Genetic Algorithms and Mathematical Models in Manpower Allocation and Cell Loading Problem

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

Cihan Babayiğit
November 2003
This thesis entitled
GENETIC ALGORITHMS AND MATHEMATICAL MODELS IN MANPOWER
ALLOCATION AND CELL LOADING PROBLEM

BY
CİHAN BABAYİĞİT

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

R. Dennis Irwin
Dean, Russ College of Engineering and Technology
Some solutions to the cell loading problem have been reported in the literature. However, Manpower Allocation and Cell Loading (MACL) problem is relatively new. Therefore, this thesis focuses on these issues. This study not only analyzes the MACL problem with a mathematical model and genetic algorithm (GA) but also it extends the mathematical model to include the number of tardy jobs concept and also adds original aspects, Multiple League and Extreme League, to the traditional GA methods. The objective of this thesis is to solve the MACL problem by both mathematical models and genetic algorithms and then compare the results in some cases. Results show that original methods in GA outperform the traditional methods in some cases. GA finds optimal or near optimal solutions much faster than a mathematical model does especially in large problems.

Approved:

Güresel A. Süer

Professor of Industrial and Manufacturing Systems Engineering
Dedication

I would like to dedicate this thesis to the people who supported me throughout all my life;

My mother Özdilek Babayiğit, my brothers Celal Babayiğit, Cemal Babayiğit, Cengiz Babayiğit, Kenan Babayiğit, my sisters Sibel Çakir, Sevgi Isik, Selma Akmaz and my father Kamer Babayiğit.
# Table of Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CHAPTER 1</strong></td>
<td>INTRODUCTION</td>
<td>13</td>
</tr>
<tr>
<td>1.1</td>
<td>Manufacturing Layouts</td>
<td>13</td>
</tr>
<tr>
<td>1.2</td>
<td>Manpower Allocation and Cell Loading</td>
<td>17</td>
</tr>
<tr>
<td>1.3</td>
<td>Thesis Structure</td>
<td>20</td>
</tr>
<tr>
<td><strong>CHAPTER 2</strong></td>
<td>BACKGROUND</td>
<td>21</td>
</tr>
<tr>
<td>2.1</td>
<td>Mathematical Models</td>
<td>21</td>
</tr>
<tr>
<td>2.2</td>
<td>Mathematical Modeling Language – ILOG OPL</td>
<td>22</td>
</tr>
<tr>
<td>2.3</td>
<td>Genetic Algorithms</td>
<td>23</td>
</tr>
<tr>
<td>2.4</td>
<td>Computer Programming Language – Visual Basic</td>
<td>25</td>
</tr>
<tr>
<td>2.5</td>
<td>Problem Statement</td>
<td>25</td>
</tr>
<tr>
<td>2.6</td>
<td>Previous Work</td>
<td>27</td>
</tr>
<tr>
<td><strong>CHAPTER 3</strong></td>
<td>METHODOLOGY – MATHEMATICAL MODELS</td>
<td>30</td>
</tr>
<tr>
<td>3.1</td>
<td>MACL Problem</td>
<td>31</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Mathematical Model</td>
<td>31</td>
</tr>
<tr>
<td>3.1.2</td>
<td>OPL Program</td>
<td>33</td>
</tr>
<tr>
<td>3.2</td>
<td>Extended Models to Minimize the Number of Tardy Jobs subject to Manpower</td>
<td>36</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Mathematical Models Developed</td>
<td>37</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Basic Model (System I)</td>
<td>38</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Common-Cell Size Model (System II)</td>
<td>41</td>
</tr>
</tbody>
</table>
3.2.4 Lot-Splitting Allowed Model (System III) ................................................ 42
3.2.5 Lot-Splitting Allowed Complete Job Model (System IV) ....................... 43
3.3 Experimentation ................................................................................................ 43
3.4 Comparison of Systems .................................................................................... 49
3.5 Cost Analysis with System I ............................................................................. 51
  3.5.1 Case-I: Equivalent MPC and TFC with TVC .............................................. 52
  3.5.2 Case-II: MPC-Dominated .......................................................................... 53
  3.5.3 Case-III: TFC-Dominated .......................................................................... 54
  3.5.4 Case-IV: Equal MPC and TFC without TVC .............................................. 55
3.6 Summary ......................................................................................................... 56

CHAPTER 4 METHODOLOGY – GENETIC ALGORITHM ....................................... 57
  4.1 Proposed GA Scheme ...................................................................................... 57
    4.1.1 Gene Representation ............................................................................... 57
    4.1.2 Initial Population ................................................................................... 58
    4.1.3 Fitness Function .................................................................................... 58
    4.1.4 Reproduction ......................................................................................... 60
    4.1.5 Mating .................................................................................................... 61
    4.1.6 Crossover ............................................................................................... 61
    4.1.7 Mutation ................................................................................................ 64
    4.1.8 Termination Criteria ............................................................................... 65
    4.1.9 An Example for Application of GA ......................................................... 66
  4.2 Original Aspect ............................................................................................. 75
4.2.1 Multiple League ...................................................................................... 75
4.2.2 Extreme League....................................................................................... 78
4.3 Genetic Algorithm Software – MACL-6.0.................................................. 79

CHAPTER 5 ANALYSIS OF RESULTS .............................................................. 87
5.1 Single League Operators and Parameters Comparison ............................ 87
5.1.1 Population Size vs. Number of Generations .......................................... 89
5.1.2 Single-Cut Crossover vs. Two-Cut Points Crossover ............................... 91
5.1.3 Random Mutation vs. Arbitrary Two Product Change Mutation .......... 92
5.1.4 Low Crossover-Mutation Probability vs. High Crossover-Mutation

5.2 Multiple League Operators and Parameters Comparison.......................... 94
5.2.1 With Extreme League vs. Without Extreme League................................. 95
5.2.2 ‘2 League’ vs. ‘3 League’ ........................................................................ 96
5.2.3 ‘1 Season’ vs. ‘2 Seasons’ vs. ‘Continuous Seasons’ ................................. 97
5.2.4 1 Changing Team vs. 2 Changing Teams ............................................... 99
5.3 Single League and Multiple League Comparison ....................................... 100
5.3.1 Case I: ML - Without EL, 3 Leagues, 1 Season, 2 Changing Teams .......... 101
5.3.2 Case II: ML - With EL, 3 Leagues, 2 Seasons, 1 Changing Team .......... 102
5.3.3 Case III: ML – With EL, 2 Leagues, Continuous Seasons, 2 Changing

Teams ........................................................................................................... 103
5.4 Genetic Algorithm vs. Mathematical Model (MACL)............................... 105

CHAPTER 6 CONCLUSION AND FUTURE RESEARCH ............................. 107
6.1 Conclusion ........................................................................................................ 107
6.2 Future Research .............................................................................................. 108
REFERENCES ...................................................................................................... 110
APPENDIX A: Mathematical Programs of Extended Models ............................. 113
APPENDIX B: Sample Model Files for Extended Models ................................. 116
APPENDIX C: Tables for Model I Tardiness Calculation .................................. 128
APPENDIX D: Table for GA Experimentation .................................................... 141
List of Tables

Table 3.1. Data for the Sample Problem ................................................................. 44
Table 3.2. Manpower Distribution to Cells at W = 55. ........................................ 45
Table 3.3. Job Distribution to Cells at W=55. ....................................................... 47
Table 3.4. Number of Tardy Jobs. ........................................................................ 50
Table 3.5. Data Input for Different Cases ............................................................ 52
Table 4.1. Processing Times of Products .............................................................. 67
Table 4.2. Random Numbers for Chromosome 1 ................................................. 68
Table 4.3. Cumulative Probabilities of Cell Assignment ....................................... 68
Table 4.4. Chromosome 1 in Decoded Form ...................................................... 69
Table 4.5. Fitness Function Values of Sample Chromosomes ......................... 71
Table 4.6. Cumulative Probabilities of Reproduction ......................................... 72
Table 4.7. Cumulative Probabilities of Mating of C1 .......................................... 73
Table 4.8. Cumulative Cell Alternatives for a Mutated Gene ............................. 74
Table 4.9. SL Terms vs. ML Terms ................................................................. 76
Table 4.10. SL Calculation vs. ML Calculations .............................................. 77
Table 4.11. MACL-6.0 Events and Their Explanations ........................................ 81
Table 4.12. Explanation of Functions in geneticAlgorithmCycle() Function .... 85
Table 5.1. Default Values for SL Comparison ................................................... 87
Table 5.2. The Optimal Solution to MACL Problem ........................................... 88
Table 5.3. Experimentation Cases ..................................................................... 89
Table 5.4. Probabilities of Crossover and Mutation Operators. ................................. 93
Table 5.5. Default Values for ML Comparison ................................................................ 95
Table 5.6. Operators and Parameters Used for all leagues in SL and ML Comparison. 100
Table 5.7. Population Size and Number of Generations for 50 Product Problem......... 105
Table 5.8. Scenarios and Results for GA and Mathematical Model Comparison. ........ 106
List of Figures

Figure 1.1. Process Layout ................................................................. 14
Figure 1.2. Product Layout .................................................................. 15
Figure 1.3. Cellular Layout ................................................................. 16
Figure 1.4. Comparison of Different Layouts ....................................... 17
Figure 1.5. Cell Loading and Manpower Allocation a & b .................... 18-19
Figure 3.1. OPL Model File for MACL Model ....................................... 35
Figure 3.2. Example Data File for MACL Model .................................... 36
Figure 3.3. OPL Model File for Basic Model ......................................... 40
Figure 3.4. Example Data File for Extended Models ........................... 41
Figure 3.5. OPL Code for Equation 3.12 ............................................... 42
Figure 3.6. OPL Code for Equation 3.13 ............................................... 43
Figure 3.7. Schedule Generated with 4 Systems at W=55, a-d ............. 48-49
Figure 3.8. System Comparison .......................................................... 50
Figure 3.9. Cost Curves for Case I ....................................................... 53
Figure 3.10. Cost Curves for Case II ...................................................... 54
Figure 3.11. Cost Curves for Case III ..................................................... 55
Figure 3.12. Cost Curves for Case IV ..................................................... 56
Figure 4.1. Chromosome and Gene Representation for GA ................... 58
Figure 4.2. Single Cut Crossover Operation ......................................... 62
Figure 4.3. Two Cut Point Crossover Operation .................................... 64
Figure 4.4. Chromosome 1 in Coded Form ................................................................. 68
Figure 4.5. Graphical Representation of ML with 3 Leagues................................. 77
Figure 4.6. Graphical Representation of EL .............................................................. 79
Figure 4.7. Parameter Determination Page of MACL-6.0 Program ....................... 82
Figure 4.8. Multiple League Parameters Page of MACL-6.0 .................................. 83
Figure 4.9. Extreme League Parameters Page of MACL-6.0 ................................. 84
Figure 4.10. Code of geneticAlgorithmCycle() Function in MACL-6.0 ................. 86
Figure 5.1. Comparison of Population Size and Number of Generations ............. 90
Figure 5.2. Comparison of Different Crossover Operators .................................... 91
Figure 5.3. Comparison of Different Mutation Operators ....................................... 92
Figure 5.4. Comparison of Different Crossover and Mutation Probabilities ........... 94
Figure 5.5. Comparison of GAs With and Without Extreme League .................... 96
Figure 5.6. Comparison of Different League Number ............................................. 97
Figure 5.7. Comparison of Different Number of Seasons ...................................... 98
Figure 5.8. Comparison of Different Number of Changing Teams ....................... 99
Figure 5.9. SL and ML Comparison for Case I ...................................................... 101
Figure 5.10. SL and ML Comparison for Case II .................................................... 103
Figure 5.11. SL and ML Comparison for Case III ............................................... 104
CHAPTER 1

INTRODUCTION

In this chapter, first different manufacturing layouts are explained briefly. Then, the problem studied is stated along with the objectives of the study. Finally, work done to accomplish these objectives is summarized.

1.1 Manufacturing Layouts

There are three basic layout types in manufacturing namely process, product, and cellular. Traditional manufacturing layout is based on Process Layout which is also called Functional Layout. In process layout, similar machines are grouped together in departments and parts (Parts, Products, and Jobs are used interchangeably throughout this thesis) travel to different departments for the operations to be completed. This layout places little constraint on the type of product manufactured and therefore it is suitable for most production situations [16]. Furthermore, different parts can be produced in small batch sizes but material-handling cost is high and system productivity is low. A sample process layout can be seen in Figure 1.1.
Product Layout is another traditional layout which aims to eliminate the transfer of parts between different departments observed in the functional layout. In a product layout, products are dedicated to certain flow lines in which all the necessary machine types to complete a product are brought together in one line. This layout is appropriate for high volume but limited product range manufacturing. It has higher productivity but lower flexibility in terms of product change compared to the process layout. A sample product layout design can be seen in Figure 1.2.
Cellular Manufacturing (CM) is the division of production processes into smaller units called cells [6]. Cellular Manufacturing gained recognition in the 1960s within the more general approach of Group Technology. A cellular approach attempts to combine the operational simplicity of a flow line with the flexibility of functional layout [16]. This is done by grouping the similar parts into families and then assigning families to the cells. Families are established based on the similarities among the manufacturing processes of parts. Once a part family is formed then all the machines that are required for manufacturing of the products in that product family are assigned to one cell. However, Ortega and Süer [10], emphasize that in some cases, manufacturing cells might process products from different families and similarly, a family can be assigned to more than one cell, to introduce more flexibility into the system. A sample cellular layout can be seen in Figure 1.3.
In this study, part families are assumed to be known in the MACL problem. Moreover, only one part family is focused on. The problem is assigning the parts in that part family to similar cells which can process all the parts. CM aims quality improvements, work-in-process inventory reduction, product changeover simplification, and manufacturing throughput time reduction [20]. Furthermore, CM simplifies planning and scheduling activities. On the other hand, low machine utilization is the main disadvantage of cellular layout. CM is mostly suitable for the manufacturing firms which produce middle volume and moderate variety of products. Comparison of process, product, and cellular layout is shown in Figure 1.4 from volume and product variety point of views.
1.2 Manpower Allocation and Cell Loading

Manufacturing cells differ from machine-intensive cells to labor-intensive cells. In machine-intensive cells, the number of machines of each type is the key factor to determine the output of the cell. The impact of labor assignment to the output is usually limited. Whereas in labor-intensive cells, machines are small, light, cheap and require more manpower involvement. Therefore, it is assumed that the availability of machines does not constitute a problem. This problem has been observed in jewelry manufacturing, medical device assembly and apparel manufacturing among others. The number of operators assigned to a cell can be changed from one period to the next to adjust to the varying demand. As a result, alternative cell sizes are generated by allocating different number of operators to each cell. However, the total number of operators in the cellular system cannot be exceeded even though the allocation from one cell to the next cell and
from one period to the next period can vary. As a result, manpower level in the cell plays an important role in the output of the cell.

In labor-intensive cells, cell loading and manpower levels are closely related. The higher the load, the higher the manpower level to meet customer demand. However, cell loading and manpower allocation is not necessarily proportional. In other words, if manpower in a cell is doubled, it does not necessarily mean that amount of work can be doubled. Cell loading and Manpower Allocation steps are shown in Figure 1.5. Basically, cell loading is assigning products to cells as shown in Figure 1.5 (a). Manpower allocation is determining the number of workers to be assigned to cells as shown in Figure 1.5 (b). There are some methods to find the optimal cell loading to minimize the total number of workers in all cells. Süer [12] proposed a mathematical model to solve this problem in which Manpower Allocation and Cell Loading (MACL) decisions are made simultaneously.
While several solutions to the cell loading problem have been reported in the literature, research on cell loading and manpower allocation together is relatively new. This thesis focuses on these issues and contributes to the field by proposing mathematical models and a Genetic Algorithm approach. The mathematical model proposed by Süer [12] is used with a slight change in this study. Furthermore, a meta-heuristic method, the Genetic Algorithm (GA), is also used to solve this problem. Main reason for using a genetic algorithm is to provide an alternative way when mathematical model becomes inefficient due to high execution time in solving big problems. In short, this thesis focuses on Manpower Allocation and Cell Loading issues. It not only analyzes the problem with a mathematical model and genetic algorithm but, it extends the
mathematical model to include the concept of number of tardy jobs and it adds an original aspect to the traditional GA methods.

1.3 Thesis Structure

This thesis covers the following topics:

- A Mathematical Model is developed to solve the MACL Problem
- Four different Mathematical Models (Extended Models) are developed to solve the MACL Problem to minimize number of tardy jobs for different scenarios
- Efficient programs are written with Mathematical Programming Software, ILOG OPL, for MACL problem model and Extended Models
- Mathematical Models are solved with the developed Mathematical Programs and some experimentation is performed with Extended Models
- A flexible Genetic Algorithm Software is developed in Visual Basic 6.0 Studio, for MACL problem
- An original aspect is introduced to the GA Software called League Based Genetic Algorithm
- Some experimentation is performed with the developed GA Software to find the best parameters
- The GA results are compared with optimal solutions found with MACL Model
CHAPTER 2

BACKGROUND

In this chapter, the methods used to solve the MACL problem are explained. These methods are mathematical models and genetic algorithms. ILOG OPL is the mathematical modeling language that is used to program the mathematical models. The Visual Basic computer language is used to develop the genetic algorithm software. General information about ILOG OPL and Visual Basic is given in this Chapter. Finally, the problem is summarized and previous studies are given.

2.1 Mathematical Models

Mathematical models use mathematics to formulate the problem. They have certain objectives and try to optimize this objective. They can be used to solve problems in a variety of application areas, which include planning, design, configuration, scheduling and resource allocation. The main advantage of mathematical models is that, upper management understands them easily since they give definite and best results [5]. Other advantages are that they can quantify trade-offs between objectives and they can examine the implications of changing resource constraints by doing sensitivity analysis. On the other hand, they have some drawbacks. The main drawback is the huge computational time required. Many integer programming and combinatorial optimization problems are still challenging from a computational standpoint; They are NP-complete. It is widely believed that no general and efficient algorithm exists for solving them. However,
manufacturing systems are dynamic systems and generally require a solution in short time.

The model that was developed by Süer [12] has been used in this study. New models to study the MACL have been developed. These new models extend the MACL problem to include the minimization of tardy jobs as an objective function. All these models are given explicitly in Chapter 3.

2.2 Mathematical Modeling Language – ILOG OPL

Traditional modeling languages such as AMPL (A Mathematical Programming Language) and GAMS (General Algebraic Modeling System) were developed to make it easy for the user to solve mathematical programming problems. The way they tried to achieve this was expressing mathematical problems in a computer language whose syntax is close to the standard presentation of these problems in books. In other words, mathematical equations are written with a computer language in a way which is similar to the algebraic notation of the equation.

OPL (Optimization Programming Language) was motivated by the modeling languages AMPL and GAMS. It provides strong support for modeling linear and integer programs like AMPL and GAMS. Furthermore, it makes it possible to solve combinatorial optimization applications, such as job-shop scheduling and variety of resource allocation problems.

OPL allows the models to be used again by separating the model and instance data. Model just declares the data and data file initializes the declared item. As a result, the same model can be used with different data files. A model and a data file are combined
and form a project. This property of OPL was useful in performing the experiment since many test samples were experimented to reach a thorough comparison among Extended Models and between MACL model and genetic algorithm. Mathematical models written with OPL language are presented in Chapter 3.

2.3 Genetic Algorithms

Evolutionary process originates by the rules of natural selection in Darwin’s theory. Techniques which use the mechanics of evolution to produce optimal solutions to the problems are known as evolutionary computation. Genetic Algorithm is one of the best known evolutionary process techniques.

GA has many different properties from conventional optimization and search procedures. First of all, most of the search algorithms work with solutions directly but GAs work with the coding of the solutions. Second, most of the search algorithms like simulated annealing start from a single solution whereas GAs start the search from a population of solutions. Finally, most of the other search algorithms like Branch and Bound use deterministic transition techniques whereas GAs use probabilistic transition techniques.

Genetic algorithms not only explore but also exploit the search space to find the best solution. Gen and Cheng [4] state that GAs combine elements of directed and stochastic search which can make a remarkable balance between exploration and exploitation of the search space. At the beginning of genetic search, population is widely random and diverse. Exploration of the solution space is dominant. As better fitness solutions evolve, genetic search directs to the exploitation of the good solutions.
GAs start with a set of solutions called chromosomes. Chromosomes are made of coding of solutions called genes. Then GAs mimic nature, on the way to reach the optimal (or best) solution. GAs use payoff information (fitness function) to evaluate the solutions (chromosomes) and compare them. There are two kinds of operations in GA; The first type is Genetic Operations such as Crossover Operator and Mutation Operator. The second type is Evolution Operation which is carried by the Selection Operator. Crossover operators operate on two chromosomes at a time and generate offspring by combining both chromosomes’ features. Mutation operators produce random variations in chromosomes by altering one or more genes in the chromosome. Selection operators form the next generation from the current chromosomes randomly. However, chromosomes with a better fitness function have a higher chance to survive.

