Cell Loading and Product Sequencing Subject to Manpower Restrictions in Synchronized Manufacturing Cells

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

Fatih Yarimoğlu
March 2009
© 2009 Fatih Yarimoğlu. All Rights Reserved.
This thesis titled
Cell Loading and Product Sequencing Subject To Manpower Restrictions in
Synchronized Manufacturing Cells

by

FATİH YARIMOĞLU

has been approved for
the Department of Industrial and Systems Engineering
and the Russ College of Engineering and Technology by

Gürsel A. Süer
Professor of Industrial and Systems Engineering

Dennis Irwin
Dean, Russ College of Engineering and Technology
ABSTRACT

YARIMOĞLU, FATİH, M.S., March 2009, Industrial and Systems Engineering

Cell Loading and Product Sequencing Subject to Manpower Restrictions in Synchronized Manufacturing Cells (185 pp.)

Director of Thesis: Gürsel A. Süer

The main characteristic of the problem is that it considers a synchronized environment. A scheduling problem is handled in a manufacturing system that is using uniform time bucket approach for synchronization. There are three main aspects of the problem. The first aspect is the allocation of manpower. The manpower levels are determined in order to maximize production rate. Then, cell loading issue is addressed for multiple cells. Finally, a schedule is generated for each cell in order to minimize the number of periods where available manpower is exceeded. Two methods are used in order to solve these three main aspects of the problem, namely mathematical modeling and genetic algorithm. It has been concluded that mathematical modeling is better for cell loading and genetic algorithm is better for product sequencing. Also, minimizing similarity among products worked better during cell loading phase. Finally, sequential solution of cell loading and product sequencing was better compared to simultaneous solution of these two phases.

Approved: _____________________________________________________________

Gürsel A. Süer

Professor of Industrial and Systems Engineering
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>3</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>7</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>10</td>
</tr>
<tr>
<td>CHAPTER 1: INTRODUCTION</td>
<td>12</td>
</tr>
<tr>
<td>Manufacturing Systems</td>
<td>12</td>
</tr>
<tr>
<td>Cellular Manufacturing Systems</td>
<td>16</td>
</tr>
<tr>
<td>Manufacturing System and Cellular Control in Avon Company</td>
<td>17</td>
</tr>
<tr>
<td>Synchronization</td>
<td>19</td>
</tr>
<tr>
<td>Genetic Algorithm</td>
<td>20</td>
</tr>
<tr>
<td>Mathematical Models</td>
<td>23</td>
</tr>
<tr>
<td>Objectives of the Thesis</td>
<td>25</td>
</tr>
<tr>
<td>Organization of the Thesis</td>
<td>25</td>
</tr>
<tr>
<td>CHAPTER 2: LITERATURE REVIEW</td>
<td>26</td>
</tr>
<tr>
<td>CHAPTER 3: PROBLEM DESCRIPTION</td>
<td>33</td>
</tr>
<tr>
<td>Determination of Stages</td>
<td>33</td>
</tr>
<tr>
<td>Manpower</td>
<td>35</td>
</tr>
<tr>
<td>Product Sequencing</td>
<td>37</td>
</tr>
<tr>
<td>Cell Loading</td>
<td>40</td>
</tr>
<tr>
<td>CHAPTER 4: METHODOLOGY</td>
<td>43</td>
</tr>
<tr>
<td>Phase 1: Allocation of Manpower</td>
<td>43</td>
</tr>
<tr>
<td>Phase 2: Cell Loading</td>
<td>46</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 1: Uniform Time Bucket Approach ................................................................. 32
Table 2: Example 1: Operations for Product "Lion Pierce" .................................... 34
Table 3: Example 2: Operation Times for 10 Products (minute/unit) ...................... 35
Table 4: Manpower Levels and Production Rates for 10 Products .......................... 36
Table 5: Demand Values and Time Buckets Needed for 10 Products ....................... 37
Table 6: Sequencing Problem in a Cell (Example 1) ............................................ 37
Table 7: Sequencing Problem in a Cell (Example 2) ............................................ 39
Table 8: Optimal Sequence ................................................................................. 40
Table 9: Product Sequencing with Just Product Type 1 ........................................ 41
Table 10: Product Sequencing with Just Product Type 2 ...................................... 41
Table 11: Product Sequencing when Cell Loading was not Proper ....................... 42
Table 12: Proposed Three Phase Methodology ..................................................... 43
Table 13: Operation Times for Product 1 (hour) ..................................................... 45
Table 14: Results: Manpower Levels for Product 1 ................................................ 45
Table 15: Manpower Levels for Products 1 and 2 .................................................... 47
Table 16: Manpower Levels for 10 Products .......................................................... 49
Table 17: Similarity Matrix for 10 Products ............................................................ 50
Table 18: Calculation of Fitness for a Specific Chromosome .................................. 54
Table 19: Calculating Reproduction Probabilities .................................................. 56
Table 20: Cumulative Reproduction Probabilities .................................................. 57
Table 21: Final Sequence Obtained Using Math Modelling .................................... 66
Table 22: Fitness Calculation for Product Sequencing ............................................ 70
Table 23: Calculating Reproduction Probabilities for Product Sequencing ............... 70
Table 24: Cumulative Reproduction Probabilities for Product Sequencing .......... 71
Table 25: Final Sequence Obtained Using GA ....................................................... 81
Table 26: Fitness Calculation for cell 1 ................................................................... 90
Table 27: Fitness Calculation for cell 2 ................................................................... 90
Table 28: Data Set 1 ............................................................................................... 93
Table 29: Usage of Data Sets........................................................................................................94
Table 30: GA or Random Approach – 20 Jobs (30 Replicates) ..............................................95
Table 31: Calculation of Combinations for Cell Loading .................................................... 98
Table 32: Experiment Results for Cell Loading - 10 Jobs (30 Replicates) ............................99
Table 33: Experiment Results for Cell Loading - 20 Jobs (30 Replicates) ............................99
Table 34: Experiment Results for Cell Loading - 40 Jobs (30 Replicates) ............................100
Table 35: Fischer Test for Cell Loading Parameters (40 Jobs) ............................................ 102
Table 36: Calculation of Combinations for Product Sequencing ....................................... 103
Table 37: Number of Optimal Results for Product Sequencing - 10 Jobs ......................... 104
Table 38: Best Parameters for Product Sequencing - 20 Jobs ............................................ 105
Table 39: Number of Optimal Results for Product Sequencing - 20 Jobs ......................... 106
Table 40: Best Parameters for Product Sequencing - 40 Jobs ............................................ 107
Table 41: Number of Optimal Results for Product Sequencing - 40 Jobs ......................... 108
Table 42: The Combination with the Best Average Result (40 Jobs) .................................. 109
Table 43 Combination of Parameters for Worst Cell Loading and Product Sequencing Result - 10 Jobs .................................................................................................................. 111
Table 44 Table 44: Best Combination of Parameters for Cell Loading and Product Sequencing- 20 Jobs .................................................................................................................. 111
Table 45: Best Combination of Parameters for Cell Loading and Product Sequencing- 40 Jobs ................................................................................................................................. 114
Table 46: Population Size vs. Number of Generations ....................................................... 115
Table 47: Results for Population size vs. Number of Generations ................................... 116
Table 48: Fischer Test for Population Size vs. Number of Generations ............................ 118
Table 49: Effect of Phases ............................................................................................... 119
Table 50: GA vs. Math Modeling for Cell Loading ......................................................... 125
Table 51: GA vs. Math Modeling for Product Sequencing ............................................ 129
Table 52: Possible Options in Order to Accomplish Cell Loading and Product Sequencing ................................................................................................................................. 130
Table 53: Possible Options with Additional Options in Order to Accomplish Cell Loading and Product Sequencing ........................................................................................................ 131
Table 54: Better Cell Loading Parameters ................................................................. 132
Table 55: Better Product Sequencing Parameters .................................................. 132
Table 56: Better Cell Loading and Product Sequencing Parameters ..................... 132
Table 57: Overall Results for 9 Options for Cell Loading and Product Sequencing (40 Jobs) .............................................................................................................. 133
Table 58: Shortcuts .................................................................................................. 135
Table 59: Better Working Parameters for GA as a Summary .................................. 136
Table 60: Data set 1 ................................................................................................. 149
Table 61: Data set 2 ................................................................................................. 150
Table 62: Data set 3 ................................................................................................. 151
Table 63: Data set 4 ................................................................................................. 152
Table 64: Calculation of Similarities in Two Cells .................................................. 185
LIST OF FIGURES

Figure 1: Manufacturing System Classification Based on Volume and Variety ............. 13
Figure 2: Product Layout ................................................................................................. 14
Figure 3: Process Layout ................................................................................................. 15
Figure 4: Fixed Layout .................................................................................................... 15
Figure 5: Cellular Layout ............................................................................................... 16
Figure 6: Operations in Cellular layout ........................................................................ 17
Figure 7: Structure of Cells at Avon Lomalinda ............................................................. 18
Figure 8: Representation of a chromosome ................................................................. 20
Figure 9: Example for Crossover .................................................................................. 21
Figure 10: Example for Crossover Continues (Rubin and Ragatz) .............................. 21
Figure 11: General Representation of the Genetic Algorithms ..................................... 23
Figure 12: Chromosome Representation for Cell Loading .......................................... 51
Figure 13: Interface of the Cell loading Program ......................................................... 52
Figure 14: GA for Cell Loading .................................................................................... 53
Figure 15: Mutation Types ............................................................................................ 58
Figure 16: Reciprocal Exchange Mutation ................................................................... 58
Figure 17: Insertion Mutation ....................................................................................... 59
Figure 18: Cell Loading Results for the Example Problem .......................................... 60
Figure 19: Sending Assigned Products to Cells ............................................................ 61
Figure 20: Products Assigned to Cell 1 ......................................................................... 61
Figure 21: Products Assigned to Cell 2 ......................................................................... 62
Figure 22: Chromosome Representation for Product Sequencing ............................ 66
Figure 23: Interface of Product Sequencing Program .................................................. 67
Figure 24: GA for Product Sequencing ........................................................................ 69
Figure 25: Crossover Strategies for Product Sequencing ............................................. 72
Figure 26: Position-based Crossover .......................................................................... 74
Figure 27: Order Crossover ......................................................................................... 76
Figure 28: Mutation Types for Product Sequencing .................................................... 77
CHAPTER 1: INTRODUCTION

In this chapter, manufacturing systems, cellular manufacturing systems, synchronization are explained. Later, manufacturing system in Avon Company is mentioned. Then, problem solving methods that are used in this paper which are genetic algorithm and mathematical modeling are introduced. Lastly, objectives of the thesis and organization of the thesis are given.

Manufacturing Systems

In general, manufacturing systems can be grouped into four main categories according to layout of a plant:

1. Product Layout
2. Process Layout
3. Fixed Layout
4. Cellular Layout

Product volume and product variety are very important in determining the layout of a plant. Therefore, a comparison according to volume vs. variety is given in Figure 1 for these layout types. (Askin & Skandridge, 1993, p.11)
When demand is high and product variety is low, the product layout is suitable. The layout is arranged for a specific product. This flow line simplifies the scheduling task. As a result, work-in-process inventory is low and efficiency is very high for these systems.

