+T322+T332+T412+T422+T432+T512+T5
22+T532+T612+T622+T632+T711+T712+T721+T722+T731+T732+T811+T812+T821
+T822+T831+T832;


//Product 1 must be assigned (position 1 of cell1 or position 1 of cell 2, wtc.
X1111+ X1121 + X1131 + X1211 + X1221 + X1231 +  X1311 + X1321 + X1331 +
X1411 + X1421+ X1431+ X1511+ X1521+ X1531+ X1611+ X1621+ X1631+ X1711+
X1721+ X1731+ X1811+ X1821+ X1831+
```

X1112+ X1122 + X1132 + X1212 + X1222 + X1232 + X1312 + X1322 + X1332 +
X1412 + X1422+ X1432+ X1512+ X1522+ X1532+ X1612+ X1622+ X1632+ X1712+
X1722+ X1732+ X1812+ X1822+ X1832 =1;

X1111+X1211+X1311+X1411+X1511+X1611+X1711+X1811 <= y11;
X1112+X1212+X1312+X1412+X1512+X1612+X1712+X1812 <= y12;
X1121+X1221+X1321+X1421+X1521+X1621+X1721+X1821 <= y21;
X1122+X1222+X1322+X1422+X1522+X1622+X1712+X1812 <= y22;
X1131+X1231+X1331+X1431+X1531+X1631+X1731+X1831 <= y31;
X1132+X1232+X1332+X1432+X1532+X1632+X1712+X1812 <= y32;

//Product 2 must be assigned (position 1 of cell1 or position 1 of cell 2, wtc.
X2111+ X2121 + X2131 + X2211 + X2221 + X2231 + X2311 + X2321 + X2331 +
X2411 + X2421+ X2431+ X2511+ X2521+ X2531+ X2611+ X2621+ X2631+ X2711+
X2721+ X2731+ X2811+ X2821+ X2831+
X2112+ X2122 + X2132 + X2212 + X2222 + X2232 + X2312 + X2322 + X2332 +
X2412 + X2422+ X2432+ X2512+ X2522+ X2532+ X2612+ X2622+ X2632+ X2712+
X2722+ X2732+ X2812+ X2822+ X2832 =1;

X2111+X2211+X2311+X2411+X2511+X2611+X2711+X2811 <= y11;
X2112+X2212+X2312+X2412+X2512+X2612+X2712+X2812 <= y12;
X2121+X2221+X2321+X2421+X2521+X2621+X2721+X2821 <= y21;
X2122+X2222+X2322+X2422+X2522+X2622+X2712+X2812 <= y22;
X2131+X2231+X2331+X2431+X2531+X2631+X2731+X2831 <= y31;
X2132+X2232+X2332+X2432+X2532+X2632+X2712+X2812 <= y32;

//Product 3 must be assigned (position 1 of cell1 or position 1 of cell 2, wtc.
X3111+ X3121 + X3131 + X3211 + X3221 + X3231 + X3311 + X3321 + X3331 +
X3411 + X3421+ X3431+ X3511+ X3521+ X3531+ X3611+ X3621+ X3631+ X3711+
X3721+ X3731+ X3811+ X3821+ X3831+
X3112+ X3122 + X3132 + X3212 + X3222 + X3232 + X3312 + X3322 + X3332 +
X3412 + X3422+ X3432+ X3512+ X3522+ X3532+ X3612+ X3622+ X3632+ X3712+
X3722+ X3732+ X3812+ X3822+ X3832 =1;

X3111+X3211+X3311+X3411+X3511+X3611+X3711+X3811 <= y11;
X3112+X3212+X3312+X3412+X3512+X3612+X3712+X3812 <= y12;
X3121+X3221+X3321+X3421+X3521+X3621+X3721+X3821 <= y21;
X3122+X3222+X3322+X3422+X3522+X3622+X3712+X3812 <= y22;
X3131+X3231+X3331+X3431+X3531+X3631+X3731+X3831 <= y31;
X3132+X3232+X3332+X3432+X3532+X3632+X3712+X3812 <= y32;

//Product 4 must be assigned (position 1 of cell1 or position 1 of cell 2, wtc.

X4111+ X4121 + X4131 + X4211 + X4221 + X4231 + X4311 + X4321 + X4331 +
X4411 + X4421+ X4431+ X4511+ X4521+ X4531+ X4611+ X4621+ X4631+ X4711+
X4721+ X4731+ X4811+ X4821+ X4831+
X4112+ X4122 + X4132 + X4212 + X4222 + X4232 + X4312 + X4322 + X4332 +
X4412 + X4422+ X4432+ X4512+ X4522+ X4532+ X4612+ X4622+ X4632+ X4712+
X4722+ X4732+ X4812+ X4822+ X4832 =1;

X4111+X4211+X4311+X4411+X4511+X4611+X4711+X4811 <= y11;
X4112+X4212+X4312+X4412+X4512+X4612+X4712+X4812 <= y12;
X4121+X4221+X4321+X4421+X4521+X4621+X4721+X4821 <= y21;
X4122+X4222+X4322+X4422+X4522+X4622+X4712+X4812 <= y22;
X4131+X4231+X4331+X4431+X4531+X4631+X4731+X4831 <= y31;
X4132+X4232+X4332+X4432+X4532+X4632+X4712+X4812 <= y32;

//Product 5 must be assigned (position 1 of cell1 or position 1 of cell 2, wtc.
X5111+ X5121 + X5131 + X5211 + X5221 + X5231 + X5311 + X5321 + X5331 +
X5411 + X5421+ X5431+ X5511+ X5521+ X5531+ X5611+ X5621+ X5631+ X5711+
X5721+ X5731+ X5811+ X5821+ X5831+
X5112+ X5122 + X5132 + X5212 + X5222 + X5232 + X5312 + X5322 + X5332 +
X5412 + X5422+ X5432+ X5512+ X5522+ X5532+ X5612+ X5622+ X5632+ X5712+
X5722+ X5732+ X5812+ X5822+ X5832 =1;

X5111+X5211+X5311+X5411+X5511+X5611+X5711+X5811 <= y11;
X5112+X5212+X5312+X5412+X5512+X5612+X5712+X5812 <= y12;
X5121+X5221+X5321+X5421+X5521+X5621+X5721+X5821 <= y21;
X5122+X5222+X5322+X5422+X5522+X5622+X5712+X5812 <= y22;
X5131+X5231+X5331+X5431+X5531+X5631+X5731+X5831 <= y31;
X5132+X5232+X5332+X5432+X5532+X5632+X5712+X5812 <= y32;

//Product 6 must be assigned (position 1 of cell1 or position 1 of cell 2, wtc.
X6111+ X6121 + X6131 + X6211 + X6221 + X6231 + X6311 + X6321 + X6331 +
X6411 + X6421+ X6431+ X6511+ X6521+ X6531+ X6611+ X6621+ X6631+ X6711+
X6721+ X6731+ X6811+ X6821+ X6831+
X6112+ X6122 + X6132 + X6212 + X6222 + X6232 + X6312 + X6322 + X6332 +
X6412 + X6422+ X6432+ X6512+ X6622+ X6532+ X6612+ X6622+ X6632+ X6712+
X6722+ X6732+ X6812+ X6822+ X6832 =1;

X6111+X6211+X6311+X6411+X6511+X6611+X6711+X6811 <= y11;
X6112+X6212+X6312+X6412+X6512+X6612+X6712+X6812 <= y12;
X6121+X6221+X6321+X6421+X6521+X6621+X6721+X6821 <= y21;
X6122+X6222+X6322+X6422+X6522+X6622+X6712+X6812 <= y22;
X6131+X6231+X6331+X6431+X6531+X6631+X6731+X6831 <= y31;
X6132+X6232+X6332+X6432+X6532+X6632+X6712+X6812 <= y32;

//Product 7 must be assigned (position 1 of cell1 or position 1 of cell 2, wtc.
X7111+ X7121 + X7131 + X7211 + X7221 + X7231 + X7311 + X7321 + X7331 +
X7411 + X7421+ X7431+ X7511+ X7521+ X7531+ X7611+ X7621+ X7631+ X7711+
X7721+ X7731+ X7811+ X7821+ X7831+
X7112+ X7122 + X7132 + X7212 + X7222 + X7232 + X7312 + X7322 + X7332 +
X7412 + X7422+ X7432+ X7512+ X7622+ X7532+ X7612+ X7622+ X7632+ X7712+
X7722+ X7732+ X7812+ X7822+ X7832 =1;

X7111+X7211+X7311+X7411+X7511+X7611+X7711+X7811 <= y11;
X7112+X7212+X7312+X7412+X7512+X7612+X7712+X7812 <= y12;
X7121+X7221+X7321+X7421+X7521+X7621+X7721+X7821 <= y21;
X7122+X7222+X7322+X7422+X7522+X7622+X7712+X7812 <= y22;
X7131+X7231+X7331+X7431+X7531+X7631+X7731+X7831 <= y31;
X7132+X7232+X7332+X7432+X7532+X7632+X7712+X7812 <= y32;

//Product 8 must be assigned (position 1 of cell1 or position 1 of cell 2, wtc.
X8111+ X8121 + X8131 + X8211 + X8221 + X8231 + X8311 + X8321 + X8331 +
X8411 + X8421+ X8431+ X8511+ X8521+ X8531+ X8611+ X8621+ X8631+ X8711+
X8721+ X8731+ X8811+ X8821+ X8831+
X8112+ X8122 + X8132 + X8212 + X8222 + X8232 + X8312 + X8322 + X8332 +
X8412 + X8422+ X8432+ X8512+ X8622+ X8532+ X8612+ X8622+ X8632+ X8712+
X8722+ X8732+ X8812+ X8822+ X8832 =1;

X8111+X8211+X8311+X8411+X8511+X8611+X8711+X8811 <= y11;
X8112+X8212+X8312+X8412+X8512+X8612+X8712+X8812 <= y12;
X8121+X8221+X8321+X8421+X8521+X8621+X8721+X8821 <= y21;
X8122+X8222+X8322+X8422+X8522+X8622+X8712+X8812 <= y22;
X8131+X8231+X8331+X8431+X8531+X8631+X8731+X8831 <= y31;
X8132+X8232+X8332+X8432+X8532+X8632+X8712+X8812 <= y32;

//At most one job should be assigned to 1st position in cell 1,..
X1111+ X2111 + X3111 + X4111 + X5111 + X6111 + X7111+ X8111<=y11;
X1112+ X2112 + X3112 + X4112 + X5112 + X6112 + X7112+ X8112<=y12;

//At most one job should be assigned to 1st position in cell 2,..
X1121+ X2121+ X3121+ X4121+ X5121+ X6121+ X7121+ X8121<=y21;
X1122+ X2122+ X3122+ X4122+ X5122+ X6122+ X7122+ X8122<=y22 ;

//At most one job should be assigned to 1st position in cell 3,..
X1131 + X2131 + X3131 + X4131 + X5131 + X6131 + X7131+ X8131<=y31;
X1132 + X2132 + X3132 + X4132 + X5132 + X6132 + X7132+ X8132<=y32;