Advantages of GAs can be summarized as:

- They do not require much mathematics
- They can work with many objective functions
- They are effective at global search
- They can be used with local heuristics or optimization techniques

Disadvantages of GAs can be summarized as:

- They do not guarantee an optimal solution
- They need problem specific parameters

The coding and fitness function used for the chromosomes in MACL problem is explained in detail in Chapter 4 together with crossover, mutation, and termination criteria strategies.
2.4 Computer Programming Language – Visual Basic

Visual Basic (VB) is a high-level programming language from Microsoft. It is based on the BASIC language created in 1964 by John Kemeny and Thomas Kurtz. It is graphically oriented. VB can be used to develop everything from simple database applications to software packages. VB is relatively easy to learn so it is often taught as a first programming language today.

Microsoft has improved BASIC in its Visual Basic product. The center of VB is the form on which components such as menus, pictures, and command boxes are dragged and dropped. These items are known as "widgets." Widgets have properties such as its color, size and events such as clicks. VB is mostly used to create quick and simple interfaces to other Microsoft products such as Excel and Word. It is also possible to create full applications with VB. Object oriented programming is the new and popular programming paradigm and VB allows the users to apply this technique in its later versions besides structured programming technique.

The software for GA was written in VB. After the 1st Version of just basic genetic algorithm features, new improvements have been made. Its final version is MACL 6.0. It is written as a structured program with event-driven approach. An Event in VB means an action like clicking on a button or moving mouse. The details and features of MACL 6.0 are given in Chapter 4.

2.5 Problem Statement

Manpower allocation and cell loading problem consists of assigning jobs to cells and determining the cell size, in other words, determining how many workers should be in
each cell. Although both cell loading and manpower allocation are done simultaneously, the main objective is to assign jobs to cells in such a way that total number of workers required in all cells is minimized. In the mathematical model, there are constraints for assigning jobs which enforce each job to be assigned to a cell while minimizing the total manpower needed (objective function). In the genetic algorithm application, each product is automatically assigned to one of the cells. Then total manpower of the chromosome is calculated which becomes the fitness function of the chromosome. Since the survival probability of chromosomes is determined by the fitness function, minimizing the total manpower is still the main objective.

The origin of this problem is *Uniform Parallel Machine Scheduling Problem to Minimize Tardy Jobs with Zero-Ready Times*. In our problem, cells correspond to machines. All the cells are ready to process the jobs simultaneously so cells can be considered as parallel machines. Since they can have different worker configuration which determines the speed of the cell, they are not identical. They are uniform instead because the same job is always completed in shorter time in a cell with higher manpower. In other words, faster cells finish the job earlier. Zero-ready times are taken since all the jobs are assumed to be available at the beginning of the period. The objective can be considered as minimizing the number of tardy jobs because this is a constraint in mathematical models and in genetic algorithm; there is a penalty cost when a job cannot be finished in one period. In genetic algorithms, due dates of all jobs can be considered as one period. The main difference between MACL problem and parallel machine scheduling problems is that in uniform parallel machine problem, the number and the
speed of the machines are certain whereas in MACL problem both the number of cells and configuration of cells have to be determined by the solution.

### 2.6 Previous Work

Although MACL is an important subject, it is difficult to find enough literature and research related on this problem. It is especially difficult to find relevant literature that is related with how genetic algorithms are applied to solve MACL problems. However, attempts to solve this problem with optimizing and heuristics methods are more common in the literature.

Cheng, Chen and Li [3] examined the trade-off curve between resource allocation and number of tardy jobs in single machine scheduling and proved that minimizing total manpower subject to a limited number of tardy jobs is NP-hard. These problems can be solved by optimizing methods such as dynamic programming, branch and bound, and integer programming [11], [12]. Furthermore, these methods find optimum solutions for small sized problems but in big problems mathematical optimization techniques are insufficient in which case meta-heuristic methods are used to solve the problem.

extended this problem to loading in a multi-period cellular environment. He analyzes different approaches with advantages and disadvantages. He gives the mathematical formulations for the different approaches. Süer [12] introduced a two-phase hierarchical methodology to find the optimal manpower allocation and cell loads simultaneously. First phase is to generate the alternative configurations and second phase is to find optimal operator assignment and to load cells. He solved the problem by using mathematical programming. Süer and Bera [15] also used a two-phase approach to solve both manpower assignment and cell loading in a multi-period cellular system. The objective was to minimize the number of tardy jobs with the available capacity in each period. It was observed that number of tardy jobs decreased as number of employees increased in the cell. Vembu and Srinivasan [18] developed heuristics to minimize the makespan on a product line. They divided the line into cells such that movement in a cell is one way and single. Production times change with manpower level. They also solved the problem with enumeration methods and genetic algorithm. The developed heuristic gave close to best solutions quickly which are found by enumeration methods and genetic algorithm.

Süer, Saiz, and Gonzales [17] analyzed cell loading for an environment in which there are many cells which can produce parts from more than one part-family and which are independent. Independent cell means the product is finished in the cell. They developed new rules and two algorithms for cell loading. None of the rules dominated the others for all performance measures taken into consideration. On the hand when only number of tardy, total tardiness, and maximum tardiness are focused, a few rules produced similar results. Süer, Saiz, Dagli, and Gonzalez [16] studied the cells which are connected. In other words, output of one cell becomes the input of another cell. One cell
can process a part from more than one family. However, there is at least one part family which can not be processed by all the cells. There was again no single rule that performed good for all performance measures. The number of tardy, total tardiness and maximum tardy behaved similarly. Although experimentation is done with real data, it can not be generalized for other environments.

There are only a few attempts to solve the MACL problem with genetic algorithms in literature. Aickelin and Dowsland [1] stated that classical genetic algorithms are not capable of handling the conflict between objectives and constraints that occur in manpower scheduling problems. The authors proposed a GA for a nurse roistering problem in a U.K. hospital; The GAs converge to good solutions quickly but crossover operator was too disruptive to make small changes. As a solution to this obstacle, attributes which provide better solutions are rewarded and those which give poor solutions are punished. Empirical solutions based on 52 weeks show that GA give practical solutions to the problem. They not only used genetic algorithms but also used mathematical programming, heuristics and co-evolution to solve the problem.

Cai and Li [2] proposed a new genetic algorithm. They considered that the workers have mixed skills and they used multi-criteria optimization. The primary objective is to minimize the cost of assigning workers over different periods. The secondary objective is to find a solution with maximum surplus of workers. The tertiary objective is to reduce the variation of worker surplus for different periods. They state that their proposed GA algorithm is different than traditional GAs because mating was done according to rank among chromosomes. Another difference is, they used a multi-point crossover operator. Their proposed GA adopts a heuristic which fixes infeasible solutions.
CHAPTER 3

METHODOLOGY – MATHEMATICAL MODELS

In this chapter, mathematical models developed for the MACL problem are discussed. Mathematical models (Extended Models) which include the concept of tardy job are developed. A job is called tardy when it is completed beyond its due date \( (c_i > d_i) \) where \( c_i \) is the completion time and \( d_i \) is the due date of job \( i \). All of the models have been solved by using the software ILOG-OPL. The codes for MACL Model and one of the Extended Models (Basic Model) are given in this chapter while the codes for other Extended Models are given in APPENDIX A. A sample code written in basic OPL for a problem consisting of 4 products, 3 cells, and 2 configurations, is given in APPENDIX B.

The main assumptions made in developing these models are:

- Jobs can be performed by all cells
- Processing time of a job depends on the manning in a cell (i.e., the higher the manpower level, the higher the production rate and hence lower in-cell time, i.e., processing time)
- Zero set-up times (i.e. negligible)
- All operators can perform all tasks (i.e., skill levels don’t affect operator assignment)
- Each job is equally important

Each model does the following tasks simultaneously:

- Determine the number of cells to open
• Determine cell size among alternatives for each opened cell
• Assign products to cells (cell loading)
• Determine product sequence in each cell (Extended Models only)

3.1 MACL Problem

The mathematical model used to solve this problem is similar to the one proposed by Süer [12]. The only difference being that cell configurations or alternative worker numbers to be assigned to cells are defined independent of the cells. In this study, this change eliminates the feasible cell - worker configuration concept. In other words, it is assumed that all the present cells have the capability of all worker configurations. As a result, some of the parameters of the original model have been eliminated.

3.1.1 Mathematical Model

Notation

The following notation is used for MACL model.

\[ i \quad \text{index for product} \quad (i=1,2,...,n) \]
\[ j \quad \text{index for cell} \quad (j=1,2,...,m) \]
\[ k \quad \text{index for alternative configuration} \quad (k=1,2,...,l) \]
\[ p_{ik} \quad \text{in-cell time for the product} \ i \ \text{for cell configuration} \ k \]
\[ b_k \quad \text{# operators for alternative configuration} \ k \]
\[ W \quad \text{total # operators available} \]
\[ X_{ijk} \begin{cases} 1 & \text{if product} \ i \ \text{is assigned to cell} \ j \ \text{with configuration} \ k \\ 0 & \text{otherwise} \end{cases} \]
MACL Model

This integer programming model minimizes the total number of manpower in all cells as given in Equation 3.1. Equation 3.2 guarantees that each job is assigned to a cell. A cell can be assigned at most one alternative configuration as represented by Equation 3.3. Equation 3.4 assures that the capacity of the cells is not exceeded. Finally, all decision variables are binary variables as indicated by Equation 3.5.

An important assumption made for this model is when the fastest cell configuration (or the highest manpower level) is assigned to all cells; all the jobs can be processed in the cells. The reason is that if the maximum number of cells with fastest cell configuration is not capable of processing all the jobs, then Equation 3.2 is violated. As a result, tardiness of a job occurs which is analyzed later in Section 3.2.

**Objective Function:**

\[
\min Z = \sum_{k=1}^{l} \sum_{j=1}^{m} b_k * Y_{jk} \tag{3.1}
\]

**Subject to:**

\[
\sum_{j=1}^{m} \sum_{k=1}^{l} X_{ijk} = 1 \quad \text{for} \quad i = 1, 2, \ldots, n \tag{3.2}
\]

\[
\sum_{k=1}^{l} Y_{jk} \leq 1 \quad \text{for} \quad j = 1, 2, \ldots, m \tag{3.3}
\]

\[
\sum_{i=1}^{n} p_{ik} * X_{ijk} \leq C * Y_{jk} \quad j = 1, 2, \ldots, m \quad k = 1, 2, \ldots, l \tag{3.4}
\]

\[
Y_{jk} = \begin{cases} 
1 & \text{if cell } j \text{ has alternative configuration } k \\
0 & \text{otherwise}
\end{cases}
\]

C capacity of a cell per period
\[ X_{ijk} \text{ and } Y_{jk} \in (0,1) \text{ for all } i, j, k \]  

(3.5)

3.1.2 OPL Program

OPL is a powerful mathematical modeling language. Although it allows the user to code the models in traditional ways by entering each constraint successively, it also enables coding in compact format through loops and data files. This property is crucial in big problems since the number of parameters and variables increases geometrically. As a result, the size of the model decreases dramatically which makes it easier to manage the program and to experiment with different data sets. Figure 3.1 illustrates OPL Model File and Figure 3.2 shows an Example Data File for the MACL Model. The MACL model is used to solve the test problems in Chapter 5.

Explanation of the Program

Referring back to Figure 3.1 and Figure 3.2, it should be noted that each of the constraints in MACL model is written within one loop in the OPL program. Furthermore, a new problem can be solved with the same model by just changing the numbers in the data file.

Number of products (nbProduct), number of cells (nbCells) and number of configurations (nbConf) are declared as integer in the first 3 lines of the code. The initializations of these parameters are done in data file as shown in Figure 3.2. Three consecutive dots at the declaration of number of products, number of cells, and number of configurations after equal sign mean that these values are going to be read from a data file which is linked to the model. Then the range of the actual number of products, cells and configurations is determined by ‘nbProduct’, ‘nbCells’, and ‘nbConf’, respectively.
A range type ‘boolean’ which consists of only 0, 1 integers is created. This range is used in the declaration of variable arrays under the keyword ‘var’. Then, an array of cell configuration (workerOpt[]), an array of processing time of products (processTime[]), and the time capacity of a cell for one period (cellCapacity) are declared. The initializations are done by reading the values from the data file. Final declarations are for product decision variable array (prDecVar[]) and cell decision variable array (clDecVar[]). Then the objective function, which aims to minimize the sum of manpower by checking all possible configurations in all cells, is declared. Next, the constraints are written in loops (forall keyword). Loops 1, 2, and 3 correspond to Equation 3.2, Equation 3.3, and Equation 3.4, respectively.
int nbProducts = ...;
int nbCells = ...;
int nbConf = ...;

range
    Products 1..nbProducts,
    Cells 1..nbCells,
    Conf 1..nbConf,
    boolean 0..1;

int workerOpt[Conf] = ...;
float processTime[Products, Conf] = ...;
float cellCapacity = ...;

var
    boolean clDecVar[Cells, Conf],
    boolean prDecVar[Products, Cells, Conf];

minimize
    sum(c in Cells & co in Conf)(clDecVar[c,co]*workerOpt[co])

subject to {
    forall(p in Products)
        sum(c in Cells & co in Conf) prDecVar[p,c,co] = 1;

    forall(c in Cells)
        sum(co in Conf)
            clDecVar[c,co] <= 1;

    forall(c in Cells & co in Conf)
        sum(p in Products)
            processTime[p,co]*prDecVar[p,c,co] <= cellCapacity*clDecVar[c,co];
}

Figure 3.1. OPL Model File for MACL Model
In this section, four different mathematical models are developed to solve the cell loading problem. The objective is to minimize the number of the tardy jobs subject to manpower restriction. Minimizing the number of tardy jobs ($n_T$) is a crucial performance measure for customer satisfaction. On the other hand, minimizing the total manpower requirement is also critical in order to reduce labor costs and hence increase the survivability of a manufacturing company. In short, both customer service and product cost are significant parameters that affect competitiveness of a manufacturing system. However, minimizing the number of tardy jobs and reducing the manpower requirement
are conflicting measures. In other words, as the number of operators increases, the number of tardy jobs is expected to be lower and vice versa. In this section, these relations are explored and an approach to find the optimal manpower level and corresponding minimum \( n_T \) is presented.

3.2.1 Mathematical Models Developed

A different mathematical model is developed for each of four different systems. System I is the basic system where products are assigned to cells to minimize the number of tardy jobs subject to available manpower, whereas System II restricts all cells to a common cell size. In other words, same number of workers is allocated to all the opened cells. System III extends System I by allowing lot-splitting. Lot-splitting means a job can be assigned to more than one cell. System IV allows lot-splitting only if the entire job can be completed by its due date. Each model determines the sequence of products in each cell which is not taken into consideration in the MACL model. Basically, the sequence of the products is determined by Earliest Due Date (EDD) rule which orders all the products at the beginning.

**Notation**

The following notation is used for Extended Models.

- \([i]\) index for the product in the \( i^{th} \) order \((i=1,2,\ldots,n)\)
- \(j\) index for cell \((j=1,2,\ldots,m)\)
- \(k\) index for alternative configuration \((k=1,2,\ldots,l)\)
- \(p_{[i]k}\) in-cell time for the product in the \( i^{th} \) order for cell configuration \(k\)
- \(b_k\) number of operators for alternative configuration \(k\)
W total number of operators available

\[ X_{i,j,k} \begin{cases} 
1 & \text{if product in the } i^{th} \text{ order is assigned to cell } j \text{ with configuration } k \\
0 & \text{otherwise} 
\end{cases} \]

\[ Y_{j,k} \begin{cases} 
1 & \text{if cell } j \text{ has alternative configuration } k \\
0 & \text{otherwise} 
\end{cases} \]

d_{i} due date of the product in the \( i^{th} \) order

T_{i} control variable for the product in the \( i^{th} \) order

3.2.2 Basic Model (System I)

This integer programming model maximizes the number of early jobs (equivalent to minimizing the number of tardy jobs) as given in Equation 3.6. Equation 3.7 guarantees that a job can be assigned to at most one cell. A cell can be assigned at most one alternative configuration as represented by Equation 3.8. Equation 3.9 assures that the due-dates of the assigned jobs are not violated (thus early jobs). Finally, the total manpower assigned to all cells cannot exceed the manpower available in the cellular system as given by Equation 3.10. All decision variables are binary variables as indicated by Equation 3.11. OPL program for Basic Model is shown in Figure 3.3. Example Data File for Basic Model as well as other Extended Models is given in Figure 3.4. There is a new type created with keyword ‘struct’ in this model. The name of this new type is ‘ProductData’ and contains the attribute of float type named ‘dueDate’. Then, ‘product[]’ array is declared as ‘ProductData’. The reason of creating this new type is so that every product has a due date attribute which makes the programming easier. The 3rd loop in constraints uses this property.
Objective Function:

$$\text{max} \quad Z = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{l} X_{[i]jk}$$  \hspace{1cm} (3.6)$$

Subject to:

$$\sum_{j=1}^{m} \sum_{k=1}^{l} X_{[i]jk} \leq 1 \quad \text{for} \quad i = 1, 2, \cdots, n$$  \hspace{1cm} (3.7)$$

$$\sum_{k=1}^{l} Y_{jk} \leq 1 \quad \text{for} \quad j = 1, 2, \cdots, m$$  \hspace{1cm} (3.8)$$

$$\sum_{i=1}^{u} p_{[i]k} * X_{[i]jk} \leq d_{[i]} * Y_{jk} \quad \text{for} \quad u = 1, 2, \cdots, n; \quad j = 1, 2, \cdots, m; \quad k = 1, 2, \cdots, l$$  \hspace{1cm} (3.9)$$

$$\sum_{k=1}^{l} \sum_{j=1}^{m} b_{k} * Y_{jk} \leq W$$  \hspace{1cm} (3.10)$$

$$X_{[i]jk} \text{ and } Y_{jk} \in (0, 1) \quad \text{for all } i, j, k$$  \hspace{1cm} (3.11)$$
float daySecTime = ...;
float avCapacity = ...;
int totMan = ...;
int nbProducts = ...;
int nbCells = ...;
int nbConf = ...;
range
  Products 1..nbProducts,
  Cells 1..nbCells,
  Conf 1..nbConf,
  boolean 0..1;
struct ProductData {
  float dueDate; }
int workerOpt[Conf] = ...;
float processTime[Products, Conf] = ...;
ProductData product[Products] = ...;
Var
  boolean clDecVar[Cells, Conf],
  boolean prDecVar[Products, Cells, Conf];
Maximize
  sum(p in Products & c in Cells & co in Conf)(prDecVar[p,c,co])
subject to {
  forall(p in Products)
    sum(c in Cells & co in Conf)
      prDecVar[p,c,co] <= 1;
  forall(c in Cells)
    sum(co in Conf)
      clDecVar[c,co] <= 1;
  forall (i in 1..nbProducts & c in Cells & co in Conf)
    sum ( j in 1..i)
      processTime[j,co]*prDecVar[j,c,co] <=
        product[i].dueDate*daySecTime*clDecVar[c,co];
  sum (c in Cells & co in Conf)
    workerOpt[co] * clDecVar[c,co] <= totMan; };

Figure 3.3. OPL Model File for Basic Model
3.2.3 Common-Cell Size Model (System II)

Sometimes, the common-cell size option is preferred by the companies since this allows supervisors to have the same control span. Many supervisors see this as a fair workload issue. The objective function and other constraints remain the same except that Equation 3.8 is replaced by Equation 3.12. This constraint restricts the number of alternative configurations to a maximum of one (common cell size) while still letting the number of cells flexible. To apply this change to the OPL program, it is enough to replace
the code in Figure 3.3 written for Equation 3.8 with the code in Figure 3.5. However, there are exactly three counters in this code, ‘c1’, ‘c2’, and ‘c3’ which is written for 3 cell configurations. For a different number of cell configuration case, number of counters should be changed accordingly.

\[
Y_{j_1} + Y_{j_2} + \cdots + Y_{j_l} \leq 1 \quad \text{for } j_1 = 1, 2, \ldots, m
\]
\[
j_2 = 1, 2, \ldots, m
\]
\[
\vdots
\]
\[
j_l = 1, 2, \ldots, m
\]

(3.12)

Figure 3.5. OPL Code for Equation 3.12

3.2.4 Lot-Splitting Allowed Model (System III)

The mixed integer programming model given in this section allows lot-splitting. Partial completion of a job is acceptable. The decision variable \( X_{[c]jik} \) is allowed to take real values as opposed to binary values in Basic Model. On the other hand, \( y_{jik} \) still remains \((0,1)\). The objective function and other constraints are the same as in Basic Model. The only change in the OPL code is defining \( X_{[c]jik} \) as a float range.
3.2.5 Lot-Splitting Allowed Complete Job Model (System IV)

This system is an extension of System-III. It allows a job to be split among different cells like System-III but requires the entire job to be completed by its due date if it will be early (i.e., no partial early completion). Since $T_{[i]}$ can take only $(0,1)$ values, either the sum of partial lots of a job will be 1 or 0 as given by Equation 3.13. OPL code for this equation is given in Figure 3.6. The restrictions on decision variables are stated in Equation 3.14.

\[
\sum_{j=1}^{m} \sum_{k=1}^{l} X_{[i]jk} = T_{[i]} \quad \text{for} \quad i = 1, 2, \ldots, n \tag{3.13}
\]

\[
0 \leq X_{[i]jk} \leq 1 \quad \text{and} \quad Y_{jk} \quad \text{and} \quad T_{[i]} \in (0, 1) \quad \tag{3.14}
\]

![Figure 3.6. OPL Code for Equation 3.13](image)

3.3 Experimentation

A 12-job problem is generated to perform experimentation. The number of cells is limited to 3 and there are three alternative configurations for each cell (15, 20 and 25 operators). The total number of workers in the cell can be only one of these three
configurations. The in-cell processing times of each product for each alternative configuration and also due dates are given in Table 3.1.