In the process layout, the machines are grouped into departments according to similarities in their processing capabilities. Therefore, these systems can handle a wide variety of products. Mostly, job shop configuration is used. In these systems, the work-in-process inventory is high and efficiency is low. This layout type is also known as functional layout.

*Figure 1: Manufacturing System Classification Based on Volume and Variety.*
Fixed layout is used only when the product remains in a fixed position during production like trains, airplanes, submarines, etc, and machines, equipment and people move around to work on the product.

Lastly, in the cellular layout, products that are similar according to processing requirements are grouped together and are assigned to a cell. Since the products are completed in a cell, moving distance is negligible. Also, work-in-process inventory, setup times, flow times, lead times are low for these systems. Furthermore, a team atmosphere is created and, as a result, productivity and quality is improved.

Product layout, Process layout, Fixed Layout and Cellular Layout are shown in Figure 2, Figure 3, Figure 4, and Figure 5, respectively.

Figure 2: Product Layout.
**Figure 3:** Process Layout.

**Figure 4:** Fixed Layout.
Cellular Manufacturing Systems

Cellular manufacturing is also known as group technology on the shop floor. It is based on specialization of part and family. The basic idea is to complete an individual product totally in a single unit instead of moving the work from one department to the other. This is possible by grouping similar products together (which leads to family description) and also organizing the facility by product family.

Kay and Suresh (Suresh N. C. & Kay J. M, 1998) summarize the benefits of this type of manufacturing as:

- Less work-in-process (WIP) inventory, so less space is required
- Smaller and predictable lead times
- Smaller setup times because of family similarities
- Smaller lot production
- Simpler work flow
• Simpler and better control, because there is no operation-wise division in responsibilities like process or product layout

The nature of cellular layout leads to lower lead times as shown in Figure 6 for a batch production (Q=Queue time, S=Setup time, O=Operation Time) (Suresh N. C. & Kay J. M, 1998).

![Functional Layout Diagram](image)

**Figure 6**: Operations in Cellular Layout.

Manufacturing System and Cellular Control in Avon Company

This study discusses a problem observed in Avon Lomalinda Inc, a jewelry manufacturing company located in San Sebastian, Puerto Rico.

The Avon Company had process layout until 1988. Later, in that year, conversion to cellular layout occurred. After the conversion, inventory levels dropped and lead times were reduced. However, the plating operation could not be incorporated into manufacturing cells. The cells in this environment are labor-intensive cells meaning that
continuous operator involvement is necessary and the machines and equipment are usually inexpensive and lighter compared to machine-intensive manufacturing cells. The crew size varies between 20 and 30 workers in a cell. There are inter-cell transfers involved for most of the products between manufacturing cells and plating cells. Some of the operations are handled in manufacturing cells and then, products are sent to the plating area. Later, parts come back to the cell for the remaining operations to be completed. The cellular structure at Avon Lomalinda is given in Figure 7.

![Figure 7: Structure of Cells at Avon Lomalinda.](image)
The problem tackled in this cellular manufacturing environment has two main aspects:

1. Cell loading
2. Cell Scheduling

Cell loading deals with the assignment of the products to the cells. On the other hand, cell scheduling is performed once the products assigned to a cell are known. It involves product sequencing and manpower allocation decisions.

Synchronization

This paper also deals with synchronous manufacturing. Synchronization can be described as: (Süer and Gonzalez, 1993)

“A systematic way that provides a perfect flow of material through the production system by making it available for the right resource at the right time using a periodic approach with the objective of simplifying the scheduling task and minimizing the work in process inventory and the flow time”.

There are several methods to achieve synchronous manufacturing according to Süer and Gonzalez (1991). They are material flow in assembly lines, period batch control, kanban system, optimized production technology and uniform time bucket approach. Among them, uniform time bucket approach has been used in the Avon Lomalinda manufacturing plant which was implemented by Süer in 1990 to 1995 (Süer, 1998).
Genetic Algorithm

Genetic Algorithm is one of the methods employed to solve the problem in this paper. Genetic Algorithm is a meta-heuristic approach and it does not guarantee the optimum. However, in many cases it finds the optimal solution. The most important advantages of Genetic approach are that it can be applied:

- to any objective
- in any condition

Genetic Algorithm can be applied to any objective; it is just about changing the fitness function. GA can also be applied in any condition; for example, sometimes we have more than one objective and it can be used to find a good solution for multiple objectives.

First of all, the order of jobs is randomly determined. These possible sequences are called chromosome. Each job in the chromosome is called gene. There is an example of a 10-job problem in Figure 8.

![Figure 8: Representation of a Chromosome.](image)

A number of chromosomes come together to create a population. Population size is the term used to show the number of chromosomes in a population. Chromosomes
of a population are created by using different operators namely, **crossover** and/or **mutation**.

Rubin and Ragatz (1995) used the following approach in their paper for crossover. For instance, let’s say that there are 6 jobs to be scheduled and assume that they are divided into 2 groups of jobs as G1= \{1, 2, 3\} and G2= \{4, 5, 6\}. Chromosomes for Parent 1 and Parent 2 are given in Figure 9. In Parent 1, jobs in G1 are re-scheduled based on the sequence of these jobs in Parent 2.

![Figure 9: Example for Crossover.](image)

Parent 1 schedule may take the sequence of G1 (1st, 2nd and 3rd jobs) jobs from Parent 2 schedule to produce offspring like shown below: (The sequence of G1 jobs in Parent 2 schedule is 1-3-2)

![Figure 10: Example for Crossover continues (Rubin and Ragatz).](image)

Parent 1 schedule may take the sequence of G1 jobs or G2 jobs from Parent 2 or both or none. This is repeated for Parent 2 schedule as well. So, eight offspring are
produced from 2 parents. Additionally, out of eight offspring, 2 of them will have exactly the same sequence as Parent 1 and Parent 2.

**Fitness** is a measure that shows how good a solution the chromosome corresponds. Therefore, it is used to choose which chromosomes will go to the next generation. **Mutation** is another GA operator and it swaps the places of genes. A mutation probability indicates the probability that a job is modified. However, Rubin and Ragatz (1995) determined this probability for a chromosome. So, they determined if a chromosome will go through mutation first. Mutation is done either by swapping adjacent jobs if the schedule improves with respect to total tardiness or by swapping two jobs in the schedule randomly. If some of the population members are produced randomly each generation in order to maintain diversity, this operation is called **immigration**. If the best known solution automatically advances to the next generation, this practice is called **elitism**. The overall idea in genetic algorithm is to obtain better individuals as the number of generations increases. In Figure 11, the general representation of the genetic approach is given (Population size is 5 for this example). Lastly, the chromosomes which will go to next generation are determined using different **selection** methods.
Mathematical Models

In order to address the issues mentioned in this study, a mathematical modeling is also employed. Mathematical models are representations which try to explain some aspects of a system or a problem. It consists of an objective function and various constraints in general.

In order to build a mathematical model, the first thing is to identify all decision variables \( (x_1, x_2, y_1, y_2, \text{etc.}) \). Decision variables are the parameters that can be controlled by the decision maker. After the identification of decision variables, the objective function and constraints should be identified. The objective function is also called “cost function” if it is of minimization type, “profit function” if it is of
maximization type. The constraints can be of equality type or inequality type. The constraints restrict the values that decision variables can take.

A solution is called “feasible solution” if the decision variables satisfy all of the constraints. The best solution among feasible solutions is called the “optimal solution”. If the objective function and all of the constraints are linear, then it is called “Linear Mathematical Model”. If there is at least one non-linear equation or if the objective is not linear, then it is called “Non-linear Mathematical Model”. Here is a very simple mathematical model to illustrate the objective function and constraints.

**objective function:**

\[
\text{maximize } z = x_1 + x_2
\]

**subject to:**

\[
x_1 \leq 3
\]

\[
x_2 \leq 2
\]

\[
x_2 + x_1 \geq 4
\]

\[
x_1, x_2 \text{ integers}
\]

There are various programs available to solve mathematical models. In this study, OPL is used to solve them.

In general, mathematical models find the optimal solution when the problem size is small. However, heuristics and meta-heuristics (like genetic algorithm) may be necessary for solving bigger problems.
Objectives of the Thesis

The objectives of this thesis can be summarized as below:

➢ To determine cell loading and product sequencing in a synchronized cell such that number of periods that available manpower is exceeded as less as possible.
➢ To develop Mathematical models to solve cell lading and product sequencing problems.
➢ To develop unique genetic algorithm approaches to solve cell loading and product sequencing problems.
➢ To compare performance of mathematical models and genetic approaches.
➢ By carrying out this study, we hope that scheduling issues in a synchronized cell will be successfully addressed.

Organization of the Thesis

This thesis consists of 6 main parts: Introduction, Literature Review, Problem Description, Methodology, Experiments & Results and finally Conclusion.

Literature review part is organized considering cell loading, cell scheduling, genetic algorithms and synchronization. Problem is divided to three main parts: Allocation of manpower, Cell loading, and Product Sequencing. Solution techniques are discussed for all of these parts. Similarly, Methodology and Experiments & Results are also organized according to these three main parts of the problem. Lastly, final discussion is made in the conclusion.
CHAPTER 2: LITERATURE REVIEW

The literature review is organized considering cell loading, cell scheduling, genetic algorithms and synchronization.

In terms of cell loading, the number of papers in the literature is not many. One of the earliest works was done by Greene & Sadowski (1984) and Greene & Cleary (1985). They developed several cell loading rules and compared them considering their advantages and disadvantages. In 1995, Süer, Saiz, Dagli and Gonzalez analyzed a real manufacturing environment and developed some cell loading rules for connected cells. Later, Süer, Saiz and Gonzalez extended their work to independent cells (1999). Süer (1997) discussed minimization of number of tardy jobs in a multi-period environment. He developed mathematical formulations for different approaches, and then compared them. Later, cell loading and cell size determination for multiple periods have been discussed by Süer and Bera (1998). If a product is assigned to a cell, it has to be assigned to the same cell for the coming periods as well because of setup, learning curve, etc. This was the most important characteristic of the problem. They proposed two phase methodology to maximize the number of products in all periods using available capacity. Süer, Arikan, and Babayigit (2008) proposed a fuzzy bi-objective mathematical model which performs several tasks at the same time: deciding number of cells to open, deciding cell sizes, assigning products to cells, determining the sequence of products in each cell. The objectives were to minimize number of tardy jobs and the total number of operators which conflict each other. They solved the problem by using fuzzy models. As a result, they concluded that fuzzy gave a chance to acquire solutions according to preferences.
There are also some papers on manpower allocation and cell loading. Süer and Dagli (1994) developed six rules and six algorithms in order to create a knowledge-based system for resource allocation for flexible manufacturing systems. In 2005, Süer and Dagli first produced a methodology to minimize total intra-cell manpower transfers. Later, they obtained machine level based similarity coefficients that measures the similarity based on the number of machines. They used these coefficients to minimize the machine requirements, space requirements and also makespan. The same machine similarity based coefficient was first developed by Süer and Ortega (1994). One of the latest works on this issue is done by Süer and Tummaluri (2008). They discussed learning and forgetting in labor-intensive cells and operator assignments.