```
//At most one job should be assigned to 2nd position in cell 1,..
X1211+ X2211 + X3211 + X4211 + X5211 + X6211+ X7211+ X8211<=y11;
X1212+ X2212 + X3212 + X4212 + X5212 + X6212+ X7212+ X8212<=y12;

//At most one job should be assigned to 1st position in cell 2,..
X1211+ X2221+ X3221+ X4221+ X5221+ X6221+ X7221+ X8221<=y21;
X1222+ X2222+ X3222+ X4222+ X5222+ X6222+ X7222+ X8222<=y22 ;

//At most one job should be assigned to 1st position in cell 3,..
X1231 + X2231 + X3231 + X4231 + X5231 + X6231+ X7231+ X8231 <=y31;
X1232 + X2232 + X3232 + X4232 + X5232 + X6232+ X7232+ X8232 <=y32;

//At most one job should be assigned to 1st position in cell 1,..
X1311+ X2311 + X3311 + X4311 + X5311 + X6311+ X7331+ X8331<=y11;
X1312+ X2312 + X3312 + X4312 + X5312 + X6312+ X7332+ X8332<=y12;

//At most one job should be assigned to 1st position in cell 2,..
X1311+ X2321+ X3321+ X4321+ X5321+ X6321+ X7321+ X8321<=y21;
X1322+ X2322+ X3322+ X4322+ X5322+ X6322+ X7322+ X8322<=y22 ;

//At most one job should be assigned to 1st position in cell 3,..
X1331 + X2331 + X3331 + X4331 + X5331 + X6331+ X7331+ X8331 <=y31;
X1332 + X2332 + X3332 + X4332 + X5332 + X6332+ X7332+ X8332 <=y32;

//At most one job should be assigned to 1st position in cell 1,..
X1421+ X2411 + X3411 + X4411 + X5411 + X6411+ X7411+ X8411<=y11;
X1412+ X2412 + X3412 + X4412 + X5412 + X6412+ X7412+ X8412<=y12;

//At most one job should be assigned to 1st position in cell 2,..
X1411+ X2421+ X3421+ X4421+ X5421+ X6421+ X7421+ X8421<=y21;
X1422+ X2422+ X3422+ X4422+ X5422+ X6422+ X7422+ X8422<=y22 ;

//At most one job should be assigned to 1st position in cell 3,..
X1431 + X2431 + X3431 + X4431 + X5431 + X6431 + X7431+ X8431 <=y31;
X1432 + X2432 + X3432 + X4432 + X5432 + X6432 + X7432+ X8432<=y32;


//At most one job should be assigned to 1st position in cell 1,..
X1521+ X2511 + X3511 + X4511 + X5511 + X6511+ X7511+ X8511<=y11;
X1512+ X2512 + X3512 + X4512 + X5512 + X6512+ X7512+ X8512<=y12;

//At most one job should be assigned to 1st position in cell 2,..
X1511+ X2521+ X3521+ X4521+ X5521+ X6521+ X7521+ X8521<=y21;
X1522+ X2522+ X3522+ X4522+ X5522+ X6522+ X7522+ X8522<=y22 ;
```

```
//At most one job should be assigned to 1st position in cell 3,..
X1531 + X2531 + X3531 + X4531 + X5531 + X6531+ X7531+ X8531 <=y31;
X1532 + X2532 + X3532 + X4532 + X5532 + X6532+ X7532+ X8532 <=y32;

//At most one job should be assigned to 1st position in cell 1,..
X1621+ X2611 + X3611 + X4611 + X5611 + X6611+ X7611+ X8611<=y11;
X1612+ X2612 + X3612 + X4612 + X5612 + X6612+ X7612+ X8612<=y12;

//At most one job should be assigned to 1st position in cell 2,..
X1611+ X2621+ X3621+ X4621+ X5621+ X6621+ X7621+ X8621<=y21;
X1622+ X2622+ X3622+ X4622+ X5622+ X6622+ X7622+ X8622<=y22 ;

//At most one job should be assigned to 1st position in cell 3,..
X1631 + X2631 + X3631 + X4631 + X5631 + X6631+ X7631+ X8631 <=y31;
X1632 + X2632 + X3632 + X4632 + X5632 + X6632+ X7632+ X8632 <=y32;


//Completion time of 1st job in cell 1
C111=6*X1111+5*X2111+4*X3111+9*X4111+8*X5111+7*X6111+8*X7111+7*X811
1;
C112=5*X1112+4*X2112+3*X3112+8*X4112+7*X5112+6*X6112+8*X7112+7*X811
2;

//Completion time of 1st job in cell 2
C121=6*X1121+5*X2121+4*X3121+9*X4121+8*X5121+7*X6121+8*X7121+7*X812
1;
C122=6*X1122+5*X2122+4*X3122+9*X4122+8*X5122+7*X6122+8*X7122+7*X812
2;

//Completion time of 1st job in cell 3
C131=6*X1131+5*X2131+4*X3131+9*X4131+8*X5131+7*X6131+8*X7131+7*X813
1;
C132=6*X1132+5*X2132+4*X3132+9*X4132+8*X5132+7*X6132+8*X7132+7*X813
2;

//Completion time of 2nd job in cell 1
C211=C111+6*X1211+5*X2211+4*X3211+9*X4211+8*X5211+7*X6211+8*X7211+7
*X8211;
C212=C112+5*X1212+4*X2212+3*X3212+8*X4212+7*X5212+6*X6212+8*X7212+7
*X8212;