Table 3.1. Data for the Sample Problem.

<table>
<thead>
<tr>
<th>Jobs</th>
<th>In-Cell Processing Times (days)</th>
<th>Due Dates (days)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Alt 1 15 op.</td>
<td>Alt 2 20 op.</td>
</tr>
<tr>
<td>[1]</td>
<td>1.67</td>
<td>1.25</td>
</tr>
<tr>
<td>[2]</td>
<td>2.08</td>
<td>1.56</td>
</tr>
<tr>
<td>[3]</td>
<td>2.50</td>
<td>1.88</td>
</tr>
<tr>
<td>[4]</td>
<td>1.15</td>
<td>0.83</td>
</tr>
<tr>
<td>[5]</td>
<td>1.35</td>
<td>0.94</td>
</tr>
<tr>
<td>[6]</td>
<td>2.08</td>
<td>1.56</td>
</tr>
<tr>
<td>[7]</td>
<td>1.25</td>
<td>0.83</td>
</tr>
<tr>
<td>[8]</td>
<td>1.67</td>
<td>1.25</td>
</tr>
<tr>
<td>[9]</td>
<td>1.56</td>
<td>1.04</td>
</tr>
<tr>
<td>[10]</td>
<td>1.25</td>
<td>1.04</td>
</tr>
<tr>
<td>[11]</td>
<td>1.67</td>
<td>1.46</td>
</tr>
<tr>
<td>[12]</td>
<td>2.08</td>
<td>1.67</td>
</tr>
</tbody>
</table>

The cell configuration results for each system at $W = 55$ are given in Table 3.2. It should be noticed that Systems I and II required only 50 operators even though the total manpower available was 55. This implies that using all 55 operators would not result in lower number of tardy jobs (both produced 2 tardy jobs). Obviously, System II could not use 55 operators since alternative configurations have 15, 20 and 25 operators and all cells are required to have same number of operators. On the other hand, both Systems III
and IV fully utilized the available manpower and hence managed to lower the number of tardy jobs to 0.42 and 1, respectively. The user will have to choose whichever system best suits his/her environment and the problem on hand.

In Table 3.3, the job sequence and their cell assignments along with their respective proportions are detailed for all four systems for the same example problem. The first number in the parenthesis denotes the cell number that the job is assigned to. The second number denotes the proportion of the job assigned to that cell. For example, the first job in the sequence is assigned to two cells based on System III results. Twelve percent of the job is assigned to cell 1 and the remaining 88% is assigned to cell 3. The initial job sequence that is input to all systems is the same. However, the actual job sequence on each cell depends on the output from the systems. The jobs that are not assigned to any

### Table 3.2. Manpower Distribution to Cells at W = 55.

<table>
<thead>
<tr>
<th>Cell</th>
<th>System I Basic</th>
<th>System II Common Cell Size</th>
<th>System III Lot Splitting Allowed</th>
<th>System IV Lot Splitting Allowed-Complete Job</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>--</td>
<td>25</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td>25</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
<td>--</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td><strong>Manpower Used</strong></td>
<td><strong>50</strong></td>
<td><strong>50</strong></td>
<td><strong>55</strong></td>
<td><strong>55</strong></td>
</tr>
<tr>
<td><strong>n_T</strong></td>
<td><strong>2</strong></td>
<td><strong>2</strong></td>
<td><strong>0.42</strong></td>
<td><strong>1</strong></td>
</tr>
</tbody>
</table>
cell become tardy and hence the sequence varies from one system to the next. If a job is not assigned to any cell, then it is tardy, like Job [2] in System I. The difference between Systems III and IV can be seen in this table. Even though job splitting is allowed in both systems, System III allows partial completion as in Job [2] with 58% completion, whereas System IV does not.

To see the effect of different due dates on the number of tardy jobs, each model is also run by decreasing the due dates of products to 80% and increasing it to 120% of original due dates. Results are presented in the last two rows of Table 3.3. As seen from the results, number of tardy jobs decrease from 80% to 120% for all systems. Enlarging due dates, gives the flexibility of squeezing extra jobs without being tardy.
Table 3.3. Job Distribution to Cells at W=55.

<table>
<thead>
<tr>
<th>Job</th>
<th>System I Basic</th>
<th>System II Common Cell Size</th>
<th>System III Lot Splitting Allowed</th>
<th>System IV Lot Splitting Allowed-Complete Job</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>(2,1.0)</td>
<td>(2,1.0)</td>
<td>(1,0.12) (3,0.88)</td>
<td>(3,1.0)</td>
</tr>
<tr>
<td>[2]</td>
<td>--</td>
<td>--</td>
<td>(3,0.58)</td>
<td>--</td>
</tr>
<tr>
<td>[3]</td>
<td>(3,1.0)</td>
<td>--</td>
<td>(1,0.04) (2,0.96)</td>
<td>(1,0.37) (2,0.63)</td>
</tr>
<tr>
<td>[4]</td>
<td>(2,1.0)</td>
<td>(2,1.0)</td>
<td>(3,1.0)</td>
<td>(3,1.0)</td>
</tr>
<tr>
<td>[5]</td>
<td>--</td>
<td>(1,1.0)</td>
<td>(1,1.0)</td>
<td>(1,1.0)</td>
</tr>
<tr>
<td>[6]</td>
<td>(3,1.0)</td>
<td>(1,1.0)</td>
<td>(1,0.2) (2,0.8)</td>
<td>(2,1.0)</td>
</tr>
<tr>
<td>[7]</td>
<td>(3,1.0)</td>
<td>(2,1.0)</td>
<td>(1,0.84) (3,0.16)</td>
<td>(1,1.0)</td>
</tr>
<tr>
<td>[8]</td>
<td>(3,1.0)</td>
<td>(1,1.0)</td>
<td>(1,0.4) (2,0.6)</td>
<td>(1,0.13) (3,0.87)</td>
</tr>
<tr>
<td>[9]</td>
<td>(2,1.0)</td>
<td>(2,1.0)</td>
<td>(1,0.04) (3,0.96)</td>
<td>(3,1.0)</td>
</tr>
<tr>
<td>[10]</td>
<td>(2,1.0)</td>
<td>(1,1.0)</td>
<td>(3,1.0)</td>
<td>(3,1.0)</td>
</tr>
<tr>
<td>[11]</td>
<td>(2,1.0)</td>
<td>(2,1.0)</td>
<td>(1,0.47) (2,0.53)</td>
<td>(2,1.0)</td>
</tr>
<tr>
<td>[12]</td>
<td>(3,1.0)</td>
<td>(1,1.0)</td>
<td>(3,1.0)</td>
<td>(1,0.51) (2,0.49)</td>
</tr>
</tbody>
</table>

\( n_T \) 

<table>
<thead>
<tr>
<th>( n_T ) (at 80% Due Dates)</th>
<th>2</th>
<th>2</th>
<th>0.42</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n_T ) (at 120% Due Dates)</td>
<td>3</td>
<td>4</td>
<td>2.14</td>
<td>3</td>
</tr>
</tbody>
</table>

\( n_T \) (at 120% Due Dates) | 1 | 2 | 0.42 | 1 |

Figure 3.7 (a), (b), (c), and (d) show the Gantt charts for the schedules generated by running Systems I, II, III and IV, respectively. The configurations used at these cells are
shown in Table 3.2. As seen from the charts the least idle time, zero idle time, is in System III where splitting of jobs is permissible. Then comes System IV in which splitting of a job is restricted to the completion of the entire job.

(a) System I

<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.8</td>
<td>5</td>
<td>8.4</td>
</tr>
<tr>
<td>3.8</td>
<td>6.9</td>
<td>8.9</td>
</tr>
</tbody>
</table>

(b) System II

<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>7.2</td>
<td>10.6</td>
</tr>
<tr>
<td>2.8</td>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>

(c) System III

<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>.6</td>
<td>.9</td>
<td>7.1</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>13</td>
</tr>
<tr>
<td>2.5</td>
<td>4.8</td>
<td>7</td>
</tr>
</tbody>
</table>
### 3.4 Comparison of Systems

Four models are solved for a total number of workers ($W$) varying from 15 to 75 since three configurations of 15, 20 and 25 were considered in the experimentation. The amount of manpower availability does not mean that all the workers are going to be used by the system. As seen in Table 3.4 and Figure 3.8, all of the systems except System III perform the same until $W = 30$ (inclusive). This is because, the same configurations are available for all three systems. After $W = 30$ System II gives more tardy jobs due to common cell restriction. For example at $W = 35$, both manpower levels 15 and 20 are available for all systems but System II.

Overall System III outperforms other systems as expected since jobs can be assigned partially which gives more flexibility. System IV is the second best system. It has an additional constraint which forces each job to be produced either totally in one period or not at all.
Table 3.4. Number of Tardy Jobs.

<table>
<thead>
<tr>
<th>Total # of Workers</th>
<th>System I</th>
<th>System II</th>
<th>System III</th>
<th>System IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>15</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>20</td>
<td>7</td>
<td>7</td>
<td>6.75</td>
<td>7</td>
</tr>
<tr>
<td>25</td>
<td>6</td>
<td>6</td>
<td>5.75</td>
<td>6</td>
</tr>
<tr>
<td>30</td>
<td>5</td>
<td>5</td>
<td>4.94</td>
<td>5</td>
</tr>
<tr>
<td>35</td>
<td>4</td>
<td>5</td>
<td>3.73</td>
<td>4</td>
</tr>
<tr>
<td>40</td>
<td>3</td>
<td>4</td>
<td>2.76</td>
<td>3</td>
</tr>
<tr>
<td>45</td>
<td>3</td>
<td>3</td>
<td>2.13</td>
<td>3</td>
</tr>
<tr>
<td>50</td>
<td>2</td>
<td>2</td>
<td>1.33</td>
<td>2</td>
</tr>
<tr>
<td>55</td>
<td>2</td>
<td>2</td>
<td>0.42</td>
<td>1</td>
</tr>
<tr>
<td>60</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>65</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>70</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>75</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 3.8. System Comparison
3.5 Cost Analysis with System I

The comparison of systems shows that as the manpower level increases, the number of tardy jobs decreases, as expected. The next question on hand is how to determine the optimal manpower level for a given scenario. For this purpose, experimentation has been performed with System I by using different combinations of manpower, fixed and variable tardiness costs. Manpower Cost (MPC) is for 5 workers per week. The fixed tardiness cost (TFC) is for a tardy job. A job is tardy if it is not finished by its due date ($c_i > d_i$). The variable tardiness cost (TVC) is per late day of each tardy job. Total cost is calculated as:

$$Total\ Cost = (MPC * W / 5) + (n_f * TFC) + (Total\ Tardiness * TVC)$$

Having assigned early jobs as given by the mathematical model, the tardy jobs are assigned to the cells according to the Minimum Completion Time rule. Tables that show the detailed calculation of tardiness is given in APPENDIX C. In the mathematical model, the tardiness of the jobs was not taken into consideration.

Another important point is assuming Total TVC for Zero Manpower. Since there is no manpower, processing times of jobs can not be calculated. Definite thing is that all the 12 jobs are tardy. Average tardiness of a job is calculated by rounding the longest processing times of jobs based on the lowest manpower (15). Finally, tardiness cost per job per day is 30. Total TVC for Zero Manpower becomes:

$$12*30*22.22=8000 \quad where \ 12 \rightarrow \ Number\ of\ Products, \ 30 \rightarrow TVC, \ 22.22 \rightarrow Average\ Tardiness\ per\ Job$$

Costs for all cases are shown in Table 3.5.
3.5.1 Case-I: Equivalent MPC and TFC with TVC

Since manpower levels are in increments of 5, MPC and TFC are considered equivalent in this graph. The worker cost increases as the manpower level increases but tardiness cost decreases. A total cost (TC) curve similar to a U-Shape graph is observed as seen in Figure 3.9. Minimum TC occurs at $W = 45$. An important point is MPC does not increase from $W = 40$ to $W = 45$ because the OPL-ILOG Software chose configurations of 15 and 25 for manpower at both points. Similarly, MPC does not increase from $W=50$ to $W=55$ for the same reason. From $W = 45$ to $W = 50$, TFC is almost same whereas there is a big increase in MPC. As a result, TC increases. TC decreases from $W = 50$ to $W = 55$ due to decrease in TFC. The same pattern is observed for the other cases too.
3.5.2 Case-II: MPC-Dominated

Since manpower cost (per 5 workers) is much higher than the fixed tardiness cost, TC increases with increasing $W$ despite the decreasing number of tardy jobs. Optimum result occurs at $W = 15$, where manpower level is very low. These results can be observed in Figure 3.10.

Figure 3.9. Cost Curves for Case I
3.5.3 Case-III: TFC-Dominated

Since fixed tardiness cost is much higher than the manpower cost, as seen in Figure 3.11, TC decreases with decreasing number of tardy jobs despite increasing $W$. Optimum occurs at $W=70$, where zero-tardy occurs for the first time. After that, TC increases a little bit with increasing $W$. 

Figure 3.10. Cost Curves for Case II
3.5.4 Case-IV: Equal MPC and TFC without TVC

The total tardiness cost and manpower cost are equal in Case-IV (since variable tardiness cost is ignored). As a result, TC stays the same as the number of tardy jobs decreases and manpower increases between $W = 15$ to $W = 45$ all of which are optimal. After $W = 45$, TC increases as the increase in manpower level is more than the decrease in number of tardy jobs. These observations can be seen in Figure 3.12.
3.6 Summary

This chapter presents mathematical models which minimize total number of workers in all cells (MACL Model) and to minimize number of tardy jobs subject to manpower restriction for different situations (Extended Models). Mathematical model and program is given for the MACL problem. The mathematical model, program and some experimentation and comparison is done for Extended Models. The four models are compared and the results show that the Lot-Splitting Allowed case gave the best results while the Common Cell case gave the worst results. As the manpower increases, the number of tardy jobs decreases but the total cost depends on the manpower costs and tardiness costs that are assigned.
CHAPTER 4
METHODOLOGY – GENETIC ALGORITHM

In this chapter, the GA developed to solve the MACL problem is explained from both concept and programming point of views. Details about gene representation are given with an example at the end of the section. All the GA operators and different strategies are discussed. The Multiple League Option which is added to GA as an original aspect is analyzed. Finally VB code of GA software is explained.

4.1 Proposed GA Scheme

Chromosome representation for the encoding space is probably the most important step in GA. In this section, both the chromosome representation and other GA operators and parameters are presented.

4.1.1 Gene Representation

Each chromosome consists of \( n \) genes \((n \leq 100)\). Each gene corresponds to a product. Position \( i \) represents part \( i \) in the chromosome and its value indicates to which cell, part \( i \) is assigned. The number of cells is determined by the user. The chromosome representation is given in Figure 4.1.
4.1.2 Initial Population

Population size or number of chromosomes is also specified by the user. The population size should be an even number. The reason is chromosomes are mated in numbers of two in crossover operator. The probability of assigning a part to a cell is another parameter determined by the user. As an example, if the user would like to assign more products to a certain cell, a higher probability can be assigned. Once these values are determined by the user then initial population is formed with random number generator according to the probabilities entered by the user. In other words, which product is assigned to which cell is determined by genetic algorithms randomly.

4.1.3 Fitness Function

The objective of this GA is minimizing the total number of workers in all cells. A fitness function is defined to meet this criterion. After finding which parts will be assigned to which cells, the value of fitness function is calculated for each chromosome individually.
Notation used throughout this chapter is identical to the notation used in Section 3.1.1 since both of the problems are MACL problem. The only difference is that in this study all cells are not assumed to be capable of doing all the jobs. When a cell is not capable of processing all the assigned jobs, then a Penalty Cost is added. The fitness function of a chromosome is calculated through following steps.

1. Go to 1st Cell. If any product (job) is assigned to this cell, perform the calculations in Step 2. Otherwise, the configuration in this cell is 0 (\( Y_{1k} = 0 \) for all \( k \)) and go to Step 3.

2. Check this cell with configuration \( b_1 \) (smallest cell configuration). If \( \sum_{i=1}^{n} p_{ik} X_{ijk} < C \) then \( Y_{11} = 1 \) and go to Step-4. Else \( Y_{11} = 0 \) and repeat computation with configuration \( b_2 \). If \( \sum_{i=1}^{n} p_{ik} X_{ijk} < C \) then \( Y_{12} = 1 \) and configuration of this cell becomes \( b_2 \) and go to Step-3. Else \( Y_{12} = 0 \) and repeat computation with configuration \( b_3 \), etc.

 Else, that is to say that if none of the previous configurations were feasible, \( b_f \) (highest cell configuration). If \( \sum_{i=1}^{n} p_{ik} X_{ijk} < C \) then \( Y_{1f} = 1 \) and configuration of the cell becomes \( b_f \). Else \( Y_{1f} = 0 \) and configuration of the cell becomes \( b_f + \) Penalty Cost. Go to Step-3.

3. Go to next cell and repeat Steps 1 & 2 for this cell too. If there is no cell remaining, go to Step-4.
4. Fitness Function for this chromosome is $FF = \sum_{j=1}^{m} b_j Y_{jk}$

4.1.4 Reproduction

The reproduction probabilities are calculated for all the chromosomes in the population. After the reproduction probabilities are calculated, a new population is formed from the old population by using these reproduction probabilities. The reproduction probabilities are computed by using fitness function values. The objective is to do cell loading to minimize total manpower. The reproduction probabilities for the chromosomes that have higher fitness value should be low and the reproduction probabilities for the chromosomes that have lower fitness value should be high.

To achieve this, the reproduction probabilities are calculated as shown below.

$$TFF = \sum_{i=1}^{i} FF_i$$  \hspace{1cm} (4.1)

$$AF_i = \frac{TFF}{FF_i}$$  \hspace{1cm} (4.2)

$$p_i = \frac{AF_i}{\sum_{i=1}^{i} AF_i}$$  \hspace{1cm} (4.3)

The notation used is given below:

$FF_i =$  \hspace{0.5cm} Fitness function value for chromosome $i$

$TFF =$  \hspace{0.5cm} Total fitness function value

$AF_i =$  \hspace{0.5cm} Adjusted fitness function value for chromosome $i$

$p_i =$  \hspace{0.5cm} Reproduction probability for chromosome $i$
s = Population size (s = 2, 4, 6, 8,...)

4.1.5 Mating

Pairs are formed randomly from the population and then mated. The probability of mating depends on how many chromosomes are present. If there are (n) chromosomes, then probability of mating the first chromosome with each of the remaining (n-1) chromosomes is $\frac{1}{n-1}$. The cumulative probabilities are calculated using individual probabilities and the random numbers are used to find which chromosomes will mate. The first chromosome to mate is also determined randomly, (it is not necessary to start initially with the first chromosome) all the chromosomes have equal probability to be chosen.

4.1.6 Crossover

The reason for doing crossover is to explore the solution space by changing the chromosomes and not to get stuck in local optima. Crossover is not necessarily performed for all pairs. For each pair a random number is generated. If this number is smaller than the Crossover Probability entered by the user, then crossover operation is applied to the pair. Otherwise, the crossover operator is not applied to this pair.