As to cell scheduling, there is plenty of work, but there has been only couple studies done on cell scheduling in a synchronized environment that uses uniform time bucket approach. Riezebos’ paper in 2003 is one of them. He sought to obtain proper work order release decisions in a synchronous manufacturing cell. In order to do that, he developed several heuristics and also a mathematical model. He found the optimal solution by solving a mathematical model. He multiplied capacity shortages in each time period with weights in the objective function in the mathematical model. As a result, he was able to solve the capacity balancing problem in a synchronized manufacturing environment with five stages. While most studies focus on performance measures like total tardiness, makespan, number of tardy jobs, average flow time, etc., the performance measure used in his study was to minimize the weighted sum of capacity shortages in each period. Concurrent design of a Period Batch Control (PBC) system by genetic
algorithms was proposed by Escobar (2003). PBC can also be considered as a
synchronization method. He performed a new methodology about embedded GA. The
objective was to minimize work-in capital investment, transfer and set-up costs. Süer and
Gonzalez proposed 4-hour uniform time bucket approach for cell scheduling in their
paper (1993). The principle is based on the “right part”, the “right quantity” and the right
time. In 1999, they extended the line balancing concept to the entire manufacturing
system again by using uniform time bucket approach (Süer & Gonzalez).

Genetic algorithms have been widely used to solve various scheduling problems.
Genetic approach was applied to traveling salesman problem by Chambers (1995) and
Michalewicz (1996), and this increased the interest to genetic algorithms. After that,
several textbooks that mention genetic algorithms applications to scheduling were
written. Cheng and Gen discusses these issues in their textbooks extensively which were
published in 1997 and 2000. There are many papers which try to solve complex
minimized the average flow time in the presence of non-zero ready times using genetic
algorithm. Babayigit (2003) used genetic algorithm to solve manpower allocation and cell
loading problem. He used three-division population (similar to soccer leagues) to
describe the problem. He put the chromosomes (he also resembled them to teams) with
similar strength to the same league in order to create a challenging environment. He also
thought mutation as transferring a new player to a team. As a result, he discovered that
genetic algorithm can give better results than traditional approaches in some cases. He
found optimal solutions or solutions close to the optimal faster than mathematical models
especially in large problems. One of the most recent works was done by Süer, Vazquez and Cortes (2005). They applied genetic algorithm to cell loading. They used three approaches to solve the problem and then compared them. They found that a hybrid genetic algorithm with local optimizer approach was able to increase the quality of the solution. On the other hand, there are no papers of genetic algorithms that deal with scheduling in a synchronized environment. So, this study can be considered first in that area.

The implementation of synchronized manufacturing in Avon Lomalinda was between 1990 and 1995. Several projects undertaken by Süer are given in chapter F4 in Group Technology and Cellular manufacturing book (Süer, 1998). Süer explains that there was a desperate need for synchronization. The main reasons for that were, the schedules were different every day and implementation of them heavily depended on the planners, schedulers, supervisors and employees. As shown in Figure 7 (Page 18), the plating area is common to all manufacturing cells. The material flow between cells and plating area needed to be synchronized in order to maximize utilization of resources. A modified version of the Uniform Time Bucket Approach was used to implement synchronization. A week was divided into 10 time buckets and processes were clustered into (s) stages. Consequently, he was able to convert this dynamic and stochastic scheduling problem into a static and deterministic problem which made scheduling a lot easier. After the implementation of the projects, jobs started and finished at regular intervals, and then in this periodic fashion, the need to prepare a different schedule everyday was eliminated.
Synchronization methods were categorized into 5 groups by Süer and Gonzalez (1991):

1. Assembly line balancing
2. Period Batch Control
3. Kanban System
4. Optimized Production Technology
5. Uniform time bucket approach

Assembly line balancing is the oldest synchronization method known. It was first implemented by Henry Ford in 1913. Wild (1972) explains the most important features of an assembly line. A cycle time is determined in which operations in a station should be finished and the products should move from one station to another at the end. There is no work-in-process inventory and the transfer size is one. However, operators should be assigned to line so that production rate is met with minimum idle time.

Burbidge explains Period Batch control in his book extensively (1975). If the products are required in period t, they are assembled in period (t-1), manufactured in period (t-2) and the raw material is received in period (t-3).

The original development of the Kanban (card) system was done by Dr. Ohno for Toyota (Ohno, 1988). According to this system, workstations produce products only when they receive a card and empty container showing that more will be needed in the subsequent production area. In short, the production is pulled to production line upon consumption. The objective is to lower inventory but it is still common to see safety
stocks in these systems. Kanban system is generally considered as the simplest and the most effective way of dealing with production and inventory control.

As other synchronization methods, optimized production technology method tries to lower the inventory level. In order to increase the throughput of the system, bottlenecks become very important since they directly determine the output of the manufacturing system. The production in the bottleneck should be protected with a buffer waiting in front of the bottleneck to keep it working and to gain maximum possible production rate. The OPT method was firstly developed by Eliyahu Goldratt. The original name of this procedure was the optimized production timetables. In 1982, he changed its name to optimized production technology. Later, Jacobs explained the most important features of optimized production technology (1983).

Uniform time bucket approach was suggested by Süer in 1989. He applied this method to multi-product and multi-stage production systems. First, time buckets are determined and products move from one stage to another at the end of each time bucket. For example, if there are 6 stages and if a set of products is scheduled to be completed at time $t$ (let’s call that $S_t$), then in period $t-1$, that set of products are processed at stage 5; in period $t-2$, they are processed at stage 4 and so on as shown in Table 1 below.
Table 1

*Uniform Time Bucket Approach*

<table>
<thead>
<tr>
<th>STAGE</th>
<th>...</th>
<th>t-6</th>
<th>t-5</th>
<th>t-4</th>
<th>t-3</th>
<th>t-2</th>
<th>t-1</th>
<th>t</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The batch size can be one or more. The number of stages and duration of time buckets affect both lead times and work-in-process inventory. Therefore, both the number of stages and duration of the time buckets should be kept as low as possible.
CHAPTER 3: PROBLEM DESCRIPTION

This study deals with labor-intensive manufacturing cells. The machines and equipment are small and inexpensive in labor-intensive manufacturing cells. As a result, machine investment costs and machine utilization is less important compared to machine-intensive manufacturing cells. Typically, there are more operators than the number of operations. Therefore, it is expected that some of the machines will have to be duplicated to utilize all the operators and hence maximize production rates. As a result, the availability and the allocation of manpower are a lot more important in these systems.

Determination of Stages

There are 5 stages in the system. It may be helpful to understand the problem better to illustrate how these stages were determined.

First of all, for example, consider a product called “Lion Pierce” with the operations, transfer sizes, processing times as given below in Table 2 (Süer & Gonzalez, 1993).
Table 2

Example - 1: Operations for Product "Lion Pierce"

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Casting</td>
<td>1</td>
<td>0.21</td>
<td>In</td>
</tr>
<tr>
<td>2</td>
<td>Degating</td>
<td>1</td>
<td>0.06</td>
<td>In</td>
</tr>
<tr>
<td>3</td>
<td>Cleaning</td>
<td>1</td>
<td>0.30</td>
<td>In</td>
</tr>
<tr>
<td>4</td>
<td>Deburring</td>
<td>1</td>
<td>0.19</td>
<td>In</td>
</tr>
<tr>
<td>5</td>
<td>Putting Sleeve</td>
<td>1</td>
<td>0.13</td>
<td>In</td>
</tr>
<tr>
<td>6</td>
<td>Tumbling</td>
<td>batch</td>
<td>210</td>
<td>In</td>
</tr>
<tr>
<td>7</td>
<td>Racking</td>
<td>1</td>
<td>0.34</td>
<td>In</td>
</tr>
<tr>
<td>8</td>
<td>Plating</td>
<td>batch</td>
<td>180</td>
<td>Out</td>
</tr>
<tr>
<td>9</td>
<td>Unracking</td>
<td>1</td>
<td>0.19</td>
<td>In</td>
</tr>
<tr>
<td>10</td>
<td>Carding</td>
<td>1</td>
<td>0.25</td>
<td>In</td>
</tr>
<tr>
<td>11</td>
<td>Packing</td>
<td>1</td>
<td>0.14</td>
<td>In</td>
</tr>
</tbody>
</table>

Since tumbling has the biggest processing time, length of time buckets should be greater than 210 minutes which equals to 3.5 hours. The time bucket is determined like this for remaining products as well, and because of the similarities in operations, this calculation of time buckets remain the same for all products. As workday is 8 hours, 4-hour time bucket was thought to be convenient in order to control batches better. Lastly, first 5 operations were grouped in first stage. Since tumbling and plating requires the entire batch, they were regarded as separate stages (stage 2 and 4). Racking is between tumbling and plating, so it was considered as another stage (stage 3). Lastly, 9th, 10th and 11th operations were grouped together as the 5th stage.

Consequently, time bucket is 4 hours, so it can be easily concluded that there are 2 time buckets in a day.
Manpower

The first problem is to determine manpower levels for all products. The processing times for all products and the total number of workers available in a cell is known. For example, assume that operations are grouped into 5 stages and manpower levels for 10 products needs to be determined. Also, assume that the operation times for 10 products are given as in Table 3.

Table 3

<table>
<thead>
<tr>
<th>Products</th>
<th>Stages</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>0.6</td>
<td>1</td>
<td>0.8</td>
<td>1.2</td>
<td>0.4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.6</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>3</td>
<td>0.6</td>
<td>0.4</td>
<td>0.4</td>
<td>1.6</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0.4</td>
<td>1.2</td>
<td>1.2</td>
<td>0.6</td>
<td>0.4</td>
</tr>
<tr>
<td>5</td>
<td>0.4</td>
<td>0.4</td>
<td>1</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>6</td>
<td>0.6</td>
<td>0.4</td>
<td>0.4</td>
<td>1.6</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0.4</td>
<td>0.4</td>
<td>1</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>8</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>9</td>
<td>0.6</td>
<td>0.6</td>
<td>1</td>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td>10</td>
<td>1.2</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Manpower levels should be known in order to start sequencing products. Our main objective is to find the sequence of products in a cell to minimize the number of time buckets where available manpower is exceeded. In our problem, the number of
products assigned to a cell is 10 considering 4 hour time bucket and one shift work. A product that starts in stage one at period one goes to stage two at period two, then stage three at period three and so on. That product leaves the system at the end of period five.

Assume that manpower levels and also production rates are given as shown in Table 4. Then if the demand values are given as in Table 5, time buckets needed for each product will be one since the demand will be met by producing each product for one time bucket.

Table 4

<table>
<thead>
<tr>
<th>Products</th>
<th>Operations</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op. 1</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Op. 2</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Op. 3</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Op. 4</td>
<td>6</td>
<td>4</td>
<td>8</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Op. 5</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Production Rate</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>units/minute</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Products</th>
<th>Production for an hour</th>
<th>300</th>
<th>300</th>
<th>300</th>
<th>300</th>
<th>300</th>
<th>300</th>
<th>300</th>
<th>300</th>
<th>360</th>
<th>units/hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Production for 4 hours</td>
<td>1200</td>
<td>1200</td>
<td>1200</td>
<td>1200</td>
<td>1200</td>
<td>1200</td>
<td>1200</td>
<td>1200</td>
<td>1200</td>
<td>1440</td>
<td>units/4 hours</td>
</tr>
</tbody>
</table>
Table 5

**Demand Values and Time Buckets Needed for 10 Products**

<table>
<thead>
<tr>
<th>Products</th>
<th>Demand</th>
<th>Calculated Time bucket</th>
<th>Exact time buckets needed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1060</td>
<td>0.88</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1080</td>
<td>0.9</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1110</td>
<td>0.93</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1200</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1150</td>
<td>0.96</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1100</td>
<td>0.92</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1100</td>
<td>0.92</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1050</td>
<td>0.88</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1070</td>
<td>0.89</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1250</td>
<td>0.87</td>
<td>1</td>
</tr>
</tbody>
</table>

Product Sequencing

After determining time buckets, these products need to be scheduled in a cell. The outcome of a random sequence of products (1-3-4-5-6-7-2-10-8-9) can be presented as in Table 6.