//Completion time of 2nd job in cell 2
```

C221=C121+6*X1221+5*X2221+4*X3221+9*X4221+8*X5221+7*X6221+8*X7221+7*X8221;
C222=C122+5*X1222+4*X2222+3*X3222+8*X4222+7*X5222+6*X6222+8*X7222+7*X8222;

//Completion time of 2nd job in cell 3
C231=C131+6*X1231+5*X2231+4*X3231+9*X4231+8*X5231+7*X6231+8*X7231+7*X8231;
C232=C132+5*X1232+4*X2232+3*X3232+8*X4232+7*X5232+6*X6232+8*X7232+7*X8232;

//Completion time of 3rd job in cell 1
C311=C211+6*X1311+5*X2311+4*X3311+9*X4311+8*X5311+7*X6311+8*X7311+7*X8311;
C312=C212+5*X1312+4*X2312+3*X3312+8*X4312+7*X5312+6*X6312+8*X7312+7*X8312;

//Completion time of 3rd job in cell 2
C321=C221+6*X1321+5*X2321+4*X3321+9*X4321+8*X5321+7*X6321+8*X7321+7*X8321;
C322=C222+5*X1322+4*X2322+3*X3322+8*X4322+7*X5322+6*X6322+8*X7322+7*X8322;

//Completion time of 3rd job in cell 3
C331=C231+6*X1331+5*X2331+4*X3331+9*X4331+8*X5331+7*X6331+8*X7331+7*X8331;
C332=C232+5*X1332+4*X2332+3*X3332+8*X4332+7*X5332+6*X6332+8*X7332+7*X8332;

//Completion time of 4th job in cell 1
C411=C311+6*X1411+5*X2411+4*X3411+9*X4411+8*X5411+7*X6411+8*X7411+7*X8411;
C412=C312+5*X1412+4*X2412+3*X3412+8*X4412+7*X5412+6*X6412+8*X7412+7*X8412;

//Completion time of 4th job in cell 2
C421=C321+6*X1421+5*X2421+4*X3421+9*X4421+8*X5421+7*X6421+8*X7421+7*X8421;
C422=C322+5*X1422+4*X2422+3*X3422+8*X4422+7*X5422+6*X6422+8*X7422+7*X8422;

//Completion time of 4th job in cell 3

C431=C331+6*X1431+5*X2431+4*X3431+9*X4431+8*X5431+7*X6431+8*X7431+7*X8431;
C432=C332+5*X1432+4*X2432+3*X3432+8*X4432+7*X5432+6*X6432+8*X7432+7*X8432;

//Completion time of 5th job in cell 1
C511=C411+6*X1511+5*X2511+4*X3511+9*X4511+8*X5511+7*X6511+8*X7511+7*X8511;
C512=C412+5*X1512+4*X2512+3*X3512+8*X4512+7*X5512+6*X6512+8*X7512+7*X8512;

//Completion time of 5th job in cell 2
C521=C421+6*X1521+5*X2521+4*X3521+9*X4521+8*X5521+7*X6521+8*X7521+7*X8521;
C522=C422+5*X1522+4*X2522+3*X3522+8*X4522+7*X5522+6*X6522+8*X7522+7*X8522;

//Completion time of 5th job in cell 3
C531=C431+6*X1531+5*X2531+4*X3531+9*X4531+8*X5531+7*X6531+8*X7531+7*X8531;
C532=C432+5*X1532+4*X2532+3*X3532+8*X4532+7*X5532+6*X6532+8*X7532+7*X8532;

//Completion time of 6th job (it may be product 1, 2, 3...6) in cell 1
C611=C511+6*X1611+5*X2611+4*X3611+9*X4611+8*X5611+7*X6611+8*X7611+7*X8611;
C612=C512+5*X1612+4*X2612+3*X3612+8*X4612+7*X5612+6*X6612+8*X7612+7*X8612;

//Completion time of 6th job in cell 2
C621=C521+6*X1621+5*X2621+4*X3621+9*X4621+8*X5621+7*X6621+8*X7621+7*X8621;
C622=C522+5*X1622+4*X2622+3*X3622+8*X4622+7*X5622+6*X6622+8*X7622+7*X8622;

//Completion time of 6th job in cell 3
C631=C531+6*X1631+5*X2631+4*X3631+9*X4631+8*X5631+7*X6631+8*X7631+7*X8631;
C632=C532+5*X1632+4*X2632+3*X3632+8*X4632+7*X5632+6*X6632+8*X7632+7*X8632;

//Tardiness of 1st job in cell 1

C111-
(8*X1111+10*X2111+7*X3111+12*X4111+16*X5111+12*X6111+18*X7111+20*X8
111)<=T111;
C112-
(8*X1112+10*X2112+7*X3112+12*X4112+16*X5112+12*X6112+18*X7112+20*X8
112)<=T112;

//Tardiness of 1st job in cell 2
C121-
(8*X1121+10*X2121+7*X3121+12*X4121+16*X5121+12*X6121+18*X7121+20*X8
121)<=T121;
C122-
(8*X1122+10*X2122+7*X3122+12*X4122+16*X5122+12*X6122+18*X7122+20*X8
122)<=T122;

//Tardiness of 1st job in cell 3
C131-
(8*X1131+10*X2131+7*X3131+12*X4131+16*X5131+12*X6131+18*X7131+20*X8
131)<=T131;
C132-
(8*X1132+10*X2132+7*X3132+12*X4132+16*X5132+12*X6132+18*X7132+20*X8
132)<=T132;

//Tardiness of 2nd job in cell 1
C211-
(8*X1211+10*X2211+7*X3211+12*X4211+16*X5211+12*X6211+18*X7211+20*X8
211)<=T211;
C212-
(8*X1212+10*X2212+7*X3212+12*X4212+16*X5212+12*X6212+18*X7212+20*X8
212)<=T212;

//Tardiness of 2nd job in cell 2
C221-
(8*X1221+10*X2221+7*X3221+12*X4221+16*X5221+12*X6221+18*X7221+20*X8
221)<=T221;
C222-
(8*X1222+10*X2222+7*X3222+12*X4222+16*X5222+12*X6222+18*X7222+20*X8
222)<=T222;

//Tardiness of 2nd job in cell 3

C231-
(8*X1231+10*X2231+7*X3231+12*X4231+16*X5231+12*X6231+18*X7231+20*X8
231)<=T231;
C232-
(8*X1232+10*X2232+7*X3232+12*X4232+16*X5232+12*X6232+18*X7232+20*X8
232)<=T232;

//Tardiness of 3rd job in cell 1
C311-
(8*X1311+10*X2311+7*X3311+12*X4311+16*X5311+12*X6311+18*X7311+20*X8
311)<=T311;
C312-
(8*X1312+10*X2312+7*X3312+12*X4312+16*X5312+12*X6312+18*X7312+20*X8
312)<=T312;

//Tardiness of 3rd job in cell 2
C321-
(8*X1321+10*X2321+7*X3321+12*X4321+16*X5321+12*X6321+18*X7321+20*X8
321)<=T321;
C322-
(8*X1322+10*X2322+7*X3322+12*X4322+16*X5322+12*X6322+18*X7322+20*X8
322)<=T322;

//Tardiness of 3rd job in cell 3
C331-
(8*X1331+10*X2331+7*X3331+12*X4331+16*X5331+12*X6331+18*X7331+20*X8
331)<=T331;
C332-
(8*X1332+10*X2332+7*X3332+12*X4332+16*X5332+12*X6332+18*X7332+20*X8
332)<=T332;

//Tardiness of 4th job in cell 1
C411-
(8*X1411+10*X2411+7*X3411+12*X4411+16*X5411+12*X6411+18*X7411+20*X8
411)<=T411;
C412-
(8*X1412+10*X2412+7*X3412+12*X4412+16*X5412+12*X6412+18*X7412+20*X8
412)<=T412;

//Tardiness of 4th job in cell 2
C421-
(8*X1421+10*X2421+7*X3421+12*X4421+16*X5421+12*X6421+18*X7421+20*X8
421)<=T421;

C422-
(8*X1422+10*X2422+7*X3422+12*X4422+16*X5422+12*X6422+18*X7422+20*X8
422)<=T422;

//Tardiness of 4th job in cell 3
C431-
(8*X1431+10*X2431+7*X3431+12*X4431+16*X5431+12*X6431+18*X7431+20*X8
431)<=T431;
C432-
(8*X1432+10*X2432+7*X3432+12*X4432+16*X5432+12*X6432+18*X7432+20*X8
432)<=T432;

//Tardiness of 5th job in cell 1
C511-
(8*X1511+10*X2511+7*X3511+12*X4511+16*X5511+12*X6511+18*X7511+20*X8
511)<=T511;
C512-
(8*X1512+10*X2512+7*X3512+12*X4512+16*X5512+12*X6512+18*X7512+20*X8
512)<=T512;

//Tardiness of 5th job in cell 2
C521-
(8*X1521+10*X2521+7*X3521+12*X4521+16*X5521+12*X6521+18*X7521+20*X8
521)<=T521;
C522-
(8*X1522+10*X2522+7*X3522+12*X4522+16*X5522+12*X6522+18*X7522+20*X8
522)<=T522;

//Tardiness of 5th job in cell 3
C531-
(8*X1531+10*X2531+7*X3531+12*X4531+16*X5531+12*X6531+18*X7531+20*X8
531)<=T531;
C532-
(8*X1532+10*X2532+7*X3532+12*X4532+16*X5532+12*X6532+18*X7532+20*X8
532)<=T532;

//Tardiness of 6th job in cell 1
C611-
(8*X1611+10*X2611+7*X3611+12*X4611+16*X5611+12*X6611+18*X7611+20*X8
611)<=T611;

C612-
(8*X1612+10*X2612+7*X3612+12*X4612+16*X5612+12*X6612+18*X7612+20*X8612)<=T612;

//Tardiness of 6th job in cell 2
C621-
(8*X1621+10*X2621+7*X3621+12*X4621+16*X5621+12*X6621+18*X7621+20*X8621)<=T621;
C622-
(8*X1622+10*X2622+7*X3622+12*X4622+16*X5622+12*X6622+18*X7622+20*X8622)<=T622;

//Tardiness of 6th job in cell 3
C631-
(8*X1631+10*X2631+7*X3631+1*X4631+16*X5631+12*X6631+18*X7631+20*X8631)<=T631;
C632-
(8*X1632+10*X2632+7*X3632+1*X4632+16*X5632+12*X6632+18*X7632+20*X8632)<=T632;

y11+y12<=1;
y21+y22<=1;
y31+y32<=1;

9*y11+10*y12+9*y21+10*y22+9*y31+10*y32<=29;

};

// Cell Loading Models (Crew Size)
// Defining the variables in the order of product, sequence (in cell) and the Cell number
var int+ X1111 in 0..maxint;
var int+ X1112 in 0..maxint;

var int+ X1121 in 0..maxint;
var int+ X1122 in 0..maxint;

var int+ X1131 in 0..maxint;
var int+ X1132 in 0..maxint;