Two types of crossover operators or strategies are tested in this study:

- Single Cut Point Crossover
- Two Cut Point Crossover
Single Cut Crossover

As the name implies, Single-Cut Point Crossover is performed over a single point which is called the cut point. Parent chromosomes exchange their genes from the cut point and onwards. The cut point is selected randomly. If there are \( n \) products (genes), there will be \( (n - 1) \) possible cut points since initial point is not used as a cut point.

Probability of each cut point is equal to each other \( \frac{1}{n-1} \). After a random number is generated, the cut point is found according to the cumulative probability of cut points.

The Single Cut Crossover Operator is presented in Figure 4.2.

![Figure 4.2. Single Cut Crossover Operation](image-url)
Two-Cut Point Crossover

Two-Cut Point Crossover is slightly different than Single-Cut Point Crossover. Two points are chosen in this type of crossover and the genes in between these two cut points are interchanged within the parents to generate offsprings. Two Cut Point Crossover Operation is presented in Figure 4.3.

If there are (n) genes, the probability of first cut point for all genes is (1/n), since it is possible to start from the first gene of the chromosome also. The probability of second cut point is dependent on the first cut point. If the first cut point selected is either the 1st gene or the last gene, the probability of second cut point is \(\left(\frac{1}{n-2}\right)\) for the remaining genes because last gene is eliminated to be a second cut point if first cut point is first gene. Similarly, the first gene is eliminated to be second cut point if first cut point is last gene. The reason for this is when the first cut point is the first gene (or the last gene) and the second cut point is the last gene, the two parents will interchange all their genes. So, entire chromosomes will be interchanged and no real mating will be done. Otherwise, the probability of second cut point is \(\left(\frac{1}{n-1}\right)\).
4.1.7 Mutation

Mutation is performed to have random variations in chromosomes. Genes are changed depending on the mutation probability ($p_m$). In this study, mutation probability is determined by the user. Two types of mutation operators have been tested in the thesis.

- Random Mutation
- Arbitrary Two Product Change Mutation

**Random Mutation**

For each gene, a random number is generated. If the random number is smaller than $p_m$, then the gene is mutated. Otherwise, the gene remains the same. All the genes are tested against $p_m$. 

![Diagram of Two Cut Point Crossover Operation](image)
If a gene is to be mutated, the next step is to find the new cell assignment for that product. Depending on how many cells are present, the assignment probability is calculated. Another random number is generated to make this decision. Cumulative probabilities are computed. The interval that this random number falls in determines the new assignment.

*Arbitrary Two Product Change Mutation*

The main characteristic of this mutation strategy is that the mutation probability is specified for each individual chromosome whereas in Random Mutation, mutation probability is for each gene. Example, if \( p_a = 0.05 \), this indicates the mutation probability for that chromosome. If a chromosome is selected for mutation, the second step is to determine the two genes to be swapped. If there are \( n \) genes, the probability of first gene to be selected is \( 1/n \). The probability of second gene to be selected is \( 1/(n-1) \), since the first selected gene should not be selected as the second gene again.

4.1.8 Termination Criteria

The termination criterion determines how the genetic cycle stops. Two types of termination criteria are used:

- Number of Generations
- Threshold Value

*Number of Generations*

This is the termination criterion that is commonly used in the GA applications. In this approach, number of generations is determined in advance and the algorithms stops
as soon as it is reached. For example, if the number of generations is specified as 1000, then the algorithm stops at the 1000\textsuperscript{th} generation.

Threshold Value

In this type of termination, the genetic cycle stops when best fitness improvement is lower than a specified threshold value for a specified number of consecutive generations. Both the threshold value and the number of generations are determined by the user. For a minimization problem threshold termination can be formulated as follows:

\[
f_{\text{best},x} - f_{\text{best},x+1} \leq th
\]

\[
f_{\text{best},x} - f_{\text{best},x+2} \leq th
\]

\[
f_{\text{best},x} - f_{\text{best},x+3} \leq th
\]

\vdots

\[
f_{\text{best},x} - f_{\text{best},x+h} \leq th
\]

Then the program stops. Where,

\( f_{\text{best},x} \)  best fitness function in iteration \( x \)

\( th \)  threshold value (minimum desired improvement)

\( h \)  number of consecutive generations

4.1.9 An Example for Application of GA

In this section, the concepts and procedures discussed in Section 4.1.1 are illustrated with an example. The sample MACL problem has a period of one week. The processing times of products at different cell configurations are given in Table 4.1. Capacity of a cell
(C), penalty cost (PC), number of products (n), number of cells (m) and number of cell configurations (l) are as follows:

\[ C = 2400 \text{ minutes / week (40hours/week * 60 minutes/hour)} \]

\[ PC = 100 \text{ Workers (penalty cost applied to cells which are incapable of processing the assigned products in any configuration)} \]

\[ n = 12 \text{ (# of products)} \]

\[ m = 4 \text{ (# of cells)} \]

\[ l = 3 \text{ (# of cell configurations)} \]

\[ [b_1 \ b_2 \ b_3] = [15 \ 20 \ 25] \text{ (cell configurations)} \]

<table>
<thead>
<tr>
<th>( b_k )</th>
<th>( P_1 )</th>
<th>( P_2 )</th>
<th>( P_3 )</th>
<th>( P_4 )</th>
<th>( P_5 )</th>
<th>( P_6 )</th>
<th>( P_7 )</th>
<th>( P_8 )</th>
<th>( P_9 )</th>
<th>( P_{10} )</th>
<th>( P_{11} )</th>
<th>( P_{12} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>800</td>
<td>1000</td>
<td>1200</td>
<td>550</td>
<td>650</td>
<td>1000</td>
<td>600</td>
<td>800</td>
<td>800</td>
<td>600</td>
<td>800</td>
<td>1000</td>
</tr>
<tr>
<td>20</td>
<td>600</td>
<td>750</td>
<td>900</td>
<td>400</td>
<td>450</td>
<td>750</td>
<td>400</td>
<td>600</td>
<td>500</td>
<td>500</td>
<td>700</td>
<td>800</td>
</tr>
<tr>
<td>25</td>
<td>450</td>
<td>600</td>
<td>650</td>
<td>325</td>
<td>350</td>
<td>550</td>
<td>300</td>
<td>500</td>
<td>400</td>
<td>450</td>
<td>550</td>
<td>650</td>
</tr>
</tbody>
</table>

**Sample Initial Population**

Assume that random numbers generated to form the 1\(^{st}\) chromosome are as shown in Table 4.2. Since there are 12 Parts (Jobs), there are 12 random numbers generated (one for each product).
There are 4 cells in this problem. It is assumed that probability of assigning a product
to each cell is equal and therefore $1/4 = 0.25$. Then the intervals for cell assignment are
determined as shown in Table 4.3.

`Table 4.2. Random Numbers for Chromosome 1.

<table>
<thead>
<tr>
<th>Parts</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
<th>P9</th>
<th>P10</th>
<th>P11</th>
<th>P12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Numbers</td>
<td>0.12</td>
<td>0.35</td>
<td>0.66</td>
<td>0.70</td>
<td>0.89</td>
<td>0.40</td>
<td>0.29</td>
<td>0.46</td>
<td>0.74</td>
<td>0.81</td>
<td>0.55</td>
<td>0.42</td>
</tr>
</tbody>
</table>

Table 4.3. Cumulative Probabilities of Cell Assignment.

<table>
<thead>
<tr>
<th>Interval Number</th>
<th>Probability Interval</th>
<th>Assigned Cell</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>0-0.25</td>
<td>Cell 1</td>
</tr>
<tr>
<td>II</td>
<td>0.251-0.50</td>
<td>Cell 2</td>
</tr>
<tr>
<td>III</td>
<td>0.501-0.75</td>
<td>Cell 3</td>
</tr>
<tr>
<td>IV</td>
<td>0.751-1.0</td>
<td>Cell 4</td>
</tr>
</tbody>
</table>

The random numbers generated are then used to determine the cell assignments by
using the intervals. The chromosome thus obtained is given in Figure 4.4.

Figure 4.4. Chromosome 1 in Coded Form
When the encoded solution in Figure 4.4 is decoded, part-cell assignment is revealed as shown in Table 4.4 for Chromosome 1.

<table>
<thead>
<tr>
<th>Cells</th>
<th>Parts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell 1</td>
<td>Part 1</td>
</tr>
<tr>
<td>Cell 2</td>
<td>Part 2</td>
</tr>
<tr>
<td>Cell 3</td>
<td>Part 3</td>
</tr>
<tr>
<td>Cell 4</td>
<td>Part 5</td>
</tr>
</tbody>
</table>

### Sample Fitness Function

Once the parts assignments to cells have been determined, the next step is to calculate the fitness function of the chromosome. Calculation is done by following the steps in Section 4.1.1. For Chromosome 1, the value of the fitness function is calculated as follows. Processing times of parts are from Table 4.1.

- **Cell 1 has only part 1**
  
The total time for 15 workers will be $800 \leq 2400$
  
  This means that 15 workers are sufficient for Cell 1

- **Cell 2 has parts 2, 6, 7, 8, and 12**
  
The total time for 15 workers will be $1000 + 1000 + 600 + 800 + 1000 = 4400 > 2400$
  
The total time for 20 workers will be $750 + 750 + 400 + 600 + 800 = 3300 > 2400$
  
The total time for 25 workers will be $600 + 550 + 300 + 500 + 650 = 2600 > 2400$
Since even the biggest configuration, 25, is also not sufficient, PC will be applied to this cell. The cell configuration will be $25 + 100 = 125$

- **Cell 3** has parts 3, 4, 9 and 11

  The total time for 15 workers will be $1200 + 550 + 750 + 800 = 3300 > 2400$
  
  The total time for 20 workers will be $900 + 400 + 500 + 700 = 2500 > 2400$
  
  The total time for 25 workers will be $650 + 325 + 400 + 550 = 1925 < 2400$

  This means that 25 Workers are needed for Cell 3

- **Cell 4** has parts 5 and 10

  The total time for 15 workers will be $650 + 600 = 1250 < 2400$

  This means that 15 Workers are sufficient for Cell 4

As a result, the fitness function for Chromosome 1 is computed by adding worker requirements for all cells.

$$FF_1 = 15 + 125 + 25 + 15 = 180$$

*Sample Reproduction*

For sample reproduction, it is assumed that there are four chromosomes and fitness functions of these chromosomes are as in Table 4.5.
The reproduction probabilities of the chromosomes should be calculated to form the new generation. The steps explained in Section 4.1.1 are followed for determining the reproduction probabilities.

\[
TFF = 80 + 65 + 70 + 75 = 290
\]

\[
AF_1 = \frac{290}{80} = 3.63
\]
\[
AF_2 = \frac{290}{65} = 4.46
\]
\[
AF_3 = \frac{290}{70} = 4.14
\]
\[
AF_4 = \frac{290}{75} = 3.87
\]

\[
TAF = AF_1 + AF_2 + AF_3 + AF_4 = 3.63 + 4.46 + 4.12 + 3.88 = 16.10
\]

The reproduction probabilities are calculated as:

\[
P_1 = \frac{3.63}{16.10} = 0.225
\]
\[
P_2 = \frac{4.46}{16.10} = 0.277
\]
\[ P_3 = \frac{4.12}{16.10} = 0.256 \]
\[ P_4 = \frac{3.88}{16.10} = 0.242 \]

Random numbers will be generated to select the chromosomes according to the cumulative reproduction probabilities given in Table 4.6. By assuming random numbers of 0.154, 0.773, 0.435, and 0.654, chromosomes of next generation are determined to be Chromosome 1, Chromosome 4, Chromosome 2, and Chromosome 3, respectively. In other words, all the current chromosomes pass to the next generation.

**Table 4.6. Cumulative Probabilities of Reproduction.**

<table>
<thead>
<tr>
<th>Interval Number</th>
<th>Probability Interval</th>
<th>Reproduced Chromosome</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>0-0.225</td>
<td>C1</td>
</tr>
<tr>
<td>II</td>
<td>0.2251-0.502</td>
<td>C2</td>
</tr>
<tr>
<td>III</td>
<td>0.5021-0.758</td>
<td>C3</td>
</tr>
<tr>
<td>IV</td>
<td>0.7581-1.0</td>
<td>C4</td>
</tr>
</tbody>
</table>

**Sample Mating**

Since there are 4 chromosomes in this example, the probability of mating one chromosome with one of the remaining ones is 0.33. The probabilities of mating C1 with C2, C1 with C3 or C1 with C4 are equal \((1/3 = 0.33)\) as shown in Table 4.7. Generated random number is, say, 0.50. As a result of this step, C1 will mate C3 and which enforces C2 and C4 mating since they are the only ones left.
Table 4.7. Cumulative Probabilities of Mating of C1.

<table>
<thead>
<tr>
<th>Chromosome Mating Alternatives</th>
<th>Interval Probabilities</th>
<th>Generated Random Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1-C2</td>
<td>0 - 0.33</td>
<td></td>
</tr>
<tr>
<td>C1-C3</td>
<td>0.331 - 0.66</td>
<td>0.50</td>
</tr>
<tr>
<td>C1-C4</td>
<td>0.661 - 1.0</td>
<td></td>
</tr>
</tbody>
</table>

**Sample Crossover**

Since there are 12 parts, there will be 11 possible cut points for Single Cut Point Crossover and 12 possible cut points for Two Cut Points Crossover. Each of the cut point has equal probability so probability of cut points in Single Cut Point Crossover is 1/11 and in Two Cut Point Crossover is 1/12. For Two Cut Point Crossover, 2\textsuperscript{nd} cut point has 11 alternatives if the 1\textsuperscript{st} cut point is different than 1\textsuperscript{st} gene and last gene, 10 alternatives otherwise.

First a random number is generated; the cut point or points are found and crossover operation is done as shown in Figure 4.2 and Figure 4.3 for Single Cut Point Crossover and Two Cut Points Crossover, respectively.

**Sample Mutation**

To give an example how the Random Mutation operation works, mutation is applied to C1 which is shown in Figure 4.4. The probability of mutation is assumed to be 0.10. Let’s say the random number generated for Gene 2 is 0.06. Since this number is smaller than mutation probability, this gene is mutated.
Mutation in this problem means assigning this product to a new cell instead of the current cell which is Cell-2. To find which cell this product will be assigned, another random number is generated. Since there are 4 cells in this particular example, the probability is equal and \(1/3 = 0.33\). Since the product was assigned to Cell 2, the remaining cells are Cell 1, Cell 3 and Cell 4. Cumulative probabilities of alternative cell assignments are shown in Table 4.8.

Table 4.8. Cumulative Cell Alternatives for a Mutated Gene.

<table>
<thead>
<tr>
<th>Cells</th>
<th>Cumulative Probabilities</th>
<th>Generated Random Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell 1</td>
<td>0-0.33</td>
<td></td>
</tr>
<tr>
<td>Cell 3</td>
<td>0.331-0.66</td>
<td>0.63</td>
</tr>
<tr>
<td>Cell 4</td>
<td>0.661-1.0</td>
<td></td>
</tr>
</tbody>
</table>

The random number is in the interval for Cell 3. This means Part 2 is removed from Cell 2 and then assigned to Cell 3. Arbitrary Two Product Change Mutation is the same with Random Mutation except that instead of checking each gene’s mutation probability, only mutation probability of the chromosome is checked. If the random number is below mutation probability, then the chromosome is definitely mutated by interchanging two genes in the chromosome. The two genes that are going to be mutated are determined exactly the same way the two cut points are determined in Two Cut Point Crossover operator. This can be found in Subsection Two-Cut Point Crossover given in Section 4.1.1.
**Sample Termination**

After mutation is applied to all chromosomes, GA will go back to reproduction. This means the whole genetic cycle is going to repeat until the termination criterion is satisfied. Termination criteria are explained in Section 4.1.8.

### 4.2 Original Aspect

The concepts discussed in Section 4.1 are well known subjects in GA literature. Although applying these concepts to MACL problem and developing GA software to experiment with this problem are important contributions to GA subject, a new approach in GA has been searched for in this study. Finally, this new approach became solid with the idea of Multiple League (ML) and Extreme League (EL) concepts. Traditional GA applications solve problems with one population which is also referred as Single League (SL) in this study. In one population random search takes place in the early stages of the GA search whereas improvement of the solutions takes place in the later stages of the GA search [4]. Dividing the population into more than one population (leagues) might have certain advantages over a single population is the driving force of this new proposed approach.

#### 4.2.1 Multiple League

When life is analyzed, it is seen that there is a classification in almost every area from cars to medicines. Sport is one of them and one of the most convenient area to explain this approach. In every sport, there are different leagues. The idea behind is to gather the teams of same strength in one league so that challenging teams will push each
other to be better instead of loosing motivation in the case of dominating teams. The same idea can work with chromosomes also. If good chromosomes mate (play) with other good chromosomes, their offspring may also be good and hopefully even better. The average and less good chromosomes will also be treated the same way. That is to say, they will also have league of their own and try to produce better offspring. Furthermore there will be seasons and just like in real sport, at the end of the season best team (chromosome) or teams will upgrade to the higher level league and worst team or teams will drop to lower level league. Mutation in a chromosome can be considered as transferring a new player for an old player. Whether ML idea works or not is checked in Chapter 5. New terms and calculations are revealed with ML idea. The terms are summarized in Table 4.9 and calculations are summarized in Table 4.10.

<table>
<thead>
<tr>
<th>Single League Term</th>
<th>Multiple League Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome</td>
<td>Team</td>
</tr>
<tr>
<td>Mate</td>
<td>Play / Game</td>
</tr>
<tr>
<td>Mutation</td>
<td>Transfer a new Player</td>
</tr>
<tr>
<td>A Certain Number of Generations</td>
<td>Season</td>
</tr>
<tr>
<td>Population</td>
<td>League</td>
</tr>
</tbody>
</table>
Table 4.10. SL Calculation vs. ML Calculations.

<table>
<thead>
<tr>
<th>Calculation of…</th>
<th>Single League</th>
<th>Multiple League</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Number of Population</td>
<td>Population Size</td>
<td>Sum of Population Sizes of All Leagues</td>
</tr>
<tr>
<td>Total Number of Generations</td>
<td>Number of Generation</td>
<td>Number of Generations times Number of Seasons</td>
</tr>
<tr>
<td>Total Number of Problems Solved</td>
<td>Total Number of Population times Total Number of Generation of SL</td>
<td>Total Number of Population times Total Number of Generation of ML</td>
</tr>
</tbody>
</table>

Graphic representation of a ML consisting of 3 leagues is shown in Figure 4.5. It should be noted that there are no teams thrown away from GA in any league. The Second and Third League send their best teams to one upper league. On the other hand, the First and Second League send their worst teams to one lower league.

![Figure 4.5. Graphical Representation of ML with 3 Leagues](image)
4.2.2 Extreme League

After ML concept has been added to the GA algorithm, another idea was a small extension of ML which is named as the Extreme League (EL). The motivation was to have an EL which is totally different from all other leagues in terms of the games played between the teams. Games which are crossover and mutation operators in SL should have extraordinary rules (values) which are crossover and mutation probabilities in SL. As a result, an EL is created as an option in ML whose crossover and mutation probabilities can be determined separately. Furthermore, EL’s initial population is formed without ranking and grouping like the other leagues to keep the randomness in the league. In other words, since this league is designed to have different properties than the other leagues, its population is formed randomly which may consist of both good and bad teams. Moreover, EL does not get any team(s) from other leagues at the end of the season. In short, it is a league which is formed randomly and has extreme crossover and mutation probabilities and is the only league that keeps its teams throughout all generations. Although good teams of EL go to the last league of regular leagues (Third League in this case), EL still keeps copies of those teams. Another difference is last league also loses its worst teams since it gets teams from EL. A graphical representation of EL together with other regular leagues is given in Figure 4.6.
4.3 Genetic Algorithm Software – MACL-6.0

GA operations require mathematical calculations. Although these calculations are simple mathematics, software is required to perform an experiment about GAs because there are many calculations. The software developed to test MACL problem with GA is written in Visual Basic which is explained in Section 2.4. It is an Event-Based Program with many Functions (called MACL-6.0). Many new features have been added to the
program since the beginning and finally 6th Version is reached, MACL-6.0. Briefly, there 
are four forms in the program. The forms are used for both user interface features like 
Command Buttons, Pictures, etc. and code of the program. Form 1 (DataInput.frm) is 
Parameter Determination page and gets most of the parameters about MACL problem 
and GA. This form also has the longest code which contains almost all the functions. 
Form2 (DataOutput.frm) is Results and Statistics page which shows the generation 
number for the best solution, worst result and average. It also gives 2nd and 3rd best results 
with their corresponding generation numbers. Furthermore, there are command buttons 
like ‘re-Run’ and ‘Additional Run’ which are explained in Table 4.11. The Form 3 
(League.frm) is Multiple League Parameters page where parameters about ML are input. 
Finally, the Form 4 (ExtremeLeague.frm) is a special type called Dialog Box which is 
similar to a normal form but includes OK and Cancel Command buttons by default. The 
Form 4 is Extreme League Parameters page to input crossover and mutation probabilities 
for EL.