Table 6

**Sequencing Problem in a Cell (Example 1)**

<table>
<thead>
<tr>
<th></th>
<th>Week-1</th>
<th>Week-2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>TU</td>
</tr>
<tr>
<td></td>
<td>am pm</td>
<td>am pm</td>
</tr>
<tr>
<td>t</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Products</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Stage 1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Stage 2</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Stage 3</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Stage 4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Stage 5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Remaining Manpower</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
As it is shown, the table has some grey cells. These grey cells which are at the left bottom of the table show scheduling decisions made in previous periods. There are also black cells at the right up side of the table. These values are estimates for the next periods, and are calculated as the average manpower levels for every stage.

The cell capacity for a period is 20 workers. Therefore, that number should not be exceeded in any period. The remaining manpower row shows the extra workers left after allocation and is calculated as (20 – total manpower level assigned to that period). For example, for period 1 (t = 1), the remaining manpower is 2 \[20 – (3 + 4 + 6 + 3 + 2)\]. When this number is negative, it means manpower capacity is exceeded (shortage of workers). In Table 6, there are three time buckets with shortages in capacity, in period 5 (-1), in period 8 (-9) and in period 9 (-1). Consequently, the objective here is to make the number of time buckets that shortage occurs as minimum as possible.

If the sequence is changed as shown in Table 7, there will be 5 periods where the available number of workers will be exceeded.
Table 7

Sequencing Problem in a Cell (Example 2)

<table>
<thead>
<tr>
<th></th>
<th>Week-1</th>
<th></th>
<th>Week-2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>TU</td>
<td>W</td>
</tr>
<tr>
<td></td>
<td>am</td>
<td>pm</td>
<td>am</td>
</tr>
<tr>
<td>t</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Products</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Stage 1</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Stage 2</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Stage 3</td>
<td>6</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Stage 4</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Stage 5</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Remaining Manpower</td>
<td>2</td>
<td>-1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As the product sequence significantly affects the performance measure, it becomes important to focus on how to solve this problem. A better solution for the same problem is given in Table 8.
Table 8

**Optimal Sequence**

<table>
<thead>
<tr>
<th></th>
<th>M</th>
<th>T</th>
<th>W</th>
<th>TH</th>
<th>FR</th>
<th>M 2</th>
<th>T</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>am</td>
<td>pm</td>
<td>am</td>
<td>pm</td>
<td>am</td>
<td>pm</td>
<td>am</td>
<td>pm</td>
</tr>
<tr>
<td>t</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Products</td>
<td>9</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>Stage 1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Stage 2</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Stage 3</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Stage 4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Stage 5</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Remaining Manpower</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.5</td>
<td>0.2</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cell Loading

Cell loading is also an important issue to resolve in this study. For example, assume that there are 2 types of products (that require 10 buckets each) to be assigned to 2 cells. If cell loading was properly fulfilled meaning that all products of type 1 are assigned to cell 1 and all products of type 2 are assigned to cell 2, then it is not a problem to put the jobs in order as it is shown in Table 9 and Table 10. However, if the cell loading was not properly fulfilled, then a schedule as shown in Table 11 may be obtained for each cell where there are 3 buckets without enough manpower capacity. As a result, cell loading needs to be addressed before starting product sequencing.
Table 9

*Product Sequencing with Just Product Type 1*

<table>
<thead>
<tr>
<th></th>
<th>Week-1</th>
<th></th>
<th>Week-2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>TU</td>
<td>W</td>
</tr>
<tr>
<td></td>
<td>am</td>
<td>pm</td>
<td>am</td>
</tr>
<tr>
<td></td>
<td>am</td>
<td>pm</td>
<td>am</td>
</tr>
<tr>
<td>t</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Products</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Stage 1</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Stage 2</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Stage 3</td>
<td>6</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Stage 4</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Stage 5</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Remaining Manpower</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 10

*Product Sequencing with Just Product Type 2*

<table>
<thead>
<tr>
<th></th>
<th>Week-1</th>
<th></th>
<th>Week-2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>TU</td>
<td>W</td>
</tr>
<tr>
<td></td>
<td>am</td>
<td>pm</td>
<td>am</td>
</tr>
<tr>
<td></td>
<td>am</td>
<td>pm</td>
<td>am</td>
</tr>
<tr>
<td>t</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Products</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Stage 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Stage 2</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Stage 3</td>
<td>6</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>Stage 4</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Stage 5</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Remaining Manpower</td>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>
Table 11

*Product Sequencing When Cell Loading was not Proper*

<table>
<thead>
<tr>
<th></th>
<th>Week-1</th>
<th></th>
<th>Week-2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>TU</td>
<td>W</td>
<td>TH</td>
</tr>
<tr>
<td></td>
<td>am</td>
<td>pm</td>
<td>am</td>
<td>pm</td>
</tr>
<tr>
<td>t</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Products</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Stage 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Stage 2</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Stage 3</td>
<td>6</td>
<td>4</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Stage 4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Stage 5</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Remaining</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Manpower</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Consequently, the problem can be summarized:

1. Determine operator assignments to operations for each product
2. Assign products to cells (cell loading)
3. Determine a sequence for the products in each cell to minimize the number of periods where available manpower is exceeded.
CHAPTER 4: METHODOLOGY

The proposed methodology consists of three phases as summarized in Table 12.

The following sections will describe the methodology in detail.

Table 12

Proposed Three Phase Methodology

<table>
<thead>
<tr>
<th>Phase</th>
<th>Objective</th>
<th>Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Manpower Allocation</td>
<td>Math Modeling</td>
</tr>
<tr>
<td>2</td>
<td>Cell Loading</td>
<td>Math Modeling Genetic Algorithms</td>
</tr>
<tr>
<td>3</td>
<td>Product Sequencing</td>
<td>Math Modeling Genetic Algorithms</td>
</tr>
</tbody>
</table>

Phase 1: Allocation of Manpower

It is assumed that production process is divided into 5 stages. Phase 1 determines manpower levels for these five stages so that the production rate is maximized. The model is given below with index, parameters and decision variables.

**Index:**

\[ i = \text{operation index} \]

**Parameters:**

\[ t_i = \text{processing time for operation } i \]

\[ U_i = \text{upper limit for the number of workers for operation } i \]

\[ T = \text{total number of workers available in a cell} \]

\[ r = \text{total number of stages} \]
**Decision Variables:**

\[ PR = \text{production rate (positive integer)} \]

\[ m_i = \text{number of workers needed for operation } i \text{ (positive integer)} \]

**Objective**

\[ \text{Maximize } z = PR \] \hspace{1cm} (1)

**Subject to**

\[ \frac{m_i}{t_i} \geq PR \quad \text{for all } i \quad (i = 0, 1 \ldots r) \] \hspace{1cm} (2)

\[ m_i \leq U_i \quad \text{for all } i \quad (i = 0, 1 \ldots r) \] \hspace{1cm} (3)

\[ \sum_{i=1}^{r} m_i \leq T \] \hspace{1cm} (4)

The objective of the model is to maximize the production rate (Eq. 1). Production rate of each operation should be greater than or equal to the production rate of the cell (Eq. 2). The 3rd equation guarantees that the number of workers assigned to operation \( i \) cannot exceed the upper limit. Finally, equation (Eq. 4) guarantees that the total number of workers for that cell is not exceeded.

The model is explained with an example problem. All of the products have 5 stages \((r = 5)\). The upper limit for the number of workers for any operation is considered to be 7 \((i.e. \ U_i = 7)\). Also, there are 20 workers in each cell \((T = 20)\). Table 13 shows the processing times for product 1.
Table 13

*Operation Times for Product 1 (Hour)*

<table>
<thead>
<tr>
<th>Operations</th>
<th>Product 1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>1</td>
<td>0.6</td>
<td>1</td>
<td>0.8</td>
<td>1.2</td>
</tr>
</tbody>
</table>

The mathematical model should be solved for each product one by one (independently). The manpower levels for product 1 are given in Table 14. According to these results, 3, 5, 4, 6 and 2 workers are assigned to operations 1, 2, 3, 4 and 5, respectively. The production rate for this example is 5 units / hour. Also the codes for OPL for product 1 are given in APPENDIX A.

Table 14

*Results: Manpower Levels for Product 1*

<table>
<thead>
<tr>
<th>Operations</th>
<th>Product 1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>PR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>5 units/ hour</td>
</tr>
</tbody>
</table>
Phase 2: Cell Loading

In the second phase, products are assigned to cells. Assignment to cells is done considering product similarity by using:

a. Linear mathematical modeling

b. Genetic algorithms

Products are assigned to cells based on product similarity. Similarity coefficient developed by Süer and Ortega (1994) is used to determine similarity among products.

Similarity between products \( i \) and \( k \) is calculated using the equation 5 below. If there are \( r \) stages (\( s \) is the index for stage number):

\[
S_{ik} = \frac{\sum_{s=1}^{r} \min (m_{is}, m_{ks})}{\sum_{s=1}^{r} \max (m_{is}, m_{ks})} \tag{5}
\]

Since there are \( n \) products, a \( nxn \) matrix will be obtained at the end which is symmetrical, i.e., \( S_{ij} = S_{ji} \). For example: \( S_{12} = S_{21} \). To illustrate the calculation of similarity coefficient between products 1 and 2, assume that manpower levels are given as in Table 15 below.
Table 15

*Manpower Levels for Products 1 and 2*

<table>
<thead>
<tr>
<th>Operations</th>
<th>Products</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

So, the calculation is given below:

\[
S_{12} = \frac{\min(3,5) + \min(5,3) + \min(4,4) + \min(6,4) + \min(2,4)}{\max(3,5) + \max(5,3) + \max(4,4) + \max(6,4) + \max(2,4)} = \frac{3 + 3 + 4 + 4 + 2}{5 + 5 + 4 + 6 + 4} = \frac{16}{24} = 0.667
\]

A modified version of p-median model is used to assign products to cells after determining product similarity. It was developed in order to deal with distance problems. It is going to be applied to our cell loading problem.