```
var int+ X1211 in 0..maxint;
var int+ X1212 in 0..maxint;

var int+ X1221 in 0..maxint;
var int+ X1222 in 0..maxint;

var int+ X1231 in 0..maxint;
var int+ X1232 in 0..maxint;

var int+ X1311 in 0..maxint;
var int+ X1312 in 0..maxint;

var int+ X1321 in 0..maxint;
var int+ X1322 in 0..maxint;

var int+ X1331 in 0..maxint;
var int+ X1332 in 0..maxint;

var int+ X1411 in 0..maxint;
var int+ X1412 in 0..maxint;

var int+ X1421 in 0..maxint;
var int+ X1422 in 0..maxint;

var int+ X1431 in 0..maxint;
var int+ X1432 in 0..maxint;

var int+ X1511 in 0..maxint;
var int+ X1512 in 0..maxint;

var int+ X1521 in 0..maxint;
var int+ X1522 in 0..maxint;


var int+ X1531 in 0..maxint;
var int+ X1532 in 0..maxint;


var int+ X1611 in 0..maxint;
var int+ X1612 in 0..maxint;

var int+ X1621 in 0..maxint;
var int+ X1622 in 0..maxint;
```

```
var int+ X1631 in 0..maxint;
var int+ X1632 in 0..maxint;

var int+ X2111 in 0..maxint;
var int+ X2112 in 0..maxint;

var int+ X2121 in 0..maxint;
var int+ X2122 in 0..maxint;

var int+ X2131 in 0..maxint;
var int+ X2132 in 0..maxint;

var int+ X2211 in 0..maxint;
var int+ X2212 in 0..maxint;

var int+ X2221 in 0..maxint;
var int+ X2222 in 0..maxint;

var int+ X2231 in 0..maxint;
var int+ X2232 in 0..maxint;

var int+ X2311 in 0..maxint;
var int+ X2312 in 0..maxint;

var int+ X2321 in 0..maxint;
var int+ X2322 in 0..maxint;

var int+ X2331 in 0..maxint;
var int+ X2332 in 0..maxint;

var int+ X2411 in 0..maxint;
var int+ X2412 in 0..maxint;

var int+ X2421 in 0..maxint;
var int+ X2422 in 0..maxint;

var int+ X2431 in 0..maxint;
var int+ X2432 in 0..maxint;

var int+ X2511 in 0..maxint;
var int+ X2512 in 0..maxint;

var int+ X2521 in 0..maxint;
var int+ X2522 in 0..maxint;
```

```
var int+ X2531 in 0..maxint;
var int+ X2532 in 0..maxint;


var int+ X2611 in 0..maxint;
var int+ X2612 in 0..maxint;

var int+ X2621 in 0..maxint;
var int+ X2622 in 0..maxint;

var int+ X2631 in 0..maxint;
var int+ X2632 in 0..maxint;

var int+ X3111 in 0..maxint;
var int+ X3112 in 0..maxint;

var int+ X3121 in 0..maxint;
var int+ X3122 in 0..maxint;

var int+ X3131 in 0..maxint;
var int+ X3132 in 0..maxint;

var int+ X3211 in 0..maxint;
var int+ X3212 in 0..maxint;

var int+ X3221 in 0..maxint;
var int+ X3222 in 0..maxint;

var int+ X3231 in 0..maxint;
var int+ X3232 in 0..maxint;

var int+ X3311 in 0..maxint;
var int+ X3312 in 0..maxint;

var int+ X3321 in 0..maxint;
var int+ X3322 in 0..maxint;

var int+ X3331 in 0..maxint;
var int+ X3332 in 0..maxint;

var int+ X3411 in 0..maxint;
var int+ X3412 in 0..maxint;
```

```
var int+ X3421 in 0..maxint;
var int+ X3422 in 0..maxint;

var int+ X3431 in 0..maxint;
var int+ X3432 in 0..maxint;

var int+ X3511 in 0..maxint;
var int+ X3512 in 0..maxint;

var int+ X3521 in 0..maxint;
var int+ X3522 in 0..maxint;


var int+ X3531 in 0..maxint;
var int+ X3532 in 0..maxint;

var int+ X3611 in 0..maxint;
var int+ X3612 in 0..maxint;

var int+ X3621 in 0..maxint;
var int+ X3622 in 0..maxint;


var int+ X3631 in 0..maxint;
var int+ X3632 in 0..maxint;



var int+ X4611 in 0..maxint;
var int+ X4612 in 0..maxint;

var int+ X4621 in 0..maxint;
var int+ X4622 in 0..maxint;

var int+ X4631 in 0..maxint;
var int+ X4632 in 0..maxint;


var int+ X4111 in 0..maxint;
var int+ X4112 in 0..maxint;

var int+ X4121 in 0..maxint;
var int+ X4122 in 0..maxint;
```

```
var int+ X4131 in 0..maxint;
var int+ X4132 in 0..maxint;

var int+ X4211 in 0..maxint;
var int+ X4212 in 0..maxint;

var int+ X4221 in 0..maxint;
var int+ X4222 in 0..maxint;

var int+ X4231 in 0..maxint;
var int+ X4232 in 0..maxint;

var int+ X4311 in 0..maxint;
var int+ X4312 in 0..maxint;

var int+ X4321 in 0..maxint;
var int+ X4322 in 0..maxint;

var int+ X4331 in 0..maxint;
var int+ X4332 in 0..maxint;

var int+ X4411 in 0..maxint;
var int+ X4412 in 0..maxint;