All the Events (Command Buttons) used in this program are explained in Table 4.11.
Table 4.11. MACL-6.0 Events and Their Explanations

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Place (Form #)</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next Cell Probability</td>
<td>Form 1</td>
<td>Assigns Product-Cell Assignment Probability</td>
</tr>
<tr>
<td>Next Configuration Demands</td>
<td>Form 1</td>
<td>Assigns both Worker # and corresponding Processing Times of Products</td>
</tr>
<tr>
<td>Start</td>
<td>Forms 1 &amp; 3</td>
<td>Starts the GA Cycle until the Termination Criterion is met</td>
</tr>
<tr>
<td>Parameter Page</td>
<td>Form 2</td>
<td>Goes to Form 1 from Form 2 to change parameters</td>
</tr>
<tr>
<td>re-Run</td>
<td>Form 2</td>
<td>Reruns with the same parameters but a different population. Statistics are initialized</td>
</tr>
<tr>
<td>Additional Run</td>
<td>Form 2</td>
<td>Continues with the last population and runs more generations as specified. Old statistics are kept</td>
</tr>
<tr>
<td>Save Excel Doc.</td>
<td>Form 2</td>
<td>Saves the parameters and best fitness functions to an excel document</td>
</tr>
<tr>
<td>OK</td>
<td>Form 4</td>
<td>Crossover and Mutation Probabilities for EL are determined</td>
</tr>
<tr>
<td>Cancel</td>
<td>Forms 3 &amp; 4</td>
<td>For Form 3, ML and for Form 4, EL option is not used in GA</td>
</tr>
<tr>
<td>Exit</td>
<td>Forms 1 &amp; 2</td>
<td>Program terminates</td>
</tr>
</tbody>
</table>

One of the main properties of the program is its flexibility. Almost every decision about GA parameters and MACL problem parameters is given by the user. Parameters for MACL problem can be seen in a snapshot of MACL-6.0 Parameter Determination page which is presented in Figure 4.7. Only ‘Number of Cells’, ‘Number of Cell Configurations’, ‘Number of Products’, and ‘Cell Conf.’ are related to MACL problem. The rest is GA parameters for MACL-6.0 program. While ‘Number of Cell Configurations’ and ‘Cell Conf.’ is totally flexible, Number of Cells can be up to 13 and Number of Products can go up to 100. These constraints are mainly because of space
limitations at user-friendly interface but since these numbers would be enough for most of the even big problems, this does not really limit the program.

Figure 4.7. Parameter Determination Page of MACL-6.0 Program
Parameters of GA are given in snapshots of MACL-6.0 Parameter Determination, Multiple League Parameters, and Extreme League Parameters pages which are presented in Figure 4.7, 4.8, and 4.9, respectively. The parameters and strategies which are required by SL are explained in Section 4.1. ML concepts and parameters are explained in 4.2.1. Additional information is MLs can have the same size or different size leagues. If they have the same size league, then the number of leagues (# of Leagues) is totally flexible. If they are preferred to have different sizes then the number of leagues can be up to 16. Finally concepts and parameters about EL are discussed in Section 4.2.2.

Figure 4.8. Multiple League Parameters Page of MACL-6.0
MACL-6.0 program is flexible and has many options. As a result, the code is long. To handle the code, each task was done as a function. There are many functions. Briefly important functions are in geneticAlgorithmCycle() which is also a function that contains all the operations of GA. SL calls this function one time for each run whereas ML calls this function (Number of Leagues * Number of Seasons) times. The code for this function is given in Figure 4.10 to give an idea about the structure of the program. Other functions in geneticAlgorithmCycle() are explained in Table 4.12. All the crossover operators are defined as functions but Number of Generations option (Termination Criteria). The reason is number of generations is assigned to Counter ‘Y’ in the code and GA cycle ends when number of iterations equals to Y. This can be noted in the outmost loop in Figure 4.10.
Table 4.12. Explanation of Functions in geneticAlgorithmCycle() Function.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewPopulation</td>
<td>Does the Reproduction operation in GA. Selects new generation from old generation</td>
</tr>
<tr>
<td>cross_single</td>
<td>Does Crossover operation in GA when Crossover strategy is Single-Cut</td>
</tr>
<tr>
<td>cross_double</td>
<td>Does Crossover operation in GA when Crossover strategy is Two-Cut Points</td>
</tr>
<tr>
<td>mut_random</td>
<td>Does Mutation operation in GA when Mutation strategy is Random</td>
</tr>
<tr>
<td>mut_2part</td>
<td>Does Mutation operation in GA when Mutation strategy is Arbitrary Two Product Change</td>
</tr>
<tr>
<td>IterStatistic</td>
<td>Keeps the statistics of the current population, i.e., best fitness function</td>
</tr>
<tr>
<td>ThreasTermination</td>
<td>Checks termination Threshold Termination</td>
</tr>
</tbody>
</table>
Figure 4.10. Code of geneticAlgorithmCycle() Function in MACL-6.0
CHAPTER 5

ANALYSIS OF RESULTS

In this chapter, the GA software MACL-6.0 is tested with two different problems. The first problem is the one used in Chapter 4. The second problem is created randomly by using normal distribution. Not only comparison within SL and ML options but also comparison between SL and ML are done. Finally, GA solutions are compared with the solutions found by the MACL mathematical program given in Chapter 3.

5.1 Single League Operators and Parameters Comparison

MACL-6.0 has many different features. Comparisons among different features are done to see if one strategy outperforms the other. The problem originally given in Section 4.1.9 is used except that the number of cells is limited to three. Default values of the operators and parameters are shown in Table 5.1.

Table 5.1. Default Values for SL Comparison.

<table>
<thead>
<tr>
<th>Operator or Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crossover Operator</td>
<td>Single-Cut Crossover</td>
</tr>
<tr>
<td>Crossover Probability</td>
<td>0.80</td>
</tr>
<tr>
<td>Mutation Operator</td>
<td>Random Mutation</td>
</tr>
<tr>
<td>Mutation Probability</td>
<td>0.05</td>
</tr>
<tr>
<td>Termination Criteria</td>
<td>Number of Generations</td>
</tr>
</tbody>
</table>
All the comparisons are done at 8 different points. These points correspond to different population size, number of generations and hence total number of problems solved as given in Table 5.3. At each point, 10 replications have been made. Comparison criterion is how many optimal solutions have been found at that point. In other words, it is the frequency of the optimum. The optimal solution for this problem is 60 Workers. It was obtained with the MACL mathematical model given in Section 3.1.1. Since this is a small problem containing just 117 variables, OPL found the optimal solution quickly. There are more than one optimal solution to this problem. One of the optimal solutions is shown in Table 5.2.

Table 5.2. The Optimal Solution to MACL Problem.

<table>
<thead>
<tr>
<th>Cell</th>
<th>Configuration</th>
<th>Assigned Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell 1</td>
<td>25</td>
<td>Part-1 Part-2 Part-9 Part-10 Part-12</td>
</tr>
<tr>
<td>Cell 2</td>
<td>15</td>
<td>Part-5 Part-6 Part-11</td>
</tr>
<tr>
<td>Cell 3</td>
<td>20</td>
<td>Part-3 Part-4 Part-7 Part-8</td>
</tr>
</tbody>
</table>

Total number of problems solved can be formulized as:

\[
\text{Total Number of Problems Solved} = \text{Population Size} \times \text{Number of Generations}
\]
Table 5.3. Experimentation Cases

<table>
<thead>
<tr>
<th></th>
<th>Point-1</th>
<th>Point-2</th>
<th>Point-3</th>
<th>Point-4</th>
<th>Point-5</th>
<th>Point-6</th>
<th>Point-7</th>
<th>Point-8</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Population Size</strong></td>
<td>8</td>
<td>8</td>
<td>12</td>
<td>12</td>
<td>16</td>
<td>16</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td><strong>Number of Generations</strong></td>
<td>6</td>
<td>10</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>50</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td><strong>Total Problems Solved</strong></td>
<td>48</td>
<td>80</td>
<td>120</td>
<td>240</td>
<td>320</td>
<td>800</td>
<td>1000</td>
<td>2000</td>
</tr>
</tbody>
</table>

5.1.1 Population Size vs. Number of Generations

This experimentation is performed by first keeping the population size constant at 8 and increasing the number of generations so that the total number of problems solved will be equal to the points represented at Table 5.3 at each point. Then, number of generations is kept constant at 8 and population size increased at further points. The total number of problems solved was again the same as the points in Table 5.3. The results are shown in Figure 5.1.
As seen from Figure 5.1, the constant number of generation (or increasing population size) outperforms or equally performs constant population size (or increasing number of generations) option except at Point-7. This means unless the total number of problems solved is high enough to find the optimum (like 1000), higher population size is a better option than the higher generation number to find the optimum.

As expected, as the total number of problems solved increased, and the frequency of optimum increased as well. The increase in frequency of optimum was consistent for the constant population size whereas the constant number of generations did not show that consistency due to Point-2 and Point-7.
5.1.2 Single-Cut Crossover vs. Two-Cut Points Crossover

Two populations are run; one with the Single-Cut Crossover operator and the other one with the Two-Cut Points Crossover. The results are presented in Figure 5.2.

![Graph showing comparison between Single-Cut Crossover and Two-Cut Points Crossover](image)

Figure 5.2. Comparison of Different Crossover Operators

The Single-Cut Crossover gives higher or same number of best solution at all points except Point-1. The total number of problems solved is lowest at Point-1 (48). Generally, GA converges to good solutions after certain number of iterations. As a result, randomness is more dominant at initial points. Two-Cut Points Crossover also drops to zero at Point-2. Moreover, as the total number of problems increases, generally GA finds the optimum more frequently.
5.1.3 Random Mutation vs. Arbitrary Two Product Change Mutation

This experimentation is done in a similar way. Two different mutation operators are used and then results are compared. The mutation probability for Random Mutation is per gene. On the other hand, the mutation probability for Arbitrary Two Product Change Mutation is per chromosome. As a result, mutation probability for Arbitrary Two Product Change Mutation is increased to a higher level which is 0.3. Otherwise, mutation probability would be too low for Arbitrary Two Product Change Mutation and the effect of this mutation would not be observed. The results are given in Figure 5.3.

![Random Mutation vs. Arbitrary Two Product Change Mutation](image)

Figure 5.3. Comparison of Different Mutation Operators
As seen in Figure 5.3, neither mutation operator outperforms the other except at the last two points. Sometimes one is better, sometimes the other, and sometimes they are equal. However, Arbitrary Two Product Change Mutation gives higher frequency of optimal solution at Point-7 and Point-8.

5.1.4 Low Crossover-Mutation Probability vs. High Crossover-Mutation Probability

The goal of this section is to observe the effect of crossover and mutation probabilities on the performance of GA. Two cases have been taken into consideration. GA is run with low probabilities in the first population and with high probabilities in the second population. Probabilities are shown in Table 5.4.

<table>
<thead>
<tr>
<th>Probability</th>
<th>Crossover</th>
<th>Mutation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>0.30</td>
<td>0.01</td>
</tr>
<tr>
<td>Medium</td>
<td>0.80</td>
<td>0.05</td>
</tr>
<tr>
<td>High</td>
<td>1.00</td>
<td>0.30</td>
</tr>
</tbody>
</table>

The results are presented in Figure 5.4. The Low Probability option shows a poor performance. Frequency of finding the optimum is less than or equal to 4 except at Point-8. Medium and High Probability options compete with each other. While the Medium Probability option is better than or equal to the High Probability option at the first three points, the High Probability option is better for the next three points. The Medium
Probability again gets better at Point-7 and at Point-8. Both options find the maximum frequency of finding the optimum. It can be concluded that Medium and High Probability options cannot outperform each other but they both outperform the Low Probability option.

Figure 5.4. Comparison of Different Crossover and Mutation Probabilities

5.2 Multiple League Operators and Parameters Comparison

ML has many features and many parameters that are determined by the user. Experimentation has been conducted to test the performance of these features and parameters. The same problem as in the previous section has been used. The comparisons
are done again at 8 points as described in Table 5.3. For each league of the ML, the default values of operators and parameters shown in Table 5.1 have been used. Double comparisons are done. The default operator and parameter values unique to ML are shown in Table 5.5.

Table 5.5. Default Values for ML Comparison

<table>
<thead>
<tr>
<th>Operator or Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extreme League</td>
<td>No Extreme League</td>
</tr>
<tr>
<td>Number of Leagues</td>
<td>2</td>
</tr>
<tr>
<td>Number of Seasons</td>
<td>1</td>
</tr>
<tr>
<td>Number of Changing Teams</td>
<td>1</td>
</tr>
</tbody>
</table>

5.2.1 With Extreme League vs. Without Extreme League

Both of the leagues that are compared are regular leagues with medium crossover and mutation probabilities, 0.80 and 0.05, respectively in ‘Without Extreme League Case’. In ‘With Extreme League Case’, the second league becomes the extreme league and has different crossover and mutation probabilities. The extreme league has been given high crossover and mutation probabilities, 1.00 and 0.30, respectively. The results of this comparison are shown in Figure 5.5. In both cases, two leagues share the total population equally. For example, at Point-1 total population size is 8. League-1 and League-2 (Extreme League in With Extreme League case) will have a population size of 4.
As seen from Figure 5.5, ‘Without Extreme League Case’ gave better or equal results at all points but Point-3. It can be concluded that with these parameters and for this problem, not using the extreme league option gives a higher number of optimal solutions.

5.2.2 ‘2 League’ vs. ‘3 League’

To see if the increase in number of leagues is more effective in finding the optimum, ‘2 League Case’ and ‘3 League Case’ have been tested. The total number of teams was attempted to be distributed evenly among the leagues. In other words, balanced population size was attempted. On the other hand, at some points different population size was inevitable. For example, since 8 total numbers of teams can not be distributed equally (as in ‘3 League’ case at Point-1), League-1 got 4 teams and League-2 and
League-3 each got 2 teams. In such situations, better leagues have been allocated to more teams. The results of this comparison can be observed in Figure 5.6.

Figure 5.6. Comparison of Different League Number

‘3 League Case’ has better or the same performance at all points except Point-3. For these conditions, ‘3 League Case’ outperforms ‘2 League Case’.

5.2.3 ‘1 Season’ vs. ‘2 Seasons’ vs. ‘Continuous Seasons’

In this case, the question is how number of seasons affect the ML performance. The number of seasons may be an important factor in ML performance because only at the end of seasons do better teams switch to better leagues. One of the motivations of the ML
concept was if better teams gather in the same league, the probability of having better offspring will be higher. The higher number of seasons means good teams will more often switch to better leagues and bad teams to lower leagues. Three cases have been studied in this section. In the first case, the number of seasons is 1 as in SL. In the second case, the number of seasons is 2 so teams switch among leagues one time. In the third case, the number of seasons is continuous which means number of generations is 1 and the total number of generations is equal to the number of seasons. In other words, there is team switching at the end of each generation. For example, at Point-1, the total number of generations is 6. The number of generations will be 1 and the number of seasons will be 6. Results of this comparison are presented in Figure 5.7.

![1 Season vs. 2 Seasons vs. Continuous Seasons](image)

Figure 5.7. Comparison of Different Number of Seasons
The results show that ‘1 Season’ is better than ‘Continuous Seasons’ and ‘Continuous Seasons’ is better than ‘2 Seasons’ case. Only at Point-4 does the ‘Continuous Seasons’ case give a higher frequency than ‘1 Season’ case. Likewise, at only Point-2 and Point-6 does the ‘2 Seasons’ case give a higher frequency than the ‘Continuous Seasons’ case. With these parameters and solutions, ‘1 Season’ case would be the first choice and ‘Continuous Seasons’ case the second.

5.2.4 1 Changing Team vs. 2 Changing Teams

At the end of seasons, better teams go to better leagues in ML. The Number of Changing Team determines how many teams will switch between consecutive leagues. ‘1 and 2 Changing Teams’ cases are tested. Results are shown in Figure 5.8.

Figure 5.8. Comparison of Different Number of Changing Teams
As seen from the figure, only at Point-4, does ‘1 Changing Team’ case find more optimum than the ‘2 Changing Teams’ case. It seems as more teams switch among consecutive leagues, better performance is observed.

5.3 Single League and Multiple League Comparison

One of the goals of this study was to test the performance of ML. In Section 5.1 better SL operators turned out to be the ones shown in Table 5.6. As a result, for all leagues of both SL and ML, Single-Cut Crossover is chosen as the crossover operator and Arbitrary Two Product Change is chosen as the mutation operator. There was a tie between Medium and High Crossover Probabilities and Medium Probability was chosen randomly as the crossover probability. Since Arbitrary Two Product Change Mutation is being used, its mutation probability is set to 0.30.

Table 5.6. Operators and Parameters Used for all leagues in SL and ML Comparison.

<table>
<thead>
<tr>
<th>Operator or Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crossover Operator</td>
<td>Single-Cut Crossover</td>
</tr>
<tr>
<td>Crossover Probability</td>
<td>Medium (0.80)</td>
</tr>
<tr>
<td>Mutation Operator</td>
<td>Arbitrary Two Product Change Mutation</td>
</tr>
<tr>
<td>Mutation Probability</td>
<td>0.30</td>
</tr>
</tbody>
</table>

Both SL and ML cases are again tested at the same 8 points shown in Table 5.3. This time, the number of replications was increased from 10 to 40 to observe any difference
clearer. Three cases of ML were compared with SL. Results are discussed in the following sections.

5.3.1 Case I: ML - Without EL, 3 Leagues, 1 Season, 2 Changing Teams

The reason of choosing these options for ML case is simply that they performed better than their alternatives when they were tested in Section 5.2. The results are presented in Figure 5.9. There is no data for Point-1 and Point-2 because with 3 Leagues and 2 Changing Teams, each league should have at least 4 teams which makes 12 teams whereas only 8 teams are available at these points.

Figure 5.9. SL and ML Comparison for Case I
As seen from the figure, SL performed better at every point. The closest that ML gets to SL is at Point-7 where SL found the optimum 38 times and ML found the optimum 35 out of 40 times. Furthermore, SL is stable because the frequency of optimum always increases as the total number of problems solved increases. On the other hand, ML is unstable because the frequency of optimum decreases from Point-4 to Point-5 and from Point-7 to Point-8. At Point-8, SL found optimum 39 times whereas ML found it only 34. For this point, it can be concluded that ML’s performance is 87% of SL.

5.3.2 Case II: ML - With EL, 3 Leagues, 2 Seasons, 1 Changing Team

In this case, ML has EL option with 0.50 crossover probability and 0.60 mutation probability. The crossover probability of EL is smaller and the mutation probability is higher than the previous experiment which is presented in Figure 5.5. The aim was to try a different parameter for EL and make it mutation dominated. The number of leagues is again 3. The Number of Seasons is increased to 2 and then number of Changing Teams is decreased to 1. The aim of these changes was again to try different parameters to see if ML will generate as good results as SL and maybe even better. The results are shown in Figure 5.10.
SL performs slightly worse than Case I whereas ML performs much better than Case I and even better than SL for the last 4 points except Point-7. Even at Point-7, both options are close to each other. ML found the optimum 33 times and SL found it 34. From these results ML can be suggested for last the 4 points which means that the total number of problems solved is greater than 240.

5.3.3 Case III: ML – With EL, 2 Leagues, Continuous Seasons, 2 Changing Teams

In this case, the number of leagues is decreased to 2 and the number of changing teams is increased to 2. Seasons are continuous as explained in the experimentation which is presented in Figure 5.7. EL’s crossover and mutation probabilities are same as in
Case II, 0.5 and 0.6, respectively. This combination is also new and results are shown in Figure 5.11.

Figure 5.11. SL and ML Comparison for Case III

ML outperforms SL. Only at Point-2 does SL perform better. Moreover, ML shows the best performance compared to ML performances in all 3 cases. Furthermore, ML gives the best results for the last 3 points. Since the frequency of finding the optimal solution is higher at these 3 points, this result becomes more crucial.
5.4 Genetic Algorithm vs. Mathematical Model (MACL)

Mathematical models solve small MACL problems in seconds. Whereas when problem gets bigger as is mostly the case in real life, execution time increases drastically. An MACL problem with 50 products, 6 Cell Configurations, and 7 Cells is created to compare the performance of GA and Mathematical Models. Processing times of products of this problem are given in a table in APPENDIX D. These times are generated according to the Normal Distribution for each configuration. The mean and standard deviation of each configuration is also given in the same table. Since GA in Section 5.3.3 had the best performance among the cases tested, the same operators and parameters have been used.