*Assignment According To Product Similarity Using Linear Mathematical Modeling (Modified Version of P-Median Model)*

Mathematical model is given below with indices, parameters and decision variables.
**Indices:**

\[ i = \text{product index} \]

\[ k = \text{product index and cell index} \]

**Parameters:**

\[ C_{ik} = \text{similarity coefficient between product } i \text{ and } k \]

\[ N = \text{total number of cells} \]

\[ U = \text{upper limit for the number of products assigned to a cell} \]

**Decision Variables:**

\[ x_{ik} = 1 \text{ if product } i \text{ is assigned to cell } k, 0 \text{ otherwise} \]

(if \( x_{ii} \\
\) = 1, it means that cell \( i \) is opened to assign a product and also it means that product \( i \) assigned to cell \( i \))

**Objective**

Maximize \( z = \sum_{i=1}^{n} \sum_{k=1}^{n} C_{ik} \cdot x_{ik} \) \hspace{1cm} (6)

**Subject to**

\[ \sum_{k=1}^{n} x_{ik} = 1 \text{ for all } i \] \hspace{1cm} (7)

\[ \sum_{i=1}^{n} x_{ik} \leq U \text{ for all } k \] \hspace{1cm} (8)

\[ x_{ik} \leq x_{kk} \text{ for all } i \text{ and for all } k \] \hspace{1cm} (9)
The objective function maximizes the similarity of products assigned to cells (Eq. 6). Equation (7) ensures that all of the products are assigned to a cell. Equation (8) guarantees that no more than $U$ products are assigned to a cell. Equation (9) shows, no product can be assigned to a cell until that cell is opened. Lastly, according to equation (10), the number of cells equals to $N$. To illustrate the model, a simple problem is going to be solved. Consider that there are 10 ($n=10$) products, and 2 cells ($N=2$) with a capacity of 5 products each ($U=5$).

Table 16

<table>
<thead>
<tr>
<th>Operations</th>
<th>Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op. 1</td>
<td>1 2 3 4 5 6 7 8 9 10</td>
</tr>
<tr>
<td>Op. 2</td>
<td>5 3 2 6 2 2 4 3 3</td>
</tr>
<tr>
<td>Op. 3</td>
<td>4 4 2 6 5 2 5 4 5 3</td>
</tr>
<tr>
<td>Op. 4</td>
<td>6 4 8 3 5 8 5 4 5 3</td>
</tr>
<tr>
<td>Op. 5</td>
<td>2 4 5 2 5 5 4 3 3</td>
</tr>
</tbody>
</table>

If the manpower levels are given as in Table 16, then similarity coefficients for 10 products are calculated using equation (5). So, the similarity matrix is obtained as in Table 17.
Table 17

Similarity Matrix for 10 Products

<table>
<thead>
<tr>
<th>PRODUCTS</th>
<th>PRODUCTS</th>
<th>PRODUCTS</th>
<th>PRODUCTS</th>
<th>PRODUCTS</th>
<th>PRODUCTS</th>
<th>PRODUCTS</th>
<th>PRODUCTS</th>
<th>PRODUCTS</th>
<th>PRODUCTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.67</td>
<td>0.6</td>
<td>0.7</td>
<td>0.63</td>
<td>0.6</td>
<td>0.63</td>
<td>0.74</td>
<td>0.77</td>
</tr>
<tr>
<td>2</td>
<td>0.67</td>
<td>1</td>
<td>0.6</td>
<td>0.56</td>
<td>0.7</td>
<td>0.6</td>
<td>0.7</td>
<td>0.9</td>
<td>0.77</td>
</tr>
<tr>
<td>3</td>
<td>0.6</td>
<td>0.6</td>
<td>1</td>
<td>0.39</td>
<td>0.7</td>
<td>1</td>
<td>0.7</td>
<td>0.6</td>
<td>0.63</td>
</tr>
<tr>
<td>4</td>
<td>0.7</td>
<td>0.56</td>
<td>0.39</td>
<td>1</td>
<td>0.58</td>
<td>0.39</td>
<td>0.58</td>
<td>0.63</td>
<td>0.65</td>
</tr>
<tr>
<td>5</td>
<td>0.63</td>
<td>0.7</td>
<td>0.7</td>
<td>0.58</td>
<td>1</td>
<td>0.7</td>
<td>1</td>
<td>0.7</td>
<td>0.81</td>
</tr>
<tr>
<td>6</td>
<td>0.6</td>
<td>0.6</td>
<td>1</td>
<td>0.39</td>
<td>0.7</td>
<td>1</td>
<td>0.7</td>
<td>0.6</td>
<td>0.63</td>
</tr>
<tr>
<td>7</td>
<td>0.63</td>
<td>0.7</td>
<td>0.7</td>
<td>0.58</td>
<td>1</td>
<td>0.7</td>
<td>1</td>
<td>0.7</td>
<td>0.81</td>
</tr>
<tr>
<td>8</td>
<td>0.74</td>
<td>0.9</td>
<td>0.6</td>
<td>0.63</td>
<td>0.7</td>
<td>0.6</td>
<td>0.7</td>
<td>1</td>
<td>0.77</td>
</tr>
<tr>
<td>9</td>
<td>0.77</td>
<td>0.77</td>
<td>0.63</td>
<td>0.65</td>
<td>0.81</td>
<td>0.63</td>
<td>0.81</td>
<td>0.77</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0.54</td>
<td>0.74</td>
<td>0.48</td>
<td>0.5</td>
<td>0.48</td>
<td>0.5</td>
<td>0.67</td>
<td>0.63</td>
<td>1</td>
</tr>
</tbody>
</table>

After solving the model, products 3, 5, 6, 7, and 9 are assigned to the first cell and products 1, 2, 4, 8, and 10 are assigned to the second cell. As a result, the objective function value is 6.1364. OPL code for the model is given in APPENDIX B.

Assignment According To Product Similarity Using Genetic Algorithms

The same problem is solved using genetic algorithms and the results are compared with the results obtained from the mathematical model. A program in C# has been written in order to solve the problem with genetic algorithm.

First of all, gene and chromosome representation needs to be determined. The representation for a chromosome with 10 genes is shown in Figure 12. Each number corresponds to a gene and 10 genes come together to create a chromosome. A
chromosome represents a schedule. For example in Figure 12, it represents 10-job schedule for 2 cells (cell 0 and cell 1).

![1 0 1 1 0 0 0 1 0 1]

*Figure 12: Chromosome Representation for Cell Loading.*

According to Figure 12, products 0,2,3,7 and 9 are assigned to the 0th cell and the remaining products are assigned to the 1st cell. As it can be seen, number of cells start from 0 in order to avoid confusion and make the programming simpler.

The important thing here is that the number of products must be multiples of number of cells. Otherwise, the program will give an error as “MAKE SURE THAT NUMBER OF PRODUCTS CAN BE DIVIDED INTO NUMBER OF CELLS”.

The “population size” is the total number of chromosomes in a generation, and the “number of generations” shows the total number of generations that the program will work. The interface of the program is shown in Figure 13.
Genetic algorithm used for cell loading problem consists of steps summarized below:

1. Create initial population
2. Calculate reproduction probabilities
3. Perform Reproduction
4. Perform Mutation
5. Do Selection
6. Create new population and continue from step 2

A program in C# has been written in order to apply these steps to the problem. These steps are explained in detail along with the explanation of the program below. These steps are also illustrated on Figure 14.
Creating Initial Population

Initial population is created totally random. So, for a 10 product, 2 cell problem with a population size of 100 in each generation, program creates 100 random chromosomes which look like the one in Figure 12.

Then, fitness is calculated for each chromosome. For example, for the chromosome in Figure 12, fitness is calculated with the help of similarity table in Table 17 as shown in Table 18. Table 18-a shows the similarity coefficients between the products in the first cell and Table 18-b shows the similarity coefficients between the products in the second cell. Finally fitness is measured as the sum of similarity coefficients in both tables.

Figure 14: GA for Cell Loading.
Table 18

*Calculation of Fitness for a Specific Chromosome*

<table>
<thead>
<tr>
<th>Products</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Products</strong></td>
<td><strong>Cell1/cell2</strong></td>
<td><strong>1</strong></td>
<td><strong>0</strong></td>
<td><strong>1</strong></td>
<td><strong>1</strong></td>
<td><strong>0</strong></td>
<td><strong>0</strong></td>
<td><strong>0</strong></td>
<td><strong>1</strong></td>
<td><strong>0</strong></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.6</td>
<td>0.7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.74</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.6</td>
<td>0</td>
<td>0</td>
<td>0.39</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.6</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0.7</td>
<td>0</td>
<td>0.39</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.63</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0.74</td>
<td>0</td>
<td>0.6</td>
<td>0.63</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0.54</td>
<td>0</td>
<td>0.48</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.67</td>
</tr>
</tbody>
</table>

Table 18-a

<table>
<thead>
<tr>
<th>Products</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Products</strong></td>
<td><strong>Cell1/cell2</strong></td>
<td><strong>0</strong></td>
<td><strong>1</strong></td>
<td><strong>0</strong></td>
<td><strong>0</strong></td>
<td><strong>1</strong></td>
<td><strong>1</strong></td>
<td><strong>1</strong></td>
<td><strong>0</strong></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0.7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0.6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.7</td>
<td>0</td>
<td>0.7</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0.7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.7</td>
<td>0</td>
<td>0</td>
<td>0.81</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>0.77</td>
<td>0</td>
<td>0</td>
<td>0.81</td>
<td>0.63</td>
<td>0.81</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 18-b

**SUM =** **FITNESS**

26.47727

Table 18-c
The similarity matrix is going to be different for every problem. For example, when a product is added or removed, the program needs to calculate it again. So, whenever, the program starts, it first calculates the similarity matrix and then, it calculates fitness using that matrix.

*Calculating reproduction probabilities*

The reproduction probabilities are assigned based on fitness values and objective function. For example, consider 3 chromosomes and fitness as given in Table 19-a. The reproduction probabilities are calculated by dividing fitness values into sum of fitness values for the maximization objective (see Table 19-a). On the other hand, for the minimization objective, 1/fitness values are calculated for each chromosome, and later they are divided into sum of 1/fitness values to find reproduction probability for each chromosome (see Table 19-b).
Table 19

*Calculating Reproduction Probabilities*

<table>
<thead>
<tr>
<th></th>
<th>Fitness</th>
<th>Reproduction probability for maximizing objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>chromosome 1</td>
<td>28.57425</td>
<td>0.348412867</td>
</tr>
<tr>
<td>chromosome 2</td>
<td>26.50009</td>
<td>0.323122043</td>
</tr>
<tr>
<td>chromosome 3</td>
<td>26.93828</td>
<td>0.32846509</td>
</tr>
</tbody>
</table>

| Sum            | 82.01262|                                                   |

Table 19-a

<table>
<thead>
<tr>
<th></th>
<th>Fitness</th>
<th>1/fitness</th>
<th>Reproduction probability for minimizing objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>chromosome 1</td>
<td>28.57425</td>
<td>0.03499654</td>
<td>0.318572746</td>
</tr>
<tr>
<td>chromosome 2</td>
<td>26.50009</td>
<td>0.03773573</td>
<td>0.343507496</td>
</tr>
<tr>
<td>chromosome 3</td>
<td>26.93828</td>
<td>0.03712189</td>
<td>0.337919758</td>
</tr>
</tbody>
</table>

| Sum            |         | 0.10985416|                                                   |

Table 19-b

Reproduction

After determination of reproduction probabilities, a random number is drawn for each chromosome and checked against cumulative probabilities in order to generate new population. For example, if we had 3 chromosomes with the reproduction probabilities as calculated in Table 19-a, then, cumulative reproduction probabilities for the objective function of maximization type are calculated as in Table 20.
Table 20

*Cumulative Reproduction Probabilities*

<table>
<thead>
<tr>
<th>Chromosome</th>
<th>Reproduction Probability</th>
<th>Cumulative Reproduction Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>chromosome 1</td>
<td>28.57425</td>
<td>0.348412867</td>
</tr>
<tr>
<td>chromosome 2</td>
<td>26.50009</td>
<td>0.67153491</td>
</tr>
<tr>
<td>chromosome 3</td>
<td>26.93828</td>
<td>1</td>
</tr>
</tbody>
</table>

Three random numbers are drawn. For instance, if the random numbers are .22, .25 and .80, then new population will have two of chromosome 1 and one of chromosome 3.