var int+ X4421 in 0..maxint;
var int+ X4422 in 0..maxint;

var int+ X4431 in 0..maxint;
var int+ X4432 in 0..maxint;

var int+ X4511 in 0..maxint;
var int+ X4512 in 0..maxint;

var int+ X4521 in 0..maxint;
var int+ X4522 in 0..maxint;


var int+ X4531 in 0..maxint;
var int+ X4532 in 0..maxint;



var int+ X5111 in 0..maxint;
```

```
var int+ X5112 in 0..maxint;

var int+ X5121 in 0..maxint;
var int+ X5122 in 0..maxint;

var int+ X5131 in 0..maxint;
var int+ X5132 in 0..maxint;

var int+ X5211 in 0..maxint;
var int+ X5212 in 0..maxint;

var int+ X5221 in 0..maxint;
var int+ X5222 in 0..maxint;

var int+ X5231 in 0..maxint;
var int+ X5232 in 0..maxint;

var int+ X5311 in 0..maxint;
var int+ X5312 in 0..maxint;

var int+ X5321 in 0..maxint;
var int+ X5322 in 0..maxint;

var int+ X5331 in 0..maxint;
var int+ X5332 in 0..maxint;

var int+ X5411 in 0..maxint;
var int+ X5412 in 0..maxint;

var int+ X5421 in 0..maxint;
var int+ X5422 in 0..maxint;

var int+ X5431 in 0..maxint;
var int+ X5432 in 0..maxint;

var int+ X5511 in 0..maxint;
var int+ X5512 in 0..maxint;

var int+ X5521 in 0..maxint;
var int+ X5522 in 0..maxint;


var int+ X5531 in 0..maxint;
var int+ X5532 in 0..maxint;
```

```
var int+ X5611 in 0..maxint;
var int+ X5612 in 0..maxint;

var int+ X5621 in 0..maxint;
var int+ X5622 in 0..maxint;

var int+ X5631 in 0..maxint;
var int+ X5632 in 0..maxint;


var int+ X6111 in 0..maxint;
var int+ X6112 in 0..maxint;

var int+ X6121 in 0..maxint;
var int+ X6122 in 0..maxint;

var int+ X6131 in 0..maxint;
var int+ X6132 in 0..maxint;

var int+ X6211 in 0..maxint;
var int+ X6212 in 0..maxint;

var int+ X6221 in 0..maxint;
var int+ X6222 in 0..maxint;

var int+ X6231 in 0..maxint;
var int+ X6232 in 0..maxint;

var int+ X6311 in 0..maxint;
var int+ X6312 in 0..maxint;

var int+ X6321 in 0..maxint;
var int+ X6322 in 0..maxint;

var int+ X6331 in 0..maxint;
var int+ X6332 in 0..maxint;

var int+ X6411 in 0..maxint;
var int+ X6412 in 0..maxint;

var int+ X6421 in 0..maxint;
var int+ X6422 in 0..maxint;
```

```
var int+ X6431 in 0..maxint;
var int+ X6432 in 0..maxint;

var int+ X6511 in 0..maxint;
var int+ X6512 in 0..maxint;

var int+ X6521 in 0..maxint;
var int+ X6522 in 0..maxint;


var int+ X6531 in 0..maxint;
var int+ X6532 in 0..maxint;


var int+ X6611 in 0..maxint;
var int+ X6612 in 0..maxint;

var int+ X6621 in 0..maxint;
var int+ X6622 in 0..maxint;

var int+ X6631 in 0..maxint;
var int+ X6632 in 0..maxint;

var int+ y11 in 0..maxint;
var int+ y12 in 0..maxint;

var int+ y21 in 0..maxint;
var int+ y22 in 0..maxint;

var int+ y31 in 0..maxint;
var int+ y32 in 0..maxint;

// Decision variables for Completion Times, and Total Tardiness

var int+ C111 in 0..maxint;
var int+ C112 in 0..maxint;

var int+ C121 in 0..maxint;
var int+ C122 in 0..maxint;

var int+ C131 in 0..maxint;
var int+ C132 in 0..maxint;
```

```
var int+ C211 in 0..maxint;
var int+ C212 in 0..maxint;

var int+ C221 in 0..maxint;
var int+ C222 in 0..maxint;

var int+ C231 in 0..maxint;
var int+ C232 in 0..maxint;



var int+ C311 in 0..maxint;
var int+ C312 in 0..maxint;

var int+ C321 in 0..maxint;
var int+ C322 in 0..maxint;

var int+ C331 in 0..maxint;
var int+ C332 in 0..maxint;

var int+ C411 in 0..maxint;
var int+ C412 in 0..maxint;

var int+ C421 in 0..maxint;
var int+ C422 in 0..maxint;

var int+ C431 in 0..maxint;
var int+ C432 in 0..maxint;

var int+ C511 in 0..maxint;
var int+ C512 in 0..maxint;

var int+ C521 in 0..maxint;
var int+ C522 in 0..maxint;

var int+ C531 in 0..maxint;
var int+ C532 in 0..maxint;

var int+ C611 in 0..maxint;
var int+ C612 in 0..maxint;

var int+ C621 in 0..maxint;
var int+ C622 in 0..maxint;
```

```
var int+ C631 in 0..maxint;
var int+ C632 in 0..maxint;


var int+ T111 in 0..maxint;
var int+ T112 in 0..maxint;

var int+ T121 in 0..maxint;
var int+ T122 in 0..maxint;

var int+ T131 in 0..maxint;
var int+ T132 in 0..maxint;

var int+ T211 in 0..maxint;
var int+ T212 in 0..maxint;

var int+ T221 in 0..maxint;
var int+ T222 in 0..maxint;

var int+ T231 in 0..maxint;
var int+ T232 in 0..maxint;


var int+ T311 in 0..maxint;
var int+ T312 in 0..maxint;

var int+ T321 in 0..maxint;
var int+ T322 in 0..maxint;

var int+ T331 in 0..maxint;
var int+ T332 in 0..maxint;

var int+ T411 in 0..maxint;
var int+ T412 in 0..maxint;

var int+ T421 in 0..maxint;
var int+ T422 in 0..maxint;

var int+ T431 in 0..maxint;
var int+ T432 in 0..maxint;

var int+ T511 in 0..maxint;
var int+ T512 in 0..maxint;
```

```
var int+ T521 in 0..maxint;
var int+ T522 in 0..maxint;

var int+ T531 in 0..maxint;
var int+ T532 in 0..maxint;

var int+ T611 in 0..maxint;
var int+ T612 in 0..maxint;

var int+ T621 in 0..maxint;
var int+ T622 in 0..maxint;

var int+ T631 in 0..maxint;
var int+ T632 in 0..maxint;


var float+ TT in 0..maxint;

minimize TT

subject to


{

TT =
T111+T121+T131+T211+T221+T231+T311+T321+T331+T411+T421+T431+T511+T5
21+T531+T611+T621+T631+
T112+T122+T132+T212+T222+T232+T312+T322+T332+T412+T422+T432+T512+T5
22+T532+T612+T622+T632;


//Product 1 must be assigned (position 1 of cell1 or position 1 of cell 2, wtc.
X1111+ X1121 + X1131 + X1211 + X1221 + X1231 +  X1311 + X1321 + X1331 +
X1411 + X1421+ X1431+ X1511+ X1521+ X1531+ X1611+ X1621+ X1631+
X1112+ X1122 + X1132 + X1212 + X1222 + X1232 +  X1312 + X1322 + X1332 +
X1412 + X1422+ X1432+ X1512+ X1522+ X1532+ X1612+ X1622+ X1632 =1;

X1111+X1211+X1311+X1411+X1511+X1611 <= y11;
X1112+X1212+X1312+X1412+X1512+X1612 <= y12;
X1121+X1221+X1321+X1421+X1521+X1621 <= y21;
X1122+X1222+X1322+X1422+X1522+X1622 <= y22;
X1131+X1231+X1331+X1431+X1531+X1631 <= y31;
```

```
X1132+X1232+X1332+X1432+X1532+X1632 <= y32;