The total number of problems solved is increased as presented in Table 5.7. Product-Cell assignment probability for each cell is equal (1/7). Both mathematical program and GA are run at the scenarios shown in Table 5.8. For each scenario, GA was run with 5 replications. For Scenario-1, the 35 Products used are the first 35 Products of all 50 Products. Likewise, for Scenario-2, the 45 Products used are the first 45 products of 50 products. The results of this comparison are also presented in Table 5.8.

<table>
<thead>
<tr>
<th>Leagues</th>
<th>Population Size</th>
<th>Number of Generations</th>
</tr>
</thead>
<tbody>
<tr>
<td>League-1</td>
<td>60</td>
<td>1500</td>
</tr>
<tr>
<td>Extreme League</td>
<td>40</td>
<td></td>
</tr>
</tbody>
</table>
Table 5.8. Scenarios and Results for GA and Mathematical Model Comparison.

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Mathematical Model</th>
<th>Genetic Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scenario-1</strong> 35 P 6 Co 7 C</td>
<td>Best Optimality 120 Yes 2min</td>
<td>R1 135 R2 135 R3 135 R4 135 R5 135 Time 5min</td>
</tr>
<tr>
<td><strong>Scenario-2</strong> 45 P 6 Co 7 C</td>
<td>Best Optimality 160 Yes 3min</td>
<td>R1 175 R2 175 R3 175 R4 175 R5 175 Time 7min</td>
</tr>
<tr>
<td><strong>Scenario-3</strong> 50 P 6 Co 7 C</td>
<td>Best Optimality 180 No 31hrs</td>
<td>R1 185 R2 190 R3 185 R4 190 R5 190 Time 10min</td>
</tr>
</tbody>
</table>

In Scenario-1, the Mathematical Model finds the optimum in short execution time. GA finds 135 Worker solution in all 5 replications. It takes GA 3 minutes more to find this solution than the mathematical model. GA’s solution is 12.5% worse than the mathematical model solution. In Scenario-2, GA finds 175 Worker solution in 7 minutes for all 5 replications. This is 9.4% worse than the optimal solution found by the mathematical model in 3 minutes. For the first two scenarios, the mathematical model (MACL model) outperforms GA at both execution time and solution. One important point is GA is consistent for all replications at both scenarios. At Scenario-3, the mathematical model cannot find the optimum solution. After 31 hours of running the program, the best solution that the mathematical program found was 180. GA could not find the best result of Mathematical Model but found close solutions. The average solution of GA for 5 replications is 188 which is just 4.5% higher than the mathematical model’s best solution. Moreover, it took only 10 minutes for GA to find these solutions which is less than 1% of the execution time of the mathematical model.
CHAPTER 6

CONCLUSION AND FUTURE RESEARCH

In this chapter, conclusions of experimentation with the Extended Models, GA and MACL model are summarized and then future work is discussed.

6.1 Conclusion

Extended Models are derived from the MACL model to include the number of tardy jobs concept. The experimentation with Extended Models demonstrated that as the total manpower increases, the number of tardy jobs decreases. The comparison between Extended Models demonstrated that System III (the Lot-Splitting Allowed case), gave the best results while System II (the Common Cell case) gave the worst results. Since zero set-up times are assumed, lot-splitting had an advantage over other cases. The Common Cell case had an extra constraint which made it less flexible in terms of cell configurations. Finally, cost experimentation with System I demonstrated that total cost depends on the manpower costs and tardiness costs assigned.

Experimentation with a GA program for SL showed that as the total number of problems solved increases, the frequency of finding optimum also increases. When we compared the effectiveness of population size and the number of generations, population size turned out to be more effective than the number of generations. Similarly, Single-Cut Crossover is more effective than Two-Cut Point Crossover. Arbitrary Two Product
Change Mutation is better than Random Mutation and finally, middle or high crossover-mutation probability is better than low crossover-mutation probability.

Experimentation with GA program for ML displayed that ‘without EL’ option is more effective than ‘with EL’. ‘3 League’ gave better results than ‘2 League’. ‘1 Season’ and ‘Continuous Seasons’ had higher performance than the ‘2 Seasons’ option. Finally, the ‘2 Changing Teams’ is better than the ‘1 Changing Team’ option.

The ML method was compared with the SL method. When ML is run without EL, 3 Leagues, 1 Season, and 2 Changing Teams, SL performed better than ML. When ML options changed to ‘with EL’, ‘2 Leagues’, ‘Continuous Seasons’, and ‘2 Changing Teams’, it outperformed SL. ML without EL, 3 Leagues, 2 Seasons, and 1 Changing Team had similar performance with SL. In short, the ML approach can outperform the traditional approach. This is an important outcome of this study.

Experimentation comparing GA and the Mathematical Model (MACL) showed that the Mathematical Model is appropriate for small problems. It finds the optimal solutions in short execution times. On the other hand, as the problem gets bigger, GA becomes more effective. It finds near optimal solutions in reasonable times for a manufacturing setting.

6.2 Future Research

New mathematical models can be developed by considering different weights for products. There could also be preferred cells for each product. Feasible cell concept for a product will allow mathematical models to apply cellular manufacturing more accurately. Lot-Splitting case can be analyzed with non-zero set-up times or the Common Cell Size.
concept. Another option would be Lot-Splitting case with Common Cell and Complete Job concepts. Finally, the models can be extended to include multi-period in which case operator learning can be added to models.

MACL-6.0 is flexible. There are many parameters that are user-defined and many GA operator alternatives. Experimentation in this study revealed the importance of GA as an optimization technique. Furthermore, the ML option can have a better performance than the SL option. There are still many parameters and operator alternatives to be tested. Much more experimentation is possible with MACL-6.0. Different parameter and operator combinations can be tested with different problems. Multiple runs of GA with the same parameters and operators will reveal the consistency level of GA. Only then can more general conclusions be reached.

ML concept can be improved by adding another option in which better teams and worse teams switch not only among consecutive leagues but among all leagues according to their performance. Furthermore, the league from which the best team is formed can be presented in the detailed solution. This will help the user to judge the difference between leagues better. Finally, Elitism (keeping the best chromosome in the new generation) can be added as a selection operator to GA.
REFERENCES


APPENDIX A

MATHEMATICAL PROGRAMS OF EXTENDED MODELS

Model II – Common Cell Size

```c
float daySectTime = ...;
float avCapacity = ...;
int totMan = ...;

int nbProducts = ...;
int nbCells = ...;
int nbConf = ...;

range Products 1..nbProducts,
    Cells 1..nbCells,
    Conf 1..nbConf,
    boolean 0..1;

struct ProductData {
    float dueDate;
};

int workerOpt[Conf] = ...;
float processTime[Products, Conf] = ...;
ProductData product[Products] = ...;

var
    boolean clDecVar[Cells, Conf],
    boolean prDecVar[Products, Cells, Conf];

maximize
    sum(p in Products & c in Cells & co in Conf) (prDecVar[p,c,co])
subject to {
    forall(p in Products)
        sum(c in Cells & co in Conf)
            prDecVar[p,c,co] <= 1;

    forall(c1 in Cells & c2 in Cells & c3 in Cells)
        if c1 <> c2 & c1 <> c3 & c2 <> c3 then
            c1DecVar[c1,1] + c1DecVar[c2,2] + c1DecVar[c3,3] <= 1
        end if;

    forall(c in Cells & co in Conf)
        sum(p in Products)
            processTime[p,co]*prDecVar[p,c,co] <= avCapacity*clDecVar[c,co];

    forall(i in 1..nbProducts & c in Cells & co in Conf)
        sum ( j in 1..i )
            processTime[i,j]*prDecVar[j,c,co] <= product[i].dueDate*daySectTime*clDecVar[c,co];

    sum (c in Cells & co in Conf)
        workerOpt[co] = clDecVar[c,co] <= totMan;
}
```
Model III- Lot-Splitting Allowed

float daySectTime = ...;
float avCapacity = ...;
int totMan = ...;

int nbProducts = ...;
int nbCells = ...;
int nbConf = ...;

range
  Products 1..nbProducts,
  Cells 1..nbCells,
  Conf 1..nbConf,
  boolean 0..1,
  float flBoolean 0..1;

struct ProductData {
  float dueDate;
};

int workerOpt[Conf] = ...;
float processTime[Products, Conf] = ...;
ProductData product[Products] = ...;

var
  boolean clDecVar[Cells, Conf],
  flBoolean prDecVar[Products, Cells, Conf];

maximize
  sum(p in Products & c in Cells & co in Conf)(prDecVar[p, c, co])
subject to {
  forall(p in Products)
    sum(c in Cells & co in Conf)
      prDecVar[p, c, co] <= 1;

  forall(c in Cells)
    sum(co in Conf)
      clDecVar[c, co] <= 1;

  forall(c in Cells & co in Conf)
    sum(p in Products)
      processTime[p, co] * prDecVar[p, c, co] <= avCapacity * clDecVar[c, co];

  forall(i in 1..nbProducts & c in Cells & co in Conf)
    sum(j in 1..1)
      processTime[j, co] * prDecVar[j, c, co] <= product[i].dueDate * daySectTime * clDecVar[c, co];

  sum(c in Cells & co in Conf)
    workerOpt[co] = clDecVar[c, co] <= totMan;
};
Model IV- Lot-Splitting Allowed Complete Job

```c
float daySectTime = ...;
float auCapacity = ...;
int totMan = ...;

int nbProducts = ...;
int nbCells = ...;
int nbConf = ...;

range
  Products 1..nbProducts,
  Cells 1..nbCells,
  Conf 1..nbConf,
  boolean 0..1,
  float flBoolean 0..1;

struct ProductData {
  float dueDate;
};

int workerOpt[Conf] = ...;
float processTime[Products, Conf] = ...;
ProductData product[Products] = ...;

var
  boolean clDecVar[Cells, Conf],
  flBoolean prDecVar[Products, Cells, Conf],
  boolean prControl[Products];

maximize
  sum(p in Products & c in Cells & co in Conf)(prDecVar[p,c,co])

subject to {
  forall(p in Products)
    sum(c in Cells & co in Conf)
      prDecVar[p,c,co] = prControl[p];

  forall(c in Cells)
    sum(co in Conf)
      clDecVar[c,co] <= 1;

  forall(c in Cells & co in Conf)
    sum(p in Products)
      processTime[p,co]*prDecVar[p,c,co] <= auCapacity*clDecVar[c,co];

  forall(i in 1..nbProducts & c in Cells & co in Conf)
    sum(j in 1..1)
      processTime[j,co]*prDecVar[j,c,co] <= product[i].dueDate*daySectTime*clDecVar[c,co];

  sum(c in Cells & co in Conf)
    workerOpt[co] = clDecVar[c,co] <= totMan;

};
```
APPENDIX B

SAMPLE MODEL FILES FOR EXTENDED MODELS

Model I

range boolean 0..1;

//cell yes-no variables

var boolean y11;
var boolean y12;
var boolean y21;
var boolean y22;
var boolean y31;
var boolean y32;

//product yes-no variables

var boolean x111;
var boolean x112;
var boolean x121;
var boolean x122;
var boolean x131;
var boolean x132;
var boolean x211;
var boolean x212;
var boolean x221;
var boolean x222;
var boolean x231;
var boolean x232;
var boolean x311;
var boolean x312;
var boolean x321;
var boolean x322;
var boolean x331;
var boolean x332;
var boolean x411;
var boolean x412;
var boolean x421;
var boolean x422;
var boolean x431;
var boolean x432;

//worker number parameters

int b1 = 15;
int b2 = 20;
//cell time available capacity parameter

int dailyAvailableTime = 160; // in minutes
float avCapacity = 2400; // in minutes for a week

//due dates

int d1 = 3*dailyAvailableTime;
int d3 = 5*dailyAvailableTime;
int d2 = 6*dailyAvailableTime;
int d4 = 7*dailyAvailableTime;

//product time parameters

float p11 = 800;
float p12 = 600;
float p21 = 1000;
float p22 = 750;
float p31 = 1200;
float p32 = 900;
float p41 = 550;
float p42 = 400;

// Objective Function variable

var int+ earlyJobs in 1..maxint;

// Objective Function

maximize earlyJobs

subject to

// Constraint Set

{

// Objective function definition

earlyJobs = x111 + x112 + x121 + x122 + x131 + x132 +
x211 + x212 + x221 + x222 + x231 + x232 +
x311 + x312 + x321 + x322 + x331 + x332 +
x411 + x412 + x421 + x422 + x431 + x432;

// All the products can be assigned only to one cell.

x111 + x112 + x121 + x122 + x131 + x132 <= 1;
x211 + x212 + x221 + x222 + x231 + x232 <= 1;
\begin{align*}
x_{311} + x_{312} + x_{321} + x_{322} + x_{331} + x_{332} & \leq 1; \\
x_{411} + x_{412} + x_{421} + x_{422} + x_{431} + x_{432} & \leq 1; \\

// Each cell can only have max one configuration
\begin{align*}
y_{11} + y_{12} & \leq 1; \\
y_{21} + y_{22} & \leq 1; \\
y_{31} + y_{32} & \leq 1; \\

// EDD math model
\begin{align*}
p_{11} x_{111} & \leq d_1 y_{11}; \\
p_{12} x_{112} & \leq d_1 y_{12}; \\
p_{11} x_{121} & \leq d_1 y_{21}; \\
p_{12} x_{122} & \leq d_1 y_{22}; \\
p_{11} x_{131} & \leq d_1 y_{31}; \\
p_{12} x_{132} & \leq d_1 y_{32}; \\
p_{11} x_{111} + p_{31} x_{311} & \leq d_3 y_{11}; \\
p_{12} x_{112} + p_{32} x_{312} & \leq d_3 y_{12}; \\
p_{11} x_{121} + p_{31} x_{321} & \leq d_3 y_{21}; \\
p_{12} x_{122} + p_{32} x_{322} & \leq d_3 y_{22}; \\
p_{11} x_{131} + p_{31} x_{331} & \leq d_3 y_{31}; \\
p_{12} x_{132} + p_{32} x_{332} & \leq d_3 y_{32}; \\
p_{11} x_{111} + p_{31} x_{311} + p_{21} x_{211} & \leq d_2 y_{11}; \\
p_{12} x_{112} + p_{32} x_{312} + p_{22} x_{212} & \leq d_2 y_{12}; \\
p_{11} x_{121} + p_{31} x_{321} + p_{21} x_{221} & \leq d_2 y_{21}; \\
p_{12} x_{122} + p_{32} x_{322} + p_{22} x_{222} & \leq d_2 y_{22}; \\
p_{11} x_{131} + p_{31} x_{331} + p_{21} x_{231} & \leq d_2 y_{31}; \\
p_{12} x_{132} + p_{32} x_{332} + p_{22} x_{232} & \leq d_2 y_{32}; \\
p_{11} x_{111} + p_{31} x_{311} + p_{21} x_{211} + p_{41} x_{411} & \leq d_4 y_{11}; \\
p_{12} x_{112} + p_{32} x_{312} + p_{22} x_{212} + p_{42} x_{412} & \leq d_4 y_{12}; \\
p_{11} x_{121} + p_{31} x_{321} + p_{21} x_{221} + p_{41} x_{421} & \leq d_4 y_{21}; \\
p_{12} x_{122} + p_{32} x_{322} + p_{22} x_{222} + p_{42} x_{422} & \leq d_4 y_{22}; \\
p_{11} x_{131} + p_{31} x_{331} + p_{21} x_{231} + p_{41} x_{431} & \leq d_4 y_{31}; \\
p_{12} x_{132} + p_{32} x_{332} + p_{22} x_{232} + p_{42} x_{432} & \leq d_4 y_{32}; \\
p_{11} x_{111} + p_{21} x_{211} + p_{31} x_{311} + p_{41} x_{411} & \leq \text{avCapacity} y_{11}; \\
p_{12} x_{112} + p_{22} x_{212} + p_{32} x_{312} + p_{42} x_{412} & \leq \text{avCapacity} y_{12}; \\
p_{11} x_{121} + p_{21} x_{221} + p_{31} x_{321} + p_{41} x_{421} & \leq \text{avCapacity} y_{21}; \\
p_{12} x_{122} + p_{22} x_{222} + p_{32} x_{322} + p_{42} x_{422} & \leq \text{avCapacity} y_{22}; \\
p_{11} x_{131} + p_{21} x_{231} + p_{31} x_{331} + p_{41} x_{431} & \leq \text{avCapacity} y_{31}; \\
p_{12} x_{132} + p_{22} x_{232} + p_{32} x_{332} + p_{42} x_{432} & \leq \text{avCapacity} y_{32}; \\

// total number of workers constraint
\begin{align*}
b_1 y_{11} + b_2 y_{12} + b_1 y_{21} + b_2 y_{22} + b_1 y_{31} + b_2 y_{32} & \leq 40; \\
\end{align*}
\end{align*}
Model II

range boolean 0..1;

// cell yes-no variables
var boolean y11;
var boolean y12;
var boolean y21;
var boolean y22;
var boolean y31;
var boolean y32;

// product yes-no variables
var boolean x111;
var boolean x112;
var boolean x121;
var boolean x122;
var boolean x131;
var boolean x132;
var boolean x211;
var boolean x212;
var boolean x221;
var boolean x222;
var boolean x231;
var boolean x232;
var boolean x311;
var boolean x312;
var boolean x321;
var boolean x322;
var boolean x331;
var boolean x332;
var boolean x411;
var boolean x412;
var boolean x421;
var boolean x422;
var boolean x431;
var boolean x432;

// worker number parameters
int b1 = 15;
int b2 = 20;

// cell time available capacity parameter
int dailyAvailableTime = 160; // in minutes
float avCapacity = 2400; // in minutes for a week
// due dates
int d1 = 3*dailyAvailableTime;
int d3 = 5*dailyAvailableTime;
int d2 = 6*dailyAvailableTime;
int d4 = 7*dailyAvailableTime;

// product time parameters
float p11 = 800;
float p12 = 600;
float p21 = 1000;
float p22 = 750;
float p31 = 1200;
float p32 = 900;
float p41 = 550;
float p42 = 400;

// Objective Function variable
var int+ earlyJobs in 1..maxint;

// Objective Function
maximize earlyJobs

subject to

// Constraint Set
{

// Objective function definition
earlyJobs = x111 + x112 + x121 + x122 + x131 + x132 +
          x211 + x212 + x221 + x222 + x231 + x232 +
          x311 + x312 + x321 + x322 + x331 + x332 +
          x411 + x412 + x421 + x422 + x431 + x432;

// All the products can be assigned only to one cell.
x111 + x112 + x121 + x122 + x131 + x132 <= 1;
x211 + x212 + x221 + x222 + x231 + x232 <= 1;
x311 + x312 + x321 + x322 + x331 + x332 <= 1;
x411 + x412 + x421 + x422 + x431 + x432 <= 1;

// Each cell can only have max one configuration
y11 + y12 <= 1;
y11 + y22 <= 1;
\[ y_{11} + y_{32} \leq 1; \]
\[ y_{21} + y_{12} \leq 1; \]
\[ y_{21} + y_{22} \leq 1; \]
\[ y_{21} + y_{32} \leq 1; \]
\[ y_{31} + y_{12} \leq 1; \]
\[ y_{31} + y_{22} \leq 1; \]
\[ y_{31} + y_{32} \leq 1; \]