In the program, during reproduction, the number of chromosomes is doubled. So, for instance, for 3 chromosomes above, 6 random numbers are drawn to increase the population size to twice of its size (During selection, half of them are selected, so population size remains the same every generation).

**Mutation**

Mutation probability is directly entered between 0-1 to the mutation probability box. For every gene, a random number is drawn and if it smaller than the mutation probability, then mutation is applied to that gene. Two mutation strategies are considered, namely, reciprocal exchange mutation and insertion mutation. In the GA program, the user can select either one. Reciprocal exchange mutation is the default method unless it is changed as shown in Figure 15.
Reciprocal Exchange mutation

Two positions are chosen randomly and swapped. The first gene was chosen according to mutation probability. The second gene is chosen among all genes randomly. So, every gene has the probability of $1 / \text{(total number of genes)}$ to be selected. So, it is even possible to choose the same gene which will force the chromosome to stay the same. For instance, for the chromosome in Figure 12, a reciprocal exchange mutation may be applied as shown in Figure 16.

Before mutation

\[
\begin{array}{cccccccc}
1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
\end{array}
\]

Mutated chromosome

\[
\begin{array}{cccccccc}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
\end{array}
\]

Figure 16: Reciprocal Exchange Mutation.
**Insertion Mutation**

One gene is chosen randomly and it is inserted in a random position. The first gene is chosen according to mutation probability and the position is chosen among all positions by drawing a random number. So, for instance, for the chromosome in Figure 12, an insertion mutation may be applied as shown in Figure 17.

**Before Mutation**

```
1 0 1 1 0 0 0 1 0 1
```

**Mutated Chromosome**

```
1 1 0 1 0 0 0 1 0 1
```

*Figure 17: Insertion Mutation.*

**Selection**

Population size is doubled during reproduction, so half of them will be able to go to the next generation. Selection is done among the best ones (Best half of doubled population size). So, fitness values of the chromosomes are calculated after mutation and the ones with better fitness values create the next generation. Later, program goes to the 2\textsuperscript{nd} step and continues from there until it reaches the last generation.

The best chromosome obtained so far is kept in the memory. It is updated every generation until the program reaches the last generation, and it is reported as the best chromosome at the completion of run.
The results obtained from the program for the same problem for which mathematical model was solved is given in Figure 18. As one can see, the optimal result (28.574) is obtained in the 13th generation. According to that chromosome, products 0, 1, 3, 7, and 9 are assigned to the first cell and the remaining products are assigned to the second cell. The results are presented based on 500 generations and a population size of 100. The total execution time was 0.5156 seconds.

![Objective](image)

**Objective**: Maximize Total Similarity

**Similarity**: 28.574252694697

**Generation Best Fitness Found**: 13

**Execution Time**: 0.515625

**Best Chromosome**: 1 1 0 1 0 1 0 0 1 0 1

*Figure 18: Cell Loading Results for the Example Problem.*

After determining which products go to which cell, clicking “Send Results to Product Sequencing Program” (Figure 19) open up two product sequencing programs (number of opened programs depends on the number of cells) which has the assigned products in with the default values entered as shown in Figure 20 and Figure 21.
Figure 19: Sending Assigned Products to Cells.

Figure 20: Products Assigned to Cell 1.
Phase 3: Product Sequencing

After assigning products to cells in phase 2, the products in each cell should be scheduled. The problem is going to be solved in two ways; using mathematical model and using genetic algorithms.

Product Sequencing Using Mathematical Model

The objective is to determine the sequence for products 1, 2, 3...10 (n=10, K=14) such that number of periods in which available capacity is exceeded in as less as possible. The mathematical model to solve the problem is given below with indices, parameters and decision variables.

Indices:

\[ i = \text{product index} \]

\[ k = \text{total period index} \]
**Parameters:**

\( m_{si} = \) manpower levels for stage \( s \) of product \( i \)

\( E_k = \) total extra manpower levels for coming periods or previous periods that are already assigned (for example for the problem in table 6, \( E_1 \)

\[ = 4 + 6 + 3 + 2 = 15; \]

\( E_2 = 4 + 4 + 3 = 11; E_3 = 5 + 2 = 7; E_4 = 4; E_{5,6,7,8,9,10} = 0; E_{11} = 3.3; \)

\( E_{12} = 3.3 + 3.2 = 6.5; E_{13} = 3.3 + 3.2 + 4.1 = 10.6; E_{14} = 3.3 + 3.2 + 4.1 + 4.6 \)

\[ = 15.2 \]

\( T = \) Total manpower available in a cell

\( n = \) number of products (assumed to be 10) \( (i = 1, 2, 3 \ldots 10) \)

\( K = \) total number of periods \( (n + 4)(1, 2, 3 \ldots 14) \)

**Decision Variables:**

\( x_{ij} = 1 \) if product \( i \) is assigned to period \( j \), 0 otherwise

\( t_k = \) total load of a period

\( s_k = 1 \) if shortage occurs in manpower level in period \( k \), 0 otherwise

**Objective**

Minimize

\[ z = \sum_{k=1}^{K} s_k \]  \hspace{1cm} (16)
subject to

\[ \sum_{j=1}^{n} x_{ij} = 1 \quad \text{for all } i \quad (17) \]

\[ \sum_{i=1}^{n} x_{ij} = 1 \quad \text{for all } i \quad (18) \]

\[ 1000 * s_k \geq t_k - T \quad \text{for all } k \quad (19) \]

\[ \sum_{i=1}^{n} m_{1i} * x_{i1} + E_1 = t_1 \quad (20) \]

\[ \sum_{i=1}^{n} m_{2i} * x_{i1} + \sum_{i=1}^{n} m_{1i} * x_{i2} + E_2 = t_2 \]

\[ \sum_{i=1}^{n} m_{3i} * x_{i1} + \sum_{i=1}^{n} m_{2i} * x_{i2} + \sum_{i=1}^{n} m_{1i} * x_{i3} + E_3 = t_3 \]

\[ \sum_{i=1}^{n} m_{4i} * x_{i1} + \sum_{i=1}^{n} m_{3i} * x_{i2} + \sum_{i=1}^{n} m_{2i} * x_{i3} + \sum_{i=1}^{n} m_{1i} * x_{i4} + E_4 = t_4 \]

\[ \sum_{i=1}^{n} m_{5i} * x_{ij} + \sum_{i=1}^{n} m_{4i} * x_{i(j+1)} + \sum_{i=1}^{n} m_{3i} * x_{i(j+2)} + \sum_{i=1}^{n} m_{2i} * x_{i(j+3)} + \sum_{i=1}^{n} m_{1i} * x_{i(j+4)} + E_{(j+4)} = t_{(j+4)} \quad \text{for all } j = 1,2,3,4,5,6 \]

\[ \sum_{i=1}^{n} m_{5i} * x_{i7} + \sum_{i=1}^{n} m_{4i} * x_{i8} + \sum_{i=1}^{n} m_{3i} * x_{i9} + \sum_{i=1}^{n} m_{2i} * x_{i10} + E_{11} = t_{11} \]
The objective function minimizes the number of time buckets where available manpower is not sufficient. Equation (17) assures that every product is assigned to a bucket. Equation (18) guarantees that each period has a product assigned to it. According to equation (19), if the total load of a bucket exceeds $T$, $s_k$ value will be 1. All the equations starting with (20) are used to calculate the total load of that period.

To illustrate the model, the data in Table 16 is used. The OPL codes for the model are given in APPENDIX C. The final sequence obtained using mathematical model is: 9-1-3-4-6-5-10-7-2-8. The objective function value is 0. Therefore, there are no shortages in any period with respect to manpower. The final sequence and remaining manpower are given in Table 21.
Table 21

**Final Sequence Obtained Using Math Modelling**

<table>
<thead>
<tr>
<th></th>
<th>Week-1</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M 1</td>
<td>M 2</td>
<td>T 1</td>
<td>T 2</td>
<td>M 3</td>
<td>T 3</td>
<td>M 4</td>
<td>T 4</td>
<td>M 5</td>
<td>T 5</td>
<td>M 6</td>
<td>T 6</td>
<td>M 7</td>
</tr>
<tr>
<td>am pm</td>
<td>am pm</td>
<td>am pm</td>
<td>am pm</td>
<td>am pm</td>
<td>am pm</td>
<td>am pm</td>
<td>am pm</td>
<td>am pm</td>
<td>am pm</td>
<td>am pm</td>
<td>am pm</td>
<td>am pm</td>
<td>am pm</td>
</tr>
<tr>
<td><strong>t</strong></td>
<td>1 2</td>
<td>3 4</td>
<td>5 6</td>
<td>7 8</td>
<td>9 10</td>
<td>11 12</td>
<td>13 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Products</strong></td>
<td>9 1</td>
<td>3 4</td>
<td>6 5</td>
<td>10 7</td>
<td>2 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Stage 1</strong></td>
<td>3 3</td>
<td>2 2</td>
<td>8 2</td>
<td>5 4</td>
<td>3.5 3.5 3.5 3.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Stage 2</strong></td>
<td>4 3</td>
<td>5 2</td>
<td>6 2</td>
<td>2 3</td>
<td>2 3</td>
<td>4 3.3 3.3 3.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Stage 3</strong></td>
<td>6 4</td>
<td>5 4</td>
<td>2 6</td>
<td>2 5</td>
<td>3 5</td>
<td>4 4 4.2 4.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Stage 4</strong></td>
<td>3 4</td>
<td>5 5</td>
<td>6 8</td>
<td>3 8</td>
<td>5 3</td>
<td>5 4 4 4.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Stage 5</strong></td>
<td>2 3</td>
<td>4 3</td>
<td>2 5</td>
<td>5 2</td>
<td>5 5</td>
<td>3 5 4 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Remaining</strong></td>
<td>2 3</td>
<td>0 3</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0.5 0.2 1.1 0.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Manpower</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Product Sequencing Using Genetic Algorithms**

The same problem is solved using genetic algorithm and the results are compared with the results obtained from the mathematical model. A program in C# has been written in order to solve the problem using genetic algorithm. In the GA program, it is possible to reach product sequencing part by clicking “Send Results to sequencing program” which was already discussed before in Figure 19, Figure 20 and Figure 21.

First of all, gene and chromosome representation needs to be determined. So, if there are 10 products, then a random chromosome representation can be shown as in Figure 22. In this figure, each number is considered as a gene.

![Figure 22: Chromosome Representation for Product Sequencing.](image-url)
This chromosome basically shows the sequence of each product. As it can be seen, numbers of products starts from 0 in order to avoid confusion and make the programming simpler.

Manpower Capacity per Period is the total workers we have for a period.

“Population size” is the total number of chromosomes in a generation, and the “number of generations” shows the total number of generations the program runs. The interface of the program is shown in Figure 23.