//Product 2 must be assigned (position 1 of cell1 or position 1 of cell 2, wtc.
X2111 + X2121 + X2131 + X2211 + X2221 + X2231 +  X2311 + X2321 + X2331 +
X2411 + X2421+ X2431+ X2511+ X2521+ X2531+ X2611+ X2621+ X2631+
X2112 + X2122 + X2132 + X2212 + X2222 + X2232 +  X2312 + X2322 + X2332 +
X2412 + X2422+ X2432+ X2512+ X2522+ X2532+ X2612+ X2622+ X2632 =1;

X2111+X2211+X2311+X2411+X2511+X2611 <= y11;
X2112+X2212+X2312+X2412+X2512+X2612 <= y12;
X2121+X2221+X2321+X2421+X2521+X2621 <= y21;
X2122+X2222+X2322+X2422+X2522+X2622 <= y22;
X2131+X2231+X2331+X2431+X2531+X2631 <= y31;
X2132+X2232+X2332+X2432+X2532+X2632 <= y32;

//Product 3 must be assigned (position 1 of cell1 or position 1 of cell 2, wtc.
X3111 + X3121 + X3131 + X3211 + X3221 + X3231 +  X3311 + X3321 + X3331 +
X3411 + X3421+X3431+X3511+X3521+X3531+X3611+X3621+X3631+
X3112 + X3122 + X3132 + X3212 + X3222 + X3232 +  X3312 + X3322 + X3332 +
X3412 + X3422+X3432+X3512+X3522+X3532+X3612+X3622+X3632=1;

X3111+X3211+X3311+X3411+X3511+X3611 <= y11;
X3112+X3212+X3312+X3412+X3512+X3612 <= y12;
X3121+X3221+X3321+X3421+X3521+X3621 <= y21;
X3122+X3222+X3322+X3422+X3522+X3622 <= y22;
X3131+X3231+X3331+X3431+X3531+X3631 <= y31;
X3132+X3232+X3332+X3432+X3532+X3632 <= y32;

//Product 4 must be assigned (position 1 of cell1 or position 1 of cell 2, wtc.
X4111 + X4121 + X4131 + X4211 + X4221 + X4231 +  X4311+ X4321 + X4331 +
X4411 + X4421+X4431+X4511+X4521+X4531+X4611+X4621+X4631+
X4112 + X4122 + X4132 + X4212 + X4222 + X4232 +  X4312 + X4322 + X4332 +
X4412 + X4422+X4432+X4512+X4522+X4532+X4612+X4622+X4632=1;

X4112+X4212+X4312+X4412+X4512+X4612 <= y12;
X4121+X4221+X4321+X4421+X4521+X4621 <= y21;
X4122+X4222+X4322+X4422+X4522+X4622 <= y22;
X4131+X4231+X4331+X4431+X4531+X4631 <= y31;
X4132+X4232+X4332+X4432+X4532+X4632 <= y32;

//Product 5 must be assigned (position 1 of cell1 or position 1 of cell 2, wtc.
X5111 + X5121 + X5131 + X5211 + X5221 + X5231 +  X5311 + X5321+ X5331 +
X5411 + X5421+X5431+X5511+X5521+X5531+X5611+X5621+X5631+
```

X5112 + X5122 + X5132 + X5212 + X5222 + X5232 + X5312 + X5322 + X5332 +
X5412 + X5422+X5432+X5512+X5522+X5532+X5612+X5622+X5632=1;

X5112+X5212+X5312+X5412+X5512+X5612 <= y12;
X5121+X5221+X5321+X5421+X5521+X5621 <= y21;
X5122+X5222+X5322+X5422+X5522+X5622 <= y22;
X5131+X5231+X5331+X5431+X5531+X5631 <= y31;
X5132+X5232+X5332+X5432+X5532+X5632 <= y32;

//Product 6 must be assigned (position 1 of cell1 or position 1 of cell 2, wtc.
X6111 + X6121 + X6131 + X6211 + X6221 + X6231 + X6311+ X6321 + X6331 +
X6411 + X6421+X6431+X6511+X6521+X6531+X6611+X6621+X6631+
X6112 + X6122 + X6132 + X6212 + X6222 + X6232 + X6312+ X6322 + X6332 +
X6412 + X6422+X6432+X6512+X6522+X6532+X6612+X6622+X6632=1;

X6112+X6212+X6312+X6412+X6512+X6612 <= y12;
X6121+X6221+X6321+X6421+X6521+X6621 <= y21;
X6122+X6222+X6322+X6422+X6522+X6622 <= y22;
X6131+X6231+X6331+X6431+X6531+X6631 <= y31;
X6132+X6232+X6332+X6432+X6532+X6632 <= y32;

//At most one job should be assigned to 1st position in cell 1,..
X1111+ X2111 + X3111 + X4111 + X5111 + X6111<=y11;
X1112+ X2112 + X3112 + X4112 + X5112 + X6112<=y12;

//At most one job should be assigned to 1st position in cell 2,..
X1121+ X2121+ X3121+ X4121+ X5121+ X6121<=y21;
X1122+ X2122+ X3122+ X4122+ X5122+ X6122<=y22 ;

//At most one job should be assigned to 1st position in cell 3,..
X1131 + X2131 + X3131 + X4131 + X5131 + X6131 <=y31;
X1132 + X2132 + X3132 + X4132 + X5132 + X6132 <=y32;

//At most one job should be assigned to 1st position in cell 1,..
X1211+ X2211 + X3211 + X4211 + X5211 + X6211<=y11;
X1212+ X2212 + X3212 + X4212 + X5212 + X6212<=y12;

//At most one job should be assigned to 1st position in cell 2,..
X1211+ X2221+ X3221+ X4221+ X5221+ X6221<=y21;
X1222+ X2222+ X3222+ X4222+ X5222+ X6222<=y22 ;

//At most one job should be assigned to 1st position in cell 3,..
X1231 + X2231 + X3231 + X4231 + X5231 + X6231 <=y31;

X1232 + X2232 + X3232 + X4232 + X5232 + X6232 <=y32;

//At most one job should be assigned to 1st position in cell 1,..
X1311+ X2311 + X3311 + X4311 + X5311 + X6311<=y11;
X1312+ X2312 + X3312 + X4312 + X5312 + X6312<=y12;

//At most one job should be assigned to 1st position in cell 2,..
X1311+ X2321+ X3321+ X4321+ X5321+ X6321<=y21;
X1322+ X2322+ X3322+ X4322+ X5322+ X6322<=y22 ;

//At most one job should be assigned to 1st position in cell 3,..
X1331 + X2331 + X3331 + X4331 + X5331 + X6331 <=y31;
X1332 + X2332 + X3332 + X4332 + X5332 + X6332 <=y32;

//At most one job should be assigned to 1st position in cell 1,..
X1421+ X2411 + X3411 + X4411 + X5411 + X6411<=y11;
X1412+ X2412 + X3412 + X4412 + X5412 + X6412<=y12;

//At most one job should be assigned to 1st position in cell 2,..
X1411+ X2421+ X3421+ X4421+ X5421+ X6421<=y21;
X1422+ X2422+ X3422+ X4422+ X5422+ X6422<=y22 ;

//At most one job should be assigned to 1st position in cell 3,..
X1431 + X2431 + X3431 + X4431 + X5431 + X6431 <=y31;
X1432 + X2432 + X3432 + X4432 + X5432 + X6432 <=y32;

//At most one job should be assigned to 1st position in cell 1,..
X1521+ X2511 + X3511 + X4511 + X5511 + X6511<=y11;
X1512+ X2512 + X3512 + X4512 + X5512 + X6512<=y12;

//At most one job should be assigned to 1st position in cell 2,..
X1511+ X2521+ X3521+ X4521+ X5521+ X6521<=y21;
X1522+ X2522+ X3522+ X4522+ X5522+ X6522<=y22 ;

//At most one job should be assigned to 1st position in cell 3,..
X1531 + X2531 + X3531 + X4531 + X5531 + X6531 <=y31;
X1532 + X2532 + X3532 + X4532 + X5532 + X6532 <=y32;

//At most one job should be assigned to 1st position in cell 1,..
X1621+ X2611 + X3611 + X4611 + X5611 + X6611<=y11;
X1612+ X2612 + X3612 + X4612 + X5612 + X6612<=y12;

//At most one job should be assigned to 1st position in cell 2,..