// EDD math model

\[ p_{11} x_{111} \leq d_1 y_{11}; \]
\[ p_{12} x_{112} \leq d_1 y_{12}; \]
\[ p_{11} x_{121} \leq d_1 y_{21}; \]
\[ p_{12} x_{122} \leq d_1 y_{22}; \]
\[ p_{11} x_{131} \leq d_1 y_{31}; \]
\[ p_{12} x_{132} \leq d_1 y_{32}; \]

\[ p_{11} x_{111} + p_{31} x_{311} \leq d_3 y_{11}; \]
\[ p_{12} x_{112} + p_{32} x_{312} \leq d_3 y_{12}; \]
\[ p_{11} x_{121} + p_{31} x_{321} \leq d_3 y_{21}; \]
\[ p_{12} x_{122} + p_{32} x_{322} \leq d_3 y_{22}; \]
\[ p_{11} x_{131} + p_{31} x_{331} \leq d_3 y_{31}; \]
\[ p_{12} x_{132} + p_{32} x_{332} \leq d_3 y_{32}; \]

\[ p_{11} x_{111} + p_{31} x_{311} + p_{21} x_{211} \leq d_2 y_{11}; \]
\[ p_{12} x_{112} + p_{32} x_{312} + p_{22} x_{212} \leq d_2 y_{12}; \]
\[ p_{11} x_{121} + p_{31} x_{321} + p_{21} x_{221} \leq d_2 y_{21}; \]
\[ p_{12} x_{122} + p_{32} x_{322} + p_{22} x_{222} \leq d_2 y_{22}; \]
\[ p_{11} x_{131} + p_{31} x_{331} + p_{21} x_{231} \leq d_2 y_{31}; \]
\[ p_{12} x_{132} + p_{32} x_{332} + p_{22} x_{232} \leq d_2 y_{32}; \]

\[ p_{11} x_{111} + p_{31} x_{311} + p_{21} x_{211} + p_{41} x_{411} \leq d_4 y_{11}; \]
\[ p_{12} x_{112} + p_{32} x_{312} + p_{22} x_{212} + p_{42} x_{412} \leq d_4 y_{12}; \]
\[ p_{11} x_{121} + p_{31} x_{321} + p_{21} x_{221} + p_{41} x_{421} \leq d_4 y_{21}; \]
\[ p_{12} x_{122} + p_{32} x_{322} + p_{22} x_{222} + p_{42} x_{422} \leq d_4 y_{22}; \]
\[ p_{11} x_{131} + p_{31} x_{331} + p_{21} x_{231} + p_{41} x_{431} \leq d_4 y_{31}; \]
\[ p_{12} x_{132} + p_{32} x_{332} + p_{22} x_{232} + p_{42} x_{432} \leq d_4 y_{32}; \]

\[ p_{11} x_{111} + p_{21} x_{211} + p_{31} x_{311} + p_{41} x_{411} \leq \text{avCapacity} y_{11}; \]
\[ p_{12} x_{112} + p_{22} x_{212} + p_{32} x_{312} + p_{42} x_{412} \leq \text{avCapacity} y_{12}; \]
\[ p_{11} x_{121} + p_{21} x_{221} + p_{31} x_{321} + p_{41} x_{421} \leq \text{avCapacity} y_{21}; \]
\[ p_{12} x_{122} + p_{22} x_{222} + p_{32} x_{322} + p_{42} x_{422} \leq \text{avCapacity} y_{22}; \]
\[ p_{11} x_{131} + p_{21} x_{231} + p_{31} x_{331} + p_{41} x_{431} \leq \text{avCapacity} y_{31}; \]
\[ p_{12} x_{132} + p_{22} x_{232} + p_{32} x_{332} + p_{42} x_{432} \leq \text{avCapacity} y_{32}; \]

// total number of workers constraint

\[ b_1 y_{11} + b_2 y_{12} + b_1 y_{21} + b_2 y_{22} + b_1 y_{31} + b_2 y_{32} \leq 40; \]

};
Model III

range boolean 0..1;
range float flNum 0..1;

//cell yes-no variables
var boolean y11;
var boolean y12;
var boolean y21;
var boolean y22;
var boolean y31;
var boolean y32;

//product yes-no variables
var flNum x111;
var flNum x112;
var flNum x121;
var flNum x122;
var flNum x131;
var flNum x132;
var flNum x211;
var flNum x212;
var flNum x221;
var flNum x222;
var flNum x231;
var flNum x232;
var flNum x311;
var flNum x312;
var flNum x321;
var flNum x322;
var flNum x331;
var flNum x332;
var flNum x411;
var flNum x412;
var flNum x421;
var flNum x422;
var flNum x431;
var flNum x432;

//worker number parameters
int b1 = 15;
int b2 = 20;

//cell time available capacity parameter
int dailyAvailableTime = 160; // in minutes
float avCapacity = 2400;      // in minutes for a week
//due dates
int d1 = 3*dailyAvailableTime;
int d3 = 5*dailyAvailableTime;
int d2 = 6*dailyAvailableTime;
int d4 = 7*dailyAvailableTime;

//product time parameters
float p11 = 800;
float p12 = 600;
float p21 = 1000;
float p22 = 750;
float p31 = 1200;
float p32 = 900;
float p41 = 550;
float p42 = 400;

// Objective Function variable
var float+ earlyJobs in 1..maxint;

// Objective Function
maximize earlyJobs

subject to

// Constraint Set
{

// Objective function definition
earlyJobs =x111+x112+x121+x122+x131+x132+
x211+x212+x221+x222+x231+x232+
x311+x312+x321+x322+x331+x332+
x411+x412+x421+x422+x431+x432;

// All the products can be assigned only to one cell.

x111+x112+x121+x122+x131+x132 <= 1;
x211+x212+x221+x222+x231+x232 <= 1;
x311+x312+x321+x322+x331+x332 <= 1;
x411+x412+x421+x422+x431+x432 <= 1;

// Each cell can only have max one configuration

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

// EDD math model

p11*x111 <= d1 * y11;
p12*x112 <= d1 * y12;
p11*x121 <= d1 * y21;
p12*x122 <= d1 * y22;
p11*x131 <= d1 * y31;
p12*x132 <= d1 * y32;

p11*x111 + p31*x311 <= d3 * y11;
p12*x112 + p32*x312 <= d3 * y12;
p11*x121 + p31*x321 <= d3 * y21;
p12*x122 + p32*x322 <= d3 * y22;
p11*x131 + p31*x331 <= d3 * y31;
p12*x132 + p32*x332 <= d3 * y32;

p11*x111 + p31*x311 + p21*x211 <= d2 * y11;
p12*x112 + p32*x312 + p22*x212 <= d2 * y12;
p11*x121 + p31*x321 + p21*x221 <= d2 * y21;
p12*x122 + p32*x322 + p22*x222 <= d2 * y22;
p11*x131 + p31*x331 + p21*x231 <= d2 * y31;
p12*x132 + p32*x332 + p22*x232 <= d2 * y32;

p11*x111 + p31*x311 + p21*x211 + p41*x411 <= d4 * y11;
p12*x112 + p32*x312 + p22*x212 + p42*x412 <= d4 * y12;
p11*x121 + p31*x321 + p21*x221 + p41*x421 <= d4 * y21;
p12*x122 + p32*x322 + p22*x222 + p42*x422 <= d4 * y22;
p11*x131 + p31*x331 + p21*x231 + p41*x431 <= d4 * y31;
p12*x132 + p32*x332 + p22*x232 + p42*x432 <= d4 * y32;

// total number of workers constraint

b1*y11 + b2*y12 + b1*y21 + b2*y22 + b1*y31 + b2*y32 <= 40;
Model IV

range boolean 0..1;
range float flNum 0..1;

//cell yes-no variables

var boolean y11;
var boolean y12;
var boolean y21;
var boolean y22;
var boolean y31;
var boolean y32;

//product yes-no variables

var flNum x111;
var flNum x112;
var flNum x121;
var flNum x122;
var flNum x131;
var flNum x132;
var flNum x211;
var flNum x212;
var flNum x221;
var flNum x222;
var flNum x231;
var flNum x232;
var flNum x311;
var flNum x312;
var flNum x321;
var flNum x322;
var flNum x331;
var flNum x332;
var flNum x411;
var flNum x412;
var flNum x421;
var flNum x422;
var flNum x431;
var flNum x432;

//control variables

var boolean t1;
var boolean t2;
var boolean t3;
var boolean t4;

//worker number parameters
int b1 = 15;
int b2 = 20;

//cell time available capacity parameter

int dailyAvailableTime = 160; // in minutes
float avCapacity = 2400;       // in minutes for a week

//due dates
int d1 =  3*dailyAvailableTime;
int d3 =  5*dailyAvailableTime;
int d2 =  6*dailyAvailableTime;
int d4 =  7*dailyAvailableTime;

//product time parameters
float p11 = 800;
float p12 = 600;
float p21 = 1000;
float p22 = 750;
float p31 = 1200;
float p32 = 900;
float p41 = 550;
float p42 = 400;

// Objective Function variable
var int+ earlyJobs in 1..maxint;

// Objective Function
maximize earlyJobs

subject to

// Constraint Set

{ }

// Objective function definition

earlyJobs =x111+x112+x121+x122+x131+x132+
x211+x212+x221+x222+x231+x232+
x311+x312+x321+x322+x331+x332+
x411+x412+x421+x422+x431+x432;

// All the products can be assigned only to one cell.

x111+x112+x121+x122+x131+x132 = t1;
x211+x212+x221+x222+x231+x232 = t2;
x311 + x312 + x321 + x322 + x331 + x332 = t3;
x411 + x412 + x421 + x422 + x431 + x432 = t4;

// Each cell can only have max one configuration

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

// EDD math model

p11 * x111 <= d1 * y11;
p12 * x112 <= d1 * y12;
p11 * x121 <= d1 * y31;
p12 * x122 <= d1 * y32;
p11 * x131 <= d1 * y31;
p12 * x132 <= d1 * y32;

p11 * x111 + p11 * x311 + p31 * x311 <= d3 * y11;
p12 * x112 + p32 * x312 <= d3 * y12;
p11 * x121 + p31 * x321 <= d3 * y31;
p12 * x122 + p32 * x322 <= d3 * y32;
p11 * x131 + p31 * x331 <= d3 * y31;
p12 * x132 + p32 * x332 <= d3 * y32;

p11 * x111 + p31 * x311 + p21 * x211 <= d2 * y11;
p12 * x112 + p32 * x312 + p22 * x212 <= d2 * y12;
p11 * x121 + p31 * x321 + p21 * x221 <= d2 * y31;
p12 * x122 + p32 * x322 + p22 * x222 <= d2 * y32;
p11 * x131 + p31 * x331 + p21 * x231 <= d2 * y31;
p12 * x132 + p32 * x332 + p22 * x232 <= d2 * y32;

p11 * x111 + p31 * x311 + p21 * x211 + p41 * x411 <= d4 * y11;
p12 * x112 + p32 * x312 + p22 * x212 + p42 * x412 <= d4 * y12;
p11 * x121 + p31 * x321 + p21 * x221 + p41 * x421 <= d4 * y31;
p12 * x122 + p32 * x322 + p22 * x222 + p42 * x422 <= d4 * y32;
p11 * x131 + p31 * x331 + p21 * x231 + p41 * x431 <= d4 * y31;
p12 * x132 + p32 * x332 + p22 * x232 + p42 * x432 <= d4 * y32;

p11 * x111 + p21 * x211 + p31 * x311 + p41 * x411 <= avCapacity * y11;
p12 * x112 + p22 * x212 + p32 * x312 + p42 * x412 <= avCapacity * y12;
p11 * x121 + p21 * x221 + p31 * x321 + p41 * x421 <= avCapacity * y31;
p12 * x122 + p22 * x222 + p32 * x322 + p42 * x422 <= avCapacity * y32;

// total number of workers constraint

b1 * y11 + b2 * y12 + b1 * y21 + b2 * y22 + b1 * y31 + b2 * y32 <= 40;
APPENDIX C

TABLES FOR MODEL I TARDINESS CALCULATION

Table 1. Total Number of Workers = 15.

<table>
<thead>
<tr>
<th>Tot No of Workers</th>
<th>Configuration</th>
<th>Products (i-j-th Sequence)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Proc. Times (Days)</td>
<td>1.35 1.15 1.25 1.25 1.56 1.67 1.67 1.67 1.67 2.08 2.08 2.08 2.50</td>
</tr>
<tr>
<td>15</td>
<td>Cumm. Proc Times</td>
<td>1.35 2.50 3.75 5.00 6.56 8.23 9.90 11.56 13.65 15.73 17.81 20.31</td>
</tr>
<tr>
<td>15</td>
<td>Due Dates</td>
<td>2.33 4.00 4.67 5.00 5.00 1.00 3.33 4.33 2.00 4.67 3.00 1.67</td>
</tr>
<tr>
<td>15</td>
<td>Tardiness</td>
<td>0.00 0.00 0.00 0.00 1.56 7.23 6.56 7.23 11.65 11.06 14.81 18.65</td>
</tr>
</tbody>
</table>
Table 2. Total Number of Workers = 20.

<table>
<thead>
<tr>
<th>Tot No of Workers</th>
<th>Configuration</th>
<th>Products (i-th Sequence)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Proc. Times (Days)</td>
<td>0.94  0.83  0.83  1.04  1.04  1.25  1.25  1.46  1.56  1.56  1.67  1.88</td>
</tr>
<tr>
<td></td>
<td>Cumm. Proc Times</td>
<td>0.94  1.77  2.60  3.65  4.69  5.94  7.19  8.65 10.21 11.77 13.44 15.31</td>
</tr>
<tr>
<td></td>
<td>Due Dates</td>
<td>2.33  4.00  4.67  5.00  5.00  1.00  3.33  4.33  2.00  4.67  3.00  1.67</td>
</tr>
<tr>
<td></td>
<td>Tardiness</td>
<td>0.00  0.00  0.00  0.00  0.00  4.94  3.85  4.31  8.21  7.10  10.44 13.65</td>
</tr>
</tbody>
</table>
Table 3. Total Number of Workers = 25.

<table>
<thead>
<tr>
<th>Tot No of Workers</th>
<th>Configuration</th>
<th>Products (i-th Sequence)</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>1 4 7 10 11 12 6 9 8 3 2 5</td>
</tr>
<tr>
<td></td>
<td>Proc. Times (Days)</td>
<td>0.94 0.73 0.68 0.63 0.94 0.83 1.04 1.15 1.15 1.25 1.35 1.35</td>
</tr>
<tr>
<td></td>
<td>Cumm. Proc Times</td>
<td>0.94 1.67 2.34 2.97 3.91 4.74 5.78 6.93 8.07 9.32 10.68 12.03</td>
</tr>
<tr>
<td></td>
<td>Due Dates</td>
<td>1.00 2.33 4.00 4.67 5.00 5.00 3.33 4.67 4.33 2.00 1.67 3.00</td>
</tr>
<tr>
<td></td>
<td>Tardiness</td>
<td>0.00 0.00 0.00 0.00 0.00 0.00 2.45 2.26 3.74 7.32 9.01 9.03</td>
</tr>
</tbody>
</table>
Table 4. Total Number of Workers = 30.

<table>
<thead>
<tr>
<th>Tot No of Workers</th>
<th>Configuration</th>
<th>Products ( [i]-th Sequence)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Proc. Times (Days)</td>
<td>1.35</td>
</tr>
<tr>
<td></td>
<td>Cumm. Proc Times</td>
<td>1.35</td>
</tr>
<tr>
<td></td>
<td>Due Dates</td>
<td>2.33</td>
</tr>
<tr>
<td></td>
<td>Tardiness</td>
<td>0.00</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Proc. Times (Days)</td>
<td>1.67</td>
</tr>
<tr>
<td></td>
<td>Cumm. Proc Times</td>
<td>1.67</td>
</tr>
<tr>
<td></td>
<td>Due Dates</td>
<td>3.33</td>
</tr>
<tr>
<td></td>
<td>Tardiness</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Table 5. Total Number of Workers = 35.

<table>
<thead>
<tr>
<th>Tot No of Workers</th>
<th>Configuration</th>
<th>Products (i)-th Sequence</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Proc. Times</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Days)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1.67</td>
<td>2.08</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>1.25</td>
<td>2.08</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.50</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cumm. Proc</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Times</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1.67</td>
<td>3.75</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>5.00</td>
<td>7.08</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9.58</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Due Dates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>4.33</td>
<td>4.67</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>5.00</td>
<td>2.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.67</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tardiness</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>0.00</td>
<td>0.00</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>0.00</td>
<td>5.08</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7.92</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Due Dates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>2.33</td>
<td>3.33</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>4.00</td>
<td>4.67</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>5.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.00</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>3.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tardiness</td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>0.00</td>
<td>0.00</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.15</td>
<td>4.81</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Products (i)-th Sequence</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proc. Times</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Days)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.94</td>
<td>1.25</td>
<td>12</td>
</tr>
<tr>
<td>0.83</td>
<td>0.83</td>
<td></td>
</tr>
<tr>
<td>1.04</td>
<td>1.25</td>
<td></td>
</tr>
<tr>
<td>1.67</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cumm. Proc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Times</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.94</td>
<td>2.19</td>
<td>12</td>
</tr>
<tr>
<td>3.02</td>
<td>3.85</td>
<td></td>
</tr>
<tr>
<td>4.90</td>
<td>6.15</td>
<td></td>
</tr>
<tr>
<td>7.81</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Due Dates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.33</td>
<td>3.33</td>
<td>12</td>
</tr>
<tr>
<td>4.00</td>
<td>4.67</td>
<td></td>
</tr>
<tr>
<td>5.00</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>3.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tardiness</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>12</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>5.15</td>
<td>4.81</td>
<td></td>
</tr>
</tbody>
</table>
Table 6. Total Number of Workers = 40.

<table>
<thead>
<tr>
<th>Tot No of Workers</th>
<th>Configuration</th>
<th>Products ( [i]-th Sequence)</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Proc. Times (Days)</td>
<td>1.67 2.08 1.25 2.08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cumm. Proc Times</td>
<td>1.67 3.75 5.00 7.08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Due Dates</td>
<td>3.33 4.67 5.00 3.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tardiness</td>
<td>0.00 0.00 0.00 4.08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Proc. Times (Days)</td>
<td>0.94 0.73 0.68 1.15</td>
<td>0.63 0.83 1.25 1.35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cumm. Proc Times</td>
<td>0.94 1.67 2.34 3.49</td>
<td>4.11 4.95 6.20 7.55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Due Dates</td>
<td>1.00 2.33 4.00 4.33 4.67 5.00 2.00 1.67</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tardiness</td>
<td>0.00 0.00 0.00 0.00 0.00 0.00 4.20 5.89</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 7. Total Number of Workers = 45.

<table>
<thead>
<tr>
<th>Tot No of Workers</th>
<th>Configuration Products (([i])-th Sequence)</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Proc. Times (Days)</td>
<td>2.08</td>
<td>1.67</td>
<td>1.25</td>
<td>2.08</td>
</tr>
<tr>
<td></td>
<td>Cumm. Proc Times</td>
<td>2.08</td>
<td>3.75</td>
<td>5.00</td>
<td>7.08</td>
</tr>
<tr>
<td></td>
<td>Due Dates</td>
<td>3.00</td>
<td>4.33</td>
<td>5.00</td>
<td>2.00</td>
</tr>
<tr>
<td></td>
<td>Tardiness</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>5.08</td>
</tr>
<tr>
<td>45</td>
<td>Proc. Times (Days)</td>
<td>0.94</td>
<td>0.73</td>
<td>0.68</td>
<td>1.15</td>
</tr>
<tr>
<td></td>
<td>Cumm. Proc Times</td>
<td>0.94</td>
<td>1.67</td>
<td>2.34</td>
<td>3.49</td>
</tr>
<tr>
<td></td>
<td>Due Dates</td>
<td>1.00</td>
<td>2.33</td>
<td>4.00</td>
<td>4.67</td>
</tr>
<tr>
<td></td>
<td>Tardiness</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>2.66</td>
</tr>
</tbody>
</table>

Configuration Products (\([i]\)-th Sequence)
Table 8. Total Number of Workers = 50.

<table>
<thead>
<tr>
<th>Tot No of Workers</th>
<th>Configuration</th>
<th>Products (i-th Sequence)</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>25</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Proc. Times (Days)</td>
<td>1.25</td>
<td>0.73</td>
<td>0.68</td>
<td>1.15</td>
<td>0.83</td>
<td>1.15</td>
</tr>
<tr>
<td></td>
<td>Cumm. Proc Times</td>
<td>1.25</td>
<td>1.98</td>
<td>2.66</td>
<td>3.80</td>
<td>4.64</td>
<td>5.78</td>
</tr>
<tr>
<td></td>
<td>Due Dates</td>
<td>2.00</td>
<td>2.33</td>
<td>4.00</td>
<td>4.33</td>
<td>5.00</td>
<td>4.67</td>
</tr>
<tr>
<td></td>
<td>Tardiness</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.11</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>25</td>
<td>1</td>
<td>5</td>
<td>6</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Proc. Times (Days)</td>
<td>0.94</td>
<td>1.35</td>
<td>1.04</td>
<td>0.63</td>
<td>0.94</td>
<td>1.35</td>
</tr>
<tr>
<td></td>
<td>Cumm. Proc Times</td>
<td>0.94</td>
<td>2.29</td>
<td>3.33</td>
<td>3.96</td>
<td>4.90</td>
<td>6.25</td>
</tr>
<tr>
<td></td>
<td>Due Dates</td>
<td>1.00</td>
<td>3.00</td>
<td>3.33</td>
<td>4.67</td>
<td>5.00</td>
<td>1.67</td>
</tr>
<tr>
<td></td>
<td>Tardiness</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>4.58</td>
</tr>
</tbody>
</table>
Table 9. Total Number of Workers = 55.