Figure 23: Interface of Product Sequencing Program.
Genetic algorithm used for product sequencing problem consists of the steps as summarized below:

1. Create initial population
2. Calculate reproduction probabilities
3. Do Reproduction
4. Determine Mating Partners
5. Perform Crossover (Mating)
6. Perform Mutation
7. Do Selection
8. Create new population and continue from step 2

The steps applied in product sequencing are close to the ones in cell loading. However, there are more steps here with more parameters. So, this genetic algorithm approach is expected to work better. A program in C# has been written in order to apply these steps to the problem. These steps are going to be explained in detail with the explanation of the program below. These steps are illustrated on Figure 24.
**Creating initial population**

Initial population is created totally random. For example, if the population size is 100, then 100 chromosomes are created randomly. An instance of a random chromosome was shown in Figure 22.

The fitness value for each chromosome also needs to be calculated in this part. So, for example for the random chromosome shown in Figure 22, the fitness value is calculated as shown in Table 22. So, the fitness value is 4 since manpower level was exceeded in 4 periods.
Table 22

*Fitness Calculation for Product Sequencing*

<table>
<thead>
<tr>
<th>Products</th>
<th>Stage 1</th>
<th>Stage 2</th>
<th>Stage 3</th>
<th>Stage 4</th>
<th>Stage 5</th>
<th>Remaining Manpower</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5 8 2 3 4 2 3 2 3 3 3</td>
<td>4 3 3 6 2 4 2 3 2 5</td>
<td>6 4 4 3 6 2 4 5 5 4</td>
<td>3 4 5 4 3 3 8 4 5 5</td>
<td>2 3 2 4 3 2 5 4 5</td>
<td>0 -2 4 0 1 6 1 1 1 -3</td>
</tr>
<tr>
<td>Week-1</td>
<td>M</td>
<td>Tu</td>
<td>W</td>
<td>Th</td>
<td>Fr</td>
<td>M</td>
</tr>
<tr>
<td>am</td>
<td>pm</td>
<td>am</td>
<td>pm</td>
<td>am</td>
<td>pm</td>
<td>am</td>
</tr>
<tr>
<td>t</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>Products</td>
<td>1</td>
<td>9</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>Stage 1</td>
<td>5</td>
<td>8</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Stage 2</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Stage 3</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Stage 4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Stage 5</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Remaining Manpower</td>
<td>0</td>
<td>-2</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>Sum =</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Calculating Reproduction Probabilities

The reproduction probabilities are assigned based on fitness values. For example, consider 3 chromosomes and fitness as given in Table 23 (Fitness values are added 1 in order to avoid division by 0). So, reproduction probability for each chromosome equals to $1/(\text{fitness}+1)$ divided by the sum of $1/(\text{fitness}+1)$ values of every chromosomes.

Table 23

*Calculating Reproduction Probabilities for Product Sequencing*

<table>
<thead>
<tr>
<th>chromosome</th>
<th>Fitness</th>
<th>1/(fitness+1)</th>
<th>Reproduction Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>chromosome 1</td>
<td>3</td>
<td>0.25</td>
<td>0.263157895</td>
</tr>
<tr>
<td>chromosome 2</td>
<td>4</td>
<td>0.2</td>
<td>0.210526316</td>
</tr>
<tr>
<td>chromosome 3</td>
<td>1</td>
<td>0.5</td>
<td>0.526315789</td>
</tr>
<tr>
<td>Sum =</td>
<td></td>
<td>0.95</td>
<td></td>
</tr>
</tbody>
</table>
Reproduction

After determining reproduction probabilities, a random number is drawn for each chromosome and checked against cumulative probabilities in order to generate new population. For example, if we had 3 chromosomes with the reproduction probabilities as calculated in Table 23, then, cumulative reproduction probabilities are calculated as in Table 24.

Table 24

*Cumulative Reproduction Probabilities for Product Sequencing*

<table>
<thead>
<tr>
<th>Chromosome</th>
<th>Reproduction Probability</th>
<th>Cumulative Reproduction Probabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>chromosome 1</td>
<td>0.263157895</td>
<td>0.263157895</td>
</tr>
<tr>
<td>chromosome 2</td>
<td>0.210526316</td>
<td>0.473684211</td>
</tr>
<tr>
<td>chromosome 3</td>
<td>0.526315789</td>
<td>1</td>
</tr>
</tbody>
</table>

Three numbers are drawn to create new population with the same number of chromosomes. For example, if the random numbers are .1, .66, and .61, then new population will consist of one of chromosome 1 and two of chromosome 3.

As a result, new population will have equal number of chromosomes with the initial population.

Determining Mating Partners

Mating partners are determined using the same way with reproduction. The only difference is, this time different random numbers are drawn. Consequently there are two populations created from the initial population: 1. New population 2. Mating partners
Crossover (Mating)

Every new population member has a match in the mating partners population. Mating is done according to crossover probability and it is entered between 0-1 to crossover probability box in the program. A random number is drawn and if it is smaller than the crossover probability, then mating is done between new population member and its match in mating partner population, otherwise, they remain the same.

New population and mating partners are mated using two crossover strategies which are position-based crossover and order crossover. It is possible to select any of these crossover strategies from options-crossover strategy tab from the program as shown in Figure 25. The default strategy is position-based crossover strategy, if not changed.

Figure 25: Crossover Strategies for Product Sequencing.

Position-Based Crossover

Two offspring are created from two parents (New population member and mating partner). First step is to select the set of positions from new population member at random. To do that, the probability number between 0-1 which was entered in Position-
based probability box is used. If the random number drawn is smaller than the position- 
based crossover probability number, then that position is chosen for crossover. This is 
repeated for every position until all of the set of positions are determined.

The second step is to copy the values on the chosen set of positions to offspring 1. 
In the third step, these copied values are deleted from the second parent (mating partner). 
Later, remaining values on second parent are placed into offspring 1 from left to right 
according to sequence. This procedure is illustrated in Figure 26. The second offspring is 
created the same way from the second parent (mating partner).
STEP 1:
New population Member

1 9 3 5 7 4 8 6 0 2

Mating Partner

8 0 6 5 9 2 3 4 7 1

STEP 2:
New population Member

1 9 3 5 7 4 8 6 0 2

Mating Partner

8 0 6 5 9 2 3 4 7 1

STEP 3:
New population Member

1 9 3 5 7 4 8 6 0 2

Mating Partner

8 0 6 5 9 2 3 4 7 1

STEP 4:
New population Member

1 9 3 5 7 4 8 6 0 2

Mating Partner

8 0 6 5 9 2 3 4 7 1

Figure 26: Position-Based Crossover.
Order Crossover

Two parents are going to create two offspring (New population member and mating partner) again. First of all, a substring is selected from new population member at random. Then in the second step, the values on the chosen substring are copied to offspring 1. In the third step, these copied values are found and deleted from the second parent (mating partner). In the fourth step, left values on the second parent are copied into offspring 1 from left to right based on their order. This procedure is illustrated in Figure 27. The second offspring is created the same way this time using the second parent (mating partner).
STEP 1:
New population Member

1 9 3 5 7 4 8 6 0 2

Mating Partner

8 0 6 5 9 2 3 4 7 1

STEP 2:
New population Member

1 9 3 5 7 4 8 6 0 2

Mating Partner

8 0 6 5 9 2 3 4 7 1

STEP 3:
New population Member

1 9 3 5 7 4 8 6 0 2

Mating Partner

8 0 6 5 9 2 3 4 7 1

STEP 4:
New population Member

1 9 3 5 7 4 8 6 0 2

Mating Partner

8 0 6 5 9 2 3 4 7 1

Offspring 1

3 5 7 4 8

Offspring 1

3 5 7 4 8

Offspring 1

0 6 3 5 7 4 8 9 2 1

Figure 27: Order Crossover.
**Mutation**

There are 2 populations which were created in steps 0 and 0 before the crossover:
1. New population and 2. Mating partners. After the crossover, the population size is doubled (equals to new population size + mating partners’ population size). This newly generated population is going through mutation in this part. There are two kinds of mutations applied:

1. Reciprocal Exchange Mutation
2. Insertion Mutation

It is possible to choose the mutation type to work with through “options – mutation” as shown in Figure 28. Mutation probability is also needed in order to give the decision of mutation for every gene. For example, if the randomly generated number is smaller than mutation probability for a specific gene, then that gene will go through mutation. That mutation probability is also entered on the program in the “mutation probability box”.

![Product Sequencing](image)

*Figure 28: Mutation Types for Product Sequencing.*
Reciprocal exchange mutation and insertion mutation were described under 4.2.4.4 before, so, it is not described here again.

**Selection**

Population size was doubled during crossover, so half of them will be able to go to the next generation. For this purpose, two selection strategies have been chosen:

1. **Roulette Wheel**

2. **Best**

**Roulette wheel**

For this selection, all the chromosomes are assigned a probability according to their fitness and half of them are chosen randomly according to these probability values.

**Best**

For this selection, better chromosomes according to their fitness value go to the next generation.

Later, program will go to 2nd step (calculating reproduction probabilities) and continue from there until it reaches the specified number of generations. This entire procedure constitutes one generation if mutation and crossover are both applied to every chromosome in that generation.

Moreover, for the experimentation purpose, the total number of generations is divided into 5 phases as shown in Figure 29.

1. In the first phase, chromosomes go through only crossover.

2. In the second phase, the population is divided into two and the first part goes through crossover and mutation while the second part goes through just crossover.
3. In the third phase, population is divided into three and the first part goes through just mutation, the second part goes through crossover and mutation and the last part goes through just crossover.

4. In the fourth phase, the population is divided into two and the first part goes through just mutation while the second part goes through crossover and mutation.

In the fifth phase, the entire population goes through just mutation.

The important thing here is that sum of percent values for divided populations in every phase equal to 100. Otherwise, program will give this error: “MAKE SURE THAT SUM EQUALS TO POPULATION SIZE”.

It is also important that the sum of percent values of phase generations equal to 100. Otherwise, program will give this error: “MAKE SURE THAT SUM OF PHASE GENERATIONS EQUAL TO THE NUMBER OF GENERATIONS”.

On the other hand, if “no phase” is clicked on the program, then program applies crossover and mutation to every chromosome in every generation based on crossover and mutation probabilities.

It is also possible to change random population percentage of the population by entering a number between 0 and 100 to the random population percentage box which is also called “Immigration”. If that number is 100, then it means that the GA program will generate just random solutions in every generation. So, it is better to keep it low. However, keeping it 0 sometimes may lead the population to repeat itself.
To illustrate the approach better, the same problem which was solved with mathematical model is solved here as well. The data in Table 16 is used as before with the values entered in the program as shown in Figure 30. As a result, the final sequence obtained using genetic algorithm is: 8 - 0 - 5 - 3 - 2 - 4 - 9 - 6 - 1 – 7, and the objective function value is 0. Therefore, there are no shortages in any period with respect to manpower. As one can see, the best chromosome was found at the 25th generation and execution time was 0.8125 seconds only. After getting the final sequence, the schedule needs to be prepared in the form of a table. It is enough to click “Send Results to Excel” button. When it is clicked, the final results table is obtained as shown in Table 25.