```
X1611+ X2621+ X3621+ X4621+ X5621+ X6621<=y21;
X1622+ X2622+ X3622+ X4622+ X5622+ X6622<=y22 ;

//At most one job should be assigned to 1st position in cell 3,..
X1631 + X2631 + X3631 + X4631 + X5631 + X6631 <=y31;
X1632 + X2632 + X3632 + X4632 + X5632 + X6632 <=y32;


//Completion time of 1st job in cell 1
C111=6*X1111+5*X2111+4*X3111+9*X4111+8*X5111+7*X6111;
C112=5*X1112+4*X2112+3*X3112+8*X4112+7*X5112+6*X6112;

//Completion time of 1st job in cell 2
C121=6*X1121+5*X2121+4*X3121+9*X4121+8*X5121+7*X6121;
C122=6*X1122+5*X2122+4*X3122+9*X4122+8*X5122+7*X6122;

//Completion time of 1st job in cell 3
C131=6*X1131+5*X2131+4*X3131+9*X4131+8*X5131+7*X6131;
C132=6*X1132+5*X2132+4*X3132+9*X4132+8*X5132+7*X6132;

//Completion time of 2nd job in cell 1
C211=C111+6*X1211+5*X2211+4*X3211+9*X4211+8*X5211+7*X6211;
C212=C112+5*X1212+4*X2212+3*X3212+8*X4212+7*X5212+6*X6212;

//Completion time of 2nd job in cell 2
C221=C121+6*X1221+5*X2221+4*X3221+9*X4221+8*X5221+7*X6221;
C222=C122+5*X1222+4*X2222+3*X3222+8*X4222+7*X5222+6*X6222;

//Completion time of 2nd job in cell 3
C231=C131+6*X1231+5*X2231+4*X3231+9*X4231+8*X5231+7*X6231;
C232=C132+5*X1232+4*X2232+3*X3232+8*X4232+7*X5232+6*X6232;

//Completion time of 3rd job in cell 1
C311=C211+6*X1311+5*X2311+4*X3311+9*X4311+8*X5311+7*X6311;
C312=C212+5*X1312+4*X2312+3*X3312+8*X4312+7*X5312+6*X6312;

//Completion time of 3rd job in cell 2
C321=C221+6*X1321+5*X2321+4*X3321+9*X4321+8*X5321+7*X6321;
C322=C222+5*X1322+4*X2322+3*X3322+8*X4322+7*X5322+6*X6322;

//Completion time of 3rd job in cell 3
C331=C231+6*X1331+5*X2331+4*X3331+9*X4331+8*X5331+7*X6331;
C332=C232+5*X1332+4*X2332+3*X3332+8*X4332+7*X5332+6*X6332;
```

```
//Completion time of 4th job in cell 1
C411=C311+6*X1411+5*X2411+4*X3411+9*X4411+8*X5411+7*X6411;
C412=C312+5*X1412+4*X2412+3*X3412+8*X4412+7*X5412+6*X6412;


//Completion time of 4th job in cell 2
C421=C321+6*X1421+5*X2421+4*X3421+9*X4421+8*X5421+7*X6421;
C422=C322+5*X1422+4*X2422+3*X3422+8*X4422+7*X5422+6*X6422;

//Completion time of 4th job in cell 3
C431=C331+6*X1431+5*X2431+4*X3431+9*X4431+8*X5431+7*X6431;
C432=C332+5*X1432+4*X2432+3*X3432+8*X4432+7*X5432+6*X6432;

//Completion time of 5th job in cell 1
C511=C411+6*X1511+5*X2511+4*X3511+9*X4511+8*X5511+7*X6511;
C512=C412+5*X1512+4*X2512+3*X3512+8*X4512+7*X5512+6*X6512;

//Completion time of 5th job in cell 2
C521=C421+6*X1521+5*X2521+4*X3521+9*X4521+8*X5521+7*X6521;
C522=C422+5*X1522+4*X2522+3*X3522+8*X4522+7*X5522+6*X6522;

//Completion time of 5th job in cell 3
C531=C431+6*X1531+5*X2531+4*X3531+9*X4531+8*X5531+7*X6531;
C532=C432+5*X1532+4*X2532+3*X3532+8*X4532+7*X5532+6*X6532;


//Completion time of 6th job (it may be product 1, 2, 3...6) in cell 1
C611=C511+6*X1611+5*X2611+4*X3611+9*X4611+8*X5611+7*X6611;
C612=C512+5*X1612+4*X2612+3*X3612+8*X4612+7*X5612+6*X6612;

//Completion time of 6th job in cell 2
C621=C521+6*X1621+5*X2621+4*X3621+9*X4621+8*X5621+7*X6621;
C622=C522+5*X1622+4*X2622+3*X3622+8*X4622+7*X5622+6*X6622;

//Completion time of 6th job in cell 3
C631=C531+6*X1631+5*X2631+4*X3631+9*X4631+8*X5631+7*X6631;
C632=C532+5*X1632+4*X2632+3*X3632+8*X4632+7*X5632+6*X6632;

//Tardiness of 1st job in cell 1
C111- (8*X1111+10*X2111+7*X3111+12*X4111+16*X5111+12*X6111)<=T111;
C112- (8*X1112+10*X2112+7*X3112+12*X4112+16*X5112+12*X6112)<=T112;

//Tardiness of 1st job in cell 2
C121-(8*X1121+10*X2121+7*X3121+12*X4121+16*X5121+12*X6121)<=T121;
```

```
C122-(8*X1122+10*X2122+7*X3122+12*X4122+16*X5122+12*X6122)<=T122;

//Tardiness of 1st job in cell 3
C131-(8*X1131+10*X2131+7*X3131+12*X4131+16*X5131+12*X6131)<=T131;
C132-(8*X1132+10*X2132+7*X3132+12*X4132+16*X5132+12*X6132)<=T132;

//Tardiness of 2nd job in cell 1
C211- (8*X1211+10*X2211+7*X3211+12*X4211+16*X5211+12*X6211)<=T211;
C212- (8*X1212+10*X2212+7*X3212+12*X4212+16*X5212+12*X6212)<=T212;


//Tardiness of 2nd job in cell 2
C221-(8*X1221+10*X2221+7*X3221+12*X4221+16*X5221+12*X6221)<=T221;
C222-(8*X1222+10*X2222+7*X3222+12*X4222+16*X5222+12*X6222)<=T222;

//Tardiness of 2nd job in cell 3
C231-(8*X1231+10*X2231+7*X3231+12*X4231+16*X5231+12*X6231)<=T231;
C232-(8*X1232+10*X2232+7*X3232+12*X4232+16*X5232+12*X6232)<=T232;

//Tardiness of 3rd job in cell 1
C311- (8*X1311+10*X2311+7*X3311+12*X4311+16*X5311+12*X6311)<=T311;
C312- (8*X1312+10*X2312+7*X3312+12*X4312+16*X5312+12*X6312)<=T312;

//Tardiness of 3rd job in cell 2
C321-(8*X1321+10*X2321+7*X3321+12*X4321+16*X5321+12*X6321)<=T321;
C322-(8*X1322+10*X2322+7*X3322+12*X4322+16*X5322+12*X6322)<=T322;

//Tardiness of 3rd job in cell 3
C331-(8*X1331+10*X2331+7*X3331+12*X4331+16*X5331+12*X6331)<=T331;
C332-(8*X1332+10*X2332+7*X3332+12*X4332+16*X5332+12*X6332)<=T332;


//Tardiness of 4th job in cell 1
C411- (8*X1411+10*X2411+7*X3411+12*X4411+16*X5411+12*X6411)<=T411;
C412- (8*X1412+10*X2412+7*X3412+12*X4412+16*X5412+12*X6412)<=T412;

//Tardiness of 4th job in cell 2
C421-(8*X1421+10*X2421+7*X3421+12*X4421+16*X5421+12*X6421)<=T421;
C422-(8*X1422+10*X2422+7*X3422+12*X4422+16*X5422+12*X6422)<=T422;

//Tardiness of 4th job in cell 3
C431-(8*X1431+10*X2431+7*X3431+12*X4431+16*X5431+12*X6431)<=T431;
C432-(8*X1432+10*X2432+7*X3432+12*X4432+16*X5432+12*X6432)<=T432;
```