<table>
<thead>
<tr>
<th>Tot No of Workers</th>
<th>Configuration</th>
<th>Products ([i]-th Sequence)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>Proc. Times (Days)</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>Cumm. Proc Times</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>Due Dates</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>Tardiness</td>
<td>0.00</td>
</tr>
<tr>
<td>55</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Proc. Times (Days)</td>
<td>1.25</td>
</tr>
<tr>
<td></td>
<td>Cumm. Proc Times</td>
<td>1.25</td>
</tr>
<tr>
<td></td>
<td>Due Dates</td>
<td>2.00</td>
</tr>
<tr>
<td></td>
<td>Tardiness</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Table 10. Total Number of Workers = 60.

<table>
<thead>
<tr>
<th>Tot No of Workers</th>
<th>Configuration</th>
<th>Products ( [i]-th Sequence)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td></td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>Proc. Times (Days)</td>
<td>2.08</td>
<td>1.25</td>
<td>1.56</td>
</tr>
<tr>
<td>Cumm. Proc Times</td>
<td>2.08</td>
<td>3.33</td>
<td>4.90</td>
</tr>
<tr>
<td>Due Dates</td>
<td>3.00</td>
<td>5.00</td>
<td>5.00</td>
</tr>
<tr>
<td>Tardiness</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>Proc. Times (Days)</td>
<td>1.56</td>
<td>1.46</td>
<td>1.56</td>
</tr>
<tr>
<td>Cumm. Proc Times</td>
<td>1.56</td>
<td>3.02</td>
<td>4.58</td>
</tr>
<tr>
<td>Due Dates</td>
<td>2.00</td>
<td>4.33</td>
<td>4.67</td>
</tr>
<tr>
<td>Tardiness</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td>Proc. Times (Days)</td>
<td>1.35</td>
<td>0.73</td>
<td>1.04</td>
</tr>
<tr>
<td>Cumm. Proc Times</td>
<td>1.35</td>
<td>2.08</td>
<td>3.13</td>
</tr>
<tr>
<td>Due Dates</td>
<td>1.67</td>
<td>2.33</td>
<td>3.33</td>
</tr>
<tr>
<td>Tardiness</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Table 11. Total Number of Workers = 65.

<table>
<thead>
<tr>
<th>Tot No of Workers</th>
<th>Configuration</th>
<th>Products ( [i]-th Sequence)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Proc. Times (Days)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cumm. Proc Times</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Due Dates</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tardiness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Configuration</td>
</tr>
<tr>
<td>65</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Proc. Times (Days)</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>Cumm. Proc Times</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>Due Dates</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>Tardiness</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Configuration</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Proc. Times (Days)</td>
<td>1.35</td>
</tr>
<tr>
<td></td>
<td>Cumm. Proc Times</td>
<td>1.35</td>
</tr>
<tr>
<td></td>
<td>Due Dates</td>
<td>1.67</td>
</tr>
<tr>
<td></td>
<td>Tardiness</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Table 12. Total Number of Workers = 70.

<table>
<thead>
<tr>
<th>Tot No of Workers</th>
<th>Configuration</th>
<th>Products ( [i]-th Sequence)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proc. Times (Days)</td>
<td>1.56</td>
<td>1.46</td>
<td>1.04</td>
</tr>
<tr>
<td>Cumm. Proc Times</td>
<td>1.56</td>
<td>3.02</td>
<td>4.06</td>
</tr>
<tr>
<td>Due Dates</td>
<td>2.00</td>
<td>4.33</td>
<td>5.00</td>
</tr>
<tr>
<td>Tardiness</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proc. Times (Days)</td>
<td>0.94</td>
<td>1.35</td>
<td>1.15</td>
</tr>
<tr>
<td>Cumm. Proc Times</td>
<td>0.94</td>
<td>2.29</td>
<td>3.44</td>
</tr>
<tr>
<td>Due Dates</td>
<td>1.00</td>
<td>3.00</td>
<td>4.67</td>
</tr>
<tr>
<td>Tardiness</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proc. Times (Days)</td>
<td>1.35</td>
<td>0.73</td>
<td>1.04</td>
</tr>
<tr>
<td>Cumm. Proc Times</td>
<td>1.35</td>
<td>2.08</td>
<td>3.13</td>
</tr>
<tr>
<td>Due Dates</td>
<td>1.67</td>
<td>2.33</td>
<td>3.33</td>
</tr>
<tr>
<td>Tardiness</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 13. Total Number of Workers = 75.

<table>
<thead>
<tr>
<th>Tot No of Workers</th>
<th>Configuration</th>
<th>Products ( i-th Sequence)</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>Proc. Times (Days)</td>
<td>1.25 1.35 1.15 0.94</td>
</tr>
<tr>
<td></td>
<td>Cumm. Proc Times</td>
<td>1.25 2.60 3.75 4.69</td>
</tr>
<tr>
<td></td>
<td>Due Dates</td>
<td>2.00 3.00 4.33 5.00</td>
</tr>
<tr>
<td></td>
<td>Tardiness</td>
<td>0.00 0.00 0.00 0.00</td>
</tr>
</tbody>
</table>

| 75                | Proc. Times (Days) | 0.94 1.15 0.83 |
|                   | Cumm. Proc Times  | 0.94 2.08 2.92 |
|                   | Due Dates         | 1.00 4.67 5.00 |
|                   | Tardiness         | 0.00 0.00 0.00 |

| 25                | Proc. Times (Days) | 1.35 0.73 1.04 0.68 0.63 |
|                   | Cumm. Proc Times  | 1.35 2.08 3.13 3.80 4.43 |
|                   | Due Dates         | 1.67 2.33 3.33 4.00 4.67 |
|                   | Tardiness         | 0 0 0 0 0 |
APPENDIX D

TABLE FOR GA EXPERIMENTATION

Table 1. Processing Times of 50 Products for 6 Configurations.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>1000</td>
<td>800</td>
<td>500</td>
<td>400</td>
<td>300</td>
<td>200</td>
</tr>
<tr>
<td>Std.Dev.</td>
<td>100</td>
<td>80</td>
<td>50</td>
<td>40</td>
<td>30</td>
<td>20</td>
</tr>
<tr>
<td>1</td>
<td>953</td>
<td>614</td>
<td>427</td>
<td>405</td>
<td>276</td>
<td>177</td>
</tr>
<tr>
<td>2</td>
<td>1004</td>
<td>895</td>
<td>516</td>
<td>385</td>
<td>265</td>
<td>173</td>
</tr>
<tr>
<td>3</td>
<td>970</td>
<td>825</td>
<td>507</td>
<td>341</td>
<td>277</td>
<td>197</td>
</tr>
<tr>
<td>4</td>
<td>1034</td>
<td>767</td>
<td>458</td>
<td>408</td>
<td>315</td>
<td>217</td>
</tr>
<tr>
<td>5</td>
<td>967</td>
<td>678</td>
<td>544</td>
<td>344</td>
<td>287</td>
<td>218</td>
</tr>
<tr>
<td>6</td>
<td>807</td>
<td>790</td>
<td>504</td>
<td>428</td>
<td>317</td>
<td>222</td>
</tr>
<tr>
<td>7</td>
<td>949</td>
<td>891</td>
<td>564</td>
<td>272</td>
<td>261</td>
<td>184</td>
</tr>
<tr>
<td>8</td>
<td>967</td>
<td>769</td>
<td>510</td>
<td>379</td>
<td>271</td>
<td>213</td>
</tr>
<tr>
<td>9</td>
<td>1137</td>
<td>776</td>
<td>472</td>
<td>392</td>
<td>320</td>
<td>205</td>
</tr>
<tr>
<td>10</td>
<td>985</td>
<td>892</td>
<td>509</td>
<td>380</td>
<td>294</td>
<td>215</td>
</tr>
<tr>
<td>11</td>
<td>889</td>
<td>839</td>
<td>526</td>
<td>430</td>
<td>307</td>
<td>211</td>
</tr>
<tr>
<td>12</td>
<td>1111</td>
<td>737</td>
<td>492</td>
<td>374</td>
<td>322</td>
<td>176</td>
</tr>
<tr>
<td>13</td>
<td>955</td>
<td>836</td>
<td>583</td>
<td>461</td>
<td>349</td>
<td>194</td>
</tr>
<tr>
<td>14</td>
<td>1062</td>
<td>733</td>
<td>577</td>
<td>408</td>
<td>289</td>
<td>198</td>
</tr>
<tr>
<td>15</td>
<td>973</td>
<td>749</td>
<td>436</td>
<td>376</td>
<td>295</td>
<td>194</td>
</tr>
<tr>
<td>16</td>
<td>1063</td>
<td>749</td>
<td>603</td>
<td>362</td>
<td>331</td>
<td>201</td>
</tr>
<tr>
<td>17</td>
<td>1111</td>
<td>775</td>
<td>488</td>
<td>409</td>
<td>314</td>
<td>205</td>
</tr>
<tr>
<td>18</td>
<td>834</td>
<td>775</td>
<td>439</td>
<td>390</td>
<td>310</td>
<td>209</td>
</tr>
<tr>
<td>19</td>
<td>1019</td>
<td>739</td>
<td>450</td>
<td>429</td>
<td>266</td>
<td>202</td>
</tr>
<tr>
<td>20</td>
<td>916</td>
<td>684</td>
<td>446</td>
<td>406</td>
<td>279</td>
<td>203</td>
</tr>
<tr>
<td>21</td>
<td>929</td>
<td>680</td>
<td>444</td>
<td>394</td>
<td>311</td>
<td>175</td>
</tr>
<tr>
<td>22</td>
<td>983</td>
<td>832</td>
<td>419</td>
<td>367</td>
<td>280</td>
<td>204</td>
</tr>
<tr>
<td>23</td>
<td>1105</td>
<td>704</td>
<td>441</td>
<td>412</td>
<td>288</td>
<td>167</td>
</tr>
</tbody>
</table>
Table 1: continued.

<table>
<thead>
<tr>
<th></th>
<th>980</th>
<th>830</th>
<th>477</th>
<th>409</th>
<th>336</th>
<th>177</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>894</td>
<td>1114</td>
<td>1043</td>
<td>959</td>
<td>997</td>
<td>1038</td>
</tr>
<tr>
<td></td>
<td>703</td>
<td>766</td>
<td>779</td>
<td>877</td>
<td>916</td>
<td>836</td>
</tr>
<tr>
<td></td>
<td>480</td>
<td>479</td>
<td>466</td>
<td>556</td>
<td>508</td>
<td>550</td>
</tr>
<tr>
<td></td>
<td>434</td>
<td>403</td>
<td>400</td>
<td>390</td>
<td>469</td>
<td>452</td>
</tr>
<tr>
<td></td>
<td>268</td>
<td>284</td>
<td>321</td>
<td>320</td>
<td>308</td>
<td>333</td>
</tr>
<tr>
<td></td>
<td>213</td>
<td>217</td>
<td>222</td>
<td>234</td>
<td>208</td>
<td>183</td>
</tr>
<tr>
<td>31</td>
<td>911</td>
<td>798</td>
<td>479</td>
<td>403</td>
<td>268</td>
<td>213</td>
</tr>
<tr>
<td></td>
<td>703</td>
<td>614</td>
<td>441</td>
<td>419</td>
<td>268</td>
<td>213</td>
</tr>
<tr>
<td></td>
<td>480</td>
<td>336</td>
<td>336</td>
<td>268</td>
<td>213</td>
<td>217</td>
</tr>
<tr>
<td>32</td>
<td>1004</td>
<td>948</td>
<td>887</td>
<td>950</td>
<td>992</td>
<td>984</td>
</tr>
<tr>
<td></td>
<td>766</td>
<td>614</td>
<td>556</td>
<td>550</td>
<td>550</td>
<td>510</td>
</tr>
<tr>
<td></td>
<td>477</td>
<td>441</td>
<td>426</td>
<td>419</td>
<td>443</td>
<td>396</td>
</tr>
<tr>
<td></td>
<td>390</td>
<td>336</td>
<td>295</td>
<td>299</td>
<td>324</td>
<td>251</td>
</tr>
<tr>
<td>33</td>
<td>959</td>
<td>798</td>
<td>556</td>
<td>477</td>
<td>830</td>
<td>916</td>
</tr>
<tr>
<td></td>
<td>703</td>
<td>477</td>
<td>508</td>
<td>409</td>
<td>780</td>
<td>508</td>
</tr>
<tr>
<td></td>
<td>480</td>
<td>336</td>
<td>469</td>
<td>268</td>
<td>409</td>
<td>213</td>
</tr>
<tr>
<td>34</td>
<td>992</td>
<td>792</td>
<td>550</td>
<td>336</td>
<td>336</td>
<td>508</td>
</tr>
<tr>
<td></td>
<td>980</td>
<td>550</td>
<td>469</td>
<td>268</td>
<td>409</td>
<td>213</td>
</tr>
<tr>
<td></td>
<td>830</td>
<td>477</td>
<td>308</td>
<td>213</td>
<td>208</td>
<td>183</td>
</tr>
<tr>
<td>35</td>
<td>984</td>
<td>910</td>
<td>510</td>
<td>452</td>
<td>419</td>
<td>336</td>
</tr>
<tr>
<td></td>
<td>916</td>
<td>508</td>
<td>469</td>
<td>268</td>
<td>208</td>
<td>183</td>
</tr>
<tr>
<td></td>
<td>830</td>
<td>477</td>
<td>308</td>
<td>213</td>
<td>208</td>
<td>183</td>
</tr>
<tr>
<td>36</td>
<td>1051</td>
<td>781</td>
<td>510</td>
<td>396</td>
<td>508</td>
<td>508</td>
</tr>
<tr>
<td></td>
<td>1114</td>
<td>766</td>
<td>477</td>
<td>409</td>
<td>336</td>
<td>177</td>
</tr>
<tr>
<td></td>
<td>703</td>
<td>403</td>
<td>284</td>
<td>213</td>
<td>217</td>
<td>217</td>
</tr>
<tr>
<td>37</td>
<td>1101</td>
<td>918</td>
<td>534</td>
<td>426</td>
<td>308</td>
<td>237</td>
</tr>
<tr>
<td></td>
<td>927</td>
<td>916</td>
<td>508</td>
<td>409</td>
<td>336</td>
<td>177</td>
</tr>
<tr>
<td></td>
<td>895</td>
<td>780</td>
<td>469</td>
<td>308</td>
<td>208</td>
<td>183</td>
</tr>
<tr>
<td>38</td>
<td>1005</td>
<td>907</td>
<td>550</td>
<td>452</td>
<td>419</td>
<td>336</td>
</tr>
<tr>
<td></td>
<td>1051</td>
<td>853</td>
<td>508</td>
<td>409</td>
<td>336</td>
<td>177</td>
</tr>
<tr>
<td></td>
<td>916</td>
<td>780</td>
<td>469</td>
<td>268</td>
<td>208</td>
<td>183</td>
</tr>
<tr>
<td>39</td>
<td>1028</td>
<td>844</td>
<td>550</td>
<td>426</td>
<td>308</td>
<td>237</td>
</tr>
<tr>
<td></td>
<td>980</td>
<td>916</td>
<td>508</td>
<td>409</td>
<td>336</td>
<td>177</td>
</tr>
<tr>
<td></td>
<td>830</td>
<td>477</td>
<td>308</td>
<td>213</td>
<td>208</td>
<td>183</td>
</tr>
<tr>
<td>40</td>
<td>1171</td>
<td>869</td>
<td>550</td>
<td>426</td>
<td>308</td>
<td>237</td>
</tr>
<tr>
<td></td>
<td>927</td>
<td>916</td>
<td>508</td>
<td>409</td>
<td>336</td>
<td>177</td>
</tr>
<tr>
<td></td>
<td>895</td>
<td>780</td>
<td>469</td>
<td>268</td>
<td>208</td>
<td>183</td>
</tr>
<tr>
<td>41</td>
<td>1005</td>
<td>907</td>
<td>550</td>
<td>452</td>
<td>419</td>
<td>336</td>
</tr>
<tr>
<td></td>
<td>1051</td>
<td>853</td>
<td>508</td>
<td>409</td>
<td>336</td>
<td>177</td>
</tr>
<tr>
<td></td>
<td>916</td>
<td>780</td>
<td>469</td>
<td>268</td>
<td>208</td>
<td>183</td>
</tr>
<tr>
<td>42</td>
<td>1101</td>
<td>918</td>
<td>534</td>
<td>426</td>
<td>308</td>
<td>237</td>
</tr>
<tr>
<td></td>
<td>927</td>
<td>916</td>
<td>508</td>
<td>409</td>
<td>336</td>
<td>177</td>
</tr>
<tr>
<td></td>
<td>895</td>
<td>780</td>
<td>469</td>
<td>268</td>
<td>208</td>
<td>183</td>
</tr>
<tr>
<td>43</td>
<td>1028</td>
<td>844</td>
<td>550</td>
<td>426</td>
<td>308</td>
<td>237</td>
</tr>
<tr>
<td></td>
<td>980</td>
<td>916</td>
<td>508</td>
<td>409</td>
<td>336</td>
<td>177</td>
</tr>
<tr>
<td></td>
<td>830</td>
<td>477</td>
<td>308</td>
<td>213</td>
<td>208</td>
<td>183</td>
</tr>
<tr>
<td>44</td>
<td>1171</td>
<td>869</td>
<td>550</td>
<td>426</td>
<td>308</td>
<td>237</td>
</tr>
<tr>
<td></td>
<td>927</td>
<td>916</td>
<td>508</td>
<td>409</td>
<td>336</td>
<td>177</td>
</tr>
<tr>
<td></td>
<td>895</td>
<td>780</td>
<td>469</td>
<td>268</td>
<td>208</td>
<td>183</td>
</tr>
<tr>
<td>45</td>
<td>1005</td>
<td>907</td>
<td>550</td>
<td>452</td>
<td>419</td>
<td>336</td>
</tr>
<tr>
<td></td>
<td>1051</td>
<td>853</td>
<td>508</td>
<td>409</td>
<td>336</td>
<td>177</td>
</tr>
<tr>
<td></td>
<td>916</td>
<td>780</td>
<td>469</td>
<td>268</td>
<td>208</td>
<td>183</td>
</tr>
<tr>
<td>46</td>
<td>1101</td>
<td>918</td>
<td>534</td>
<td>426</td>
<td>308</td>
<td>237</td>
</tr>
<tr>
<td></td>
<td>927</td>
<td>916</td>
<td>508</td>
<td>409</td>
<td>336</td>
<td>177</td>
</tr>
<tr>
<td></td>
<td>895</td>
<td>780</td>
<td>469</td>
<td>268</td>
<td>208</td>
<td>183</td>
</tr>
<tr>
<td>47</td>
<td>1028</td>
<td>844</td>
<td>550</td>
<td>426</td>
<td>308</td>
<td>237</td>
</tr>
<tr>
<td></td>
<td>980</td>
<td>916</td>
<td>508</td>
<td>409</td>
<td>336</td>
<td>177</td>
</tr>
<tr>
<td></td>
<td>830</td>
<td>477</td>
<td>308</td>
<td>213</td>
<td>208</td>
<td>183</td>
</tr>
<tr>
<td>48</td>
<td>1101</td>
<td>918</td>
<td>534</td>
<td>426</td>
<td>308</td>
<td>237</td>
</tr>
<tr>
<td></td>
<td>927</td>
<td>916</td>
<td>508</td>
<td>409</td>
<td>336</td>
<td>177</td>
</tr>
<tr>
<td></td>
<td>895</td>
<td>780</td>
<td>469</td>
<td>268</td>
<td>208</td>
<td>183</td>
</tr>
<tr>
<td>49</td>
<td>1028</td>
<td>844</td>
<td>550</td>
<td>426</td>
<td>308</td>
<td>237</td>
</tr>
<tr>
<td></td>
<td>980</td>
<td>916</td>
<td>508</td>
<td>409</td>
<td>336</td>
<td>177</td>
</tr>
<tr>
<td></td>
<td>830</td>
<td>477</td>
<td>308</td>
<td>213</td>
<td>208</td>
<td>183</td>
</tr>
<tr>
<td>50</td>
<td>954</td>
<td>928</td>
<td>518</td>
<td>409</td>
<td>237</td>
<td>187</td>
</tr>
</tbody>
</table>