```
//Tardiness of 5th job in cell 1
C511- (8*X1511+10*X2511+7*X3511+12*X4511+16*X5511+12*X6511)<=T511;
C512- (8*X1512+10*X2512+7*X3512+12*X4512+16*X5512+12*X6512)<=T512;

//Tardiness of 5th job in cell 2
C521-(8*X1521+10*X2521+7*X3521+12*X4521+16*X5521+12*X6521)<=T521;
C522-(8*X1522+10*X2522+7*X3522+12*X4522+16*X5522+12*X6522)<=T522;


//Tardiness of 5th job in cell 3
C531-(8*X1531+10*X2531+7*X3531+12*X4531+16*X5531+12*X6531)<=T531;
C532-(8*X1532+10*X2532+7*X3532+12*X4532+16*X5532+12*X6532)<=T532;


//Tardiness of 6th job in cell 1
C611- (8*X1611+10*X2611+7*X3611+12*X4611+16*X5611+12*X6611)<=T611;
C612- (8*X1612+10*X2612+7*X3612+12*X4612+16*X5612+12*X6612)<=T612;


//Tardiness of 6th job in cell 2
C621-(8*X1621+10*X2621+7*X3621+12*X4621+16*X5621+12*X6621)<=T621;
C622-(8*X1622+10*X2622+7*X3622+12*X4622+16*X5622+12*X6622)<=T622;

//Tardiness of 6th job in cell 3
C631-(8*X1631+10*X2631+7*X3631+1*X4631+16*X5631+12*X6631)<=T631;
C632-(8*X1632+10*X2632+7*X3632+1*X4632+16*X5632+12*X6632)<=T632;


y11+y12<=1;
y21+y22<=1;
y31+y32<=1;

9*y11+10*y12+9*y21+10*y22+9*y31+10*y32<=29;

};
```

```
/*********************************************
 * OPL 4.2 Model (Phase I a Model)
 * Author: kk142504
 * Creation Date: Sat May 26 12:21:28 2007
 *********************************************/

// operations i.e. operator to operation of y(kj) is defined as an array

int nbOperation = ...;
int nbOperator = ...;

range Operation = 1..nbOperation;
range Operator = 1..nbOperator;

float operation_times_dataset[Operation] = ...;

dvar int+ y[Operator] [Operation] in 0..1;


// Defining Z as the value of R where R is the production rate
dvar float+ z in 0..maxint;
dvar float+ R in 0..maxint;

maximize z; //objective function to find the maximum production rate

subject to

{

z == R;

//constraint 1 computing prod rate on which operator-operation assignment is chosen
  forall (j in Operation)
  {
    operation_times_dataset[j]*(sum(k in Operator)(y[k][j]))-R>=0;

  }

//constraint i.e. every operator is assigned only once to a single operation

  forall (a in Operator)
  {
    sum(b in Operation)(y[a][b]))<=1;
```

```
  }

}
/*******************************************
 * OPL 4.2 Model (Phase I b Model)
 * Author: kk142504
 * Creation Date: Sat May 26 12:21:28 2007
 *******************************************/

// operations i.e. operator to operation of y(kj) is defined as an array

int nbOperation = ...;
int nbOperator = ...;

range Operation = 1..nbOperation;
range Operator = 1..nbOperator;

float operation_times_dataset[Operation] = ...;

dvar float+ y[Operator] [Operation] in 0..1;


// Defining Z as the value of R where R is the production rate
dvar float+ z in 0..maxint;
dvar float+ R in 0..maxint;

maximize z; //objective function to find the maximum production rate

subject to

{

z == R;

//constraint 1
  forall (j in Operation)
  {
    operation_times_dataset[j]*(sum(k in Operator)(y[k][j]))-R>=0;

  }

//constraint 2

  forall (a in Operator)
```

```
  {
    sum(b in Operation)(y[a][b]))<=1;
  }

}
/*********************************************
 * OPL 4.2 Model (Phase I c Model)
 * Author: kk142504
 * Creation Date: Sat May 26 12:21:28 2007
 *********************************************/
// operations i.e. operator to operation of y(kj) is defined as an array

int nbOperation = ...;
int nbOperator = ...;
int nbCell= ...;

range Operation = 1..nbOperation;
range Operator = 1..nbOperator;
range Cell  = 1..nbCell;

float operation_times_dataset[Operation] = ...;


dvar float+ y[Operator] [Operation] in 0..1;
dvar int+ w[Cell] [Operation] in 0..1;



// Defining Z as the value of R where R is the production rate
dvar float+ z in 0..maxint;
dvar float+ R in 0..maxint;

maximize z; //objective function to find the maximum production rate

subject to

{

z == R;

//constraint 1
  forall (j in Operation)
  {
   (60/operation_times_dataset[j])*(sum(k in Operator)(y[k][j]))-R>=0;
```

```
  }

//constraint 2

  forall (a in Operator)
  {
    sum(b in Operation)(y[a][b])<=1;
  }



//constraint 3 i.e. maximum number of operations per operator

  forall (a in Cell)
  {
   sum(b in Operation)(w[a][b])<=2;
  }



//constraint 4 i.e. linking the integer value of w to the real value of y
  forall (a in Cell)
  {
   forall (b in Operation)
       {
           (w[a][b])-(y[a][b])>=0;
        }

}
}
```

DATASET

```
nbOperation = 6;
nbOperator = 19;
nbCell=3;
operation_times_dataset =[0.45, 0.38, 0.33, 0.82, 0.39, 0.71];
```

```
/*********************************************
 * OPL 4.2 Model (Phase II Model)
 * Author: kk142504
 * Creation Date: Sat May 26 12:21:28 2007
 *********************************************/

int nbProduct=…;
int nbSequence=…;
int nbCell=…;
int nbOption=…;

range Product=1.. nbProduct;
range Sequence=1.. nbSequence;
range Cell=1.. nbCell;
range Option=1.. nbOption;

range newSequence=2.. nbSequence;



float ProcessingTime[Product] = ...;
int Crewsize=…;
int Operator[Option]=…;

dvar  int+ X[Product][Sequence][Cell][Option] in 0..maxint;
dvar int+ y[Cell][Option] in 0..maxint;
dvar int+ C[Sequence][Cell][Option] in 0..maxint;
dvar int+ T[Sequence][Cell][Option] in 0..maxint;

dvar float+ TT in 0..maxint;

minimize TT;

subject to

{
  //objective
  TT== sum( i in Sequence) sum( j in Cell)sum( K in Option)(T[i][j][k]);

  //constraint 1 i.e. single assignment for each sequence, cell and option combination
  forall(a in Product)
   {
       sum( i in Sequence) sum( j in Cell)sum( K in Option)(X[a][i][j][k])==1;
   }
```

```
//constraint 2
forall(a in Product)
  {
     forall(b in Cell)
            {
                forall(c in Option)
                 {
                           sum( d in Sequence) X[a][d][b][c])<=y[b][c];

                   }
            }

  }


//constraint 3
  forall(e in Sequence)
  {
        forall(f in Cell)
        {
            forall(g in Option)
             {
                  sum( h in Product) X[h][e][f][g])<=y[f][g];

             }

        }

  }
//constraint 4
        forall(u in Cell)
        {
            forall(v in Option)
             {
                  C[1][u][v]==sum( w in Product)( ProcessingTime[w]*X[w][1][u][v]);

             }

        }

  forall(x in newSequence)
  {
          forall(y in Cell)
```

```
            {
                forall(z in Option)
                 {
                         C[x][y][z]== C[x-1][y][z]+
                        sum( aa in Product)( ProcessingTime[aa]*X[aa][x][y][z]);

                 }

            }

    }


    //constraint5
     forall(m in Sequence)
      {
            forall(l in Cell)
            {
                forall(n in Option)
                 {
                        C[m][l][n]-
                sum( o in Product)( ProcessingTime[o] *   X[o][m][l][n])<=T[m][l][n];
                 }

            }

      }



//constraint 6
  forall(p in Cell)
   {
           sum(q in Option)(y[p][q])<=1;
   }
//constraint 7

  Sum(r in Cell)sum(t in Option)(Operator[t]*y[r][t])<=Crewsize;

}
```

DATASET

nbProduct=8;

```
nbSequence=8;
nbCell=3;
nbOption=2;
Operator = [9, 10];
ProcessingTime=[8, 10, 7, 12, 16, 12, 18, 20];
```