**Figure 29: Phases.**
**Figure 30:** The data in table 15 entered in the program.

### Table 25

**Final Sequence Obtained Using GA**

<table>
<thead>
<tr>
<th>Periods = t</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage / Products</td>
<td>8</td>
<td>0</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>9</td>
<td>6</td>
<td>1</td>
<td>7</td>
<td>3.5</td>
<td>3.5</td>
<td>3.5</td>
<td>3.5</td>
</tr>
<tr>
<td>Stage 1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>3.5</td>
<td>3.5</td>
<td>3.5</td>
<td>3.5</td>
</tr>
<tr>
<td>Stage 2</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3.3</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>Stage 3</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4.2</td>
<td>4.2</td>
</tr>
<tr>
<td>Stage 4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>3</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4.8</td>
</tr>
<tr>
<td>Stage 5</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Remaining Manpower</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0.2</td>
<td>1.1</td>
<td>0.2</td>
<td></td>
</tr>
</tbody>
</table>
There is also a step by step option in the program. Therefore, it is possible to see the change in the population during every generation. To illustrate that, let’s solve the same problem with a smaller population size for a small number of generations (for example, 10 for both). Each step is given in Figure 31, Figure 32, Figure 33, Figure 34, Figure 35, and Figure 36. As one can see, at the end of the first generation, the best chromosome so far has the fitness of two. So, we need to go more to decrease the objective value (fitness).
**Figure 31: Step 1.**
Figure 32: Step 2.
Figure 33: Steps 3 and 4.
Figure 34: Steps 5 and 6.
Figure 35: Step 7.
Figure 36: Step 2 (Next Generation).
Cell Loading and Product Sequencing Using Genetic Algorithms

One of the great advantages of the genetic algorithm is that it is sometimes possible to combine two problems together. This problem type is one of those cases. Therefore, a genetic algorithm approach is developed to do both cell loading and product sequencing at the same time by just changing the chromosome representation and the fitness function. The rest of the approach is the same as the product sequencing approach.

The chromosome is divided into as many parts as the number of cells. For example, the chromosome representation for a 20 product problem with 2 cells (10 products in each cell) is given in Figure 37.

<table>
<thead>
<tr>
<th>First cell</th>
<th>Second cell</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 3 15 16 2 19 7 14 11 12 10 4 0 13 18 17 6 1 9 8</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 37: Chromosome Representation for Cell Loading and Product Sequencing.*

The fitness function is the sum of periods where available manpower in each cell is exceeded. For the chromosome given above, it is going to be 6 since the manpower level exceeded in each cell is 3 (Sum of them is 6) as shown in Table 26 and Table 27.
Table 26

*Fitness Calculation for Cell 1*

<table>
<thead>
<tr>
<th>CELL1</th>
<th>Periods = t</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage / Products</td>
<td></td>
<td>5</td>
<td>3</td>
<td>15</td>
<td>16</td>
<td>2</td>
<td>19</td>
<td>7</td>
<td>14</td>
<td>11</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stage 1</td>
<td></td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>4</td>
<td>3</td>
<td>4.2</td>
<td>4.2</td>
<td>4.2</td>
<td>4.2</td>
</tr>
<tr>
<td>Stage 2</td>
<td></td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3.6</td>
<td>3.6</td>
</tr>
<tr>
<td>Stage 3</td>
<td></td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3.8</td>
<td>3.8</td>
</tr>
<tr>
<td>Stage 4</td>
<td></td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>8</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>4.9</td>
</tr>
<tr>
<td>Stage 5</td>
<td></td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Remaining Manpower</td>
<td></td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>-1</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>-9</td>
<td>0</td>
<td>4</td>
<td>0.8</td>
<td>2.2</td>
<td>-4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Table 27

*Fitness Calculation for Cell 2*

<table>
<thead>
<tr>
<th>CELL2</th>
<th>Periods = t</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage / Products</td>
<td></td>
<td>10</td>
<td>4</td>
<td>0</td>
<td>13</td>
<td>18</td>
<td>17</td>
<td>6</td>
<td>1</td>
<td>9</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stage 1</td>
<td></td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>8</td>
<td>3</td>
<td>3.6</td>
<td>3.6</td>
<td>3.6</td>
<td>3.6</td>
</tr>
<tr>
<td>Stage 2</td>
<td></td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3.8</td>
<td>3.8</td>
<td>3.8</td>
</tr>
<tr>
<td>Stage 3</td>
<td></td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>7</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>4.5</td>
<td>4.5</td>
<td></td>
</tr>
<tr>
<td>Stage 4</td>
<td></td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Stage 5</td>
<td></td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Remaining Manpower</td>
<td></td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>-4</td>
<td>3</td>
<td>4</td>
<td>-3</td>
<td>-5</td>
<td>2</td>
<td>1.4</td>
<td>0.6</td>
<td>0.1</td>
<td>1.1</td>
</tr>
</tbody>
</table>

There is also a graphics option in the GA program. In this option, the program runs using the specified parameters and shows a graphic as shown in Figure 38. The Y axis of the graph shows fitness while X axis shows the current number of generations. Green line shows the best solution found until that generation, yellow line shows best
solution of the current generation, blue line shows the average solution of the current generation and lastly red line shows the worst solution of the current generation.

Figure 38: Graphic of GA.
CHAPTER 5: EXPERIMENTS & RESULTS

For experimentation purpose, phase one which is the allocation of manpower is skipped. So, experiments concerning cell loading and product sequencing are discussed in this part.

Firstly, data generation is mentioned. Later, it is proved that genetic approach is a better approach than a random approach. Then, better parameters for cell loading, product sequencing, cell loading and product sequencing are determined. Next, better population size and number of generation are determined. Later, a new GA approach (5-phase approach) is compared against traditional GA. After that, genetic algorithm runs are made based on the parameter information obtained; these results are compared with math modeling first for cell loading, then for product sequencing. Finally, problem is considered as a whole (cell loading + product sequencing) and the best method is determined among possible options.

Data set Generation

The number of cells has been assumed to be 2 throughout this experimentation. The number of stages is taken as 5. There is only one shift in a day and each work day is 8 hours with 4-hour time buckets with a lunch break in the middle. So, the number of products that can be processed in a day is 4 (2 products a cell * 2). Four different data sets have been generated for experimentation purposes:

- Data set 1: 10 Jobs (work load for half of a week)
- Data set 2: 20 Jobs (work load for a week)
- Data set 3: 40 Jobs (work load for 2 weeks)
- Data set 4: 80 Jobs (work load for a month)
It is assumed that manpower levels are already calculated for 5 stages as shown in Table 28. The only constraint used while generating manpower levels was the sum of manpower levels. The total manpower available for a cell is limited to 20. Data set 1 created for 10 jobs are given in Table 28. The demand is also important. For the data set in Table 28, it was assumed that every product needs only one bucket. This is increased for data sets 2, 3 and 4. In the 2nd data set, every product needs two buckets. In the 3rd data set, every product needs 4 buckets and in the 4th data set, every product needs 8 buckets. Data sets 2, 3 and 4 are also given in APPENDIX D.

Table 28

<table>
<thead>
<tr>
<th>Product</th>
<th>DATA SET 1</th>
<th>MANPOWER LEVELS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Stage 1</td>
<td>Stage 2</td>
</tr>
<tr>
<td>Product 0</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Product 1</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Product 2</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>Product 3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Product 4</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Product 5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Product 6</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Product 7</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Product 8</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Product 9</td>
<td>8</td>
<td>3</td>
</tr>
</tbody>
</table>

The different data sets are used for different experiments and this information is summarized in Table 29.
Why Genetic Algorithm (Traditional)? Why not just Random Solutions?

The first experiment is to show why we are using genetic algorithms instead of just looking at random solutions. It is better to first compare Traditional Genetic Algorithms against a totally random approach. In order to accomplish this experiment, GA program for Cell Loading and Product Sequencing has been used. The data set 2 (for 20 Jobs) which is given in APPENDIX D is used.

The population size is 100, number of generations is 2000, mutation type is reciprocal exchange mutation, mutation probability is 0.1, crossover type is position-based crossover, selection type is best, random population percentage is 0, position-based crossover probability is 0.1 and manpower is 20. For the random approach, 100*2000 random sequences are generated and checked against traditional GA approach. The range of results varies from 3 to 5 for random approach and from 3 to 4 for genetic algorithms. The results are given in Table 30.
Table 30

GA or Random Approach – 20 Jobs (30 replications)

<table>
<thead>
<tr>
<th></th>
<th>Frequency of 3</th>
<th>Frequency of 4</th>
<th>Frequency of 5</th>
<th>Average Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td>3.333333333</td>
</tr>
<tr>
<td>RANDOM APPROACH</td>
<td>2</td>
<td>25</td>
<td>3</td>
<td>4.033333333</td>
</tr>
</tbody>
</table>

GA can be thought as systematic randomization. What GA is trying to do is to improve population (set of solutions) quality and try to find a good solution at the end. As a conclusion, totally random approach cannot find good results as often as GA approach does.

According to Fischer test, LSD (least significant difference) is calculated as 0.231551 for 95% confidence level. The difference between means is 0.7. (0.7>0.23) So, it can be concluded that there is a significant difference between Genetic algorithm approach and the random approach. The ANOVA table and boxplot table are also given below.
One-way ANOVA: y versus x

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>SS</th>
<th>MS</th>
<th>F</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
<td>7.350</td>
<td>7.350</td>
<td>36.64</td>
<td>0.000</td>
</tr>
<tr>
<td>Error</td>
<td>58</td>
<td>11.633</td>
<td>0.201</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>59</td>
<td>18.983</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S = 0.4479   R-Sq = 38.72%   R-Sq(adj) = 37.66%

Individual 95% CIs For Mean Based on Pooled StDev

<table>
<thead>
<tr>
<th>Level</th>
<th>N</th>
<th>Mean</th>
<th>StDev</th>
<th>Pooled StDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30</td>
<td>3.3333</td>
<td>0.4795</td>
<td>(-----*-----)</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>4.0333</td>
<td>0.4138</td>
<td>(-----*-----)</td>
</tr>
</tbody>
</table>

Pooled StDev = 0.4479

---

**Figure 39:** Boxplot for GA vs. Random Approach.
Moreover, when we keep mutation probability, crossover probability, random population percentage high, and then GA gets closer to the totally random approach. On the other hand, when we use them too small, then there is not much improvement between generations. So, we need to determine the optimal parameters for GA first.

Determining GA Parameters for Cell loading

For the cell loading purpose, two methods were proposed: Genetic Algorithms and mathematical modeling. In this part, the experiments in order to determine better parameters are analyzed for cell loading. The experiments are performed using GA program on 3 data sets given in APPENDIX D. The objective is to maximize the total similarity.

There are a total of 6 combinations (2*3) as calculated in Table 31. There are two mutation methods: Reciprocal exchange mutation and insertion mutation, three mutation Probabilities: 0.1, 0.3 and 0.5. Other parameters are kept fixed which are number of cells (2), population size (100) and number of generations (400).
Table 31

**Calculation of Combinations for Cell Loading**

<table>
<thead>
<tr>
<th>Calculation of Combinations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mutation Types</strong></td>
</tr>
<tr>
<td>Exchange Mutation</td>
</tr>
<tr>
<td><strong>Mutation Probabilities</strong></td>
</tr>
<tr>
<td>0.1</td>
</tr>
<tr>
<td>0.3</td>
</tr>
<tr>
<td>0.5</td>
</tr>
<tr>
<td><strong>Number Of Cells</strong></td>
</tr>
<tr>
<td>Fixed at 2</td>
</tr>
<tr>
<td><strong>Population size</strong></td>
</tr>
<tr>
<td>Fixed at 100</td>
</tr>
<tr>
<td><strong>Number of Generations</strong></td>
</tr>
<tr>
<td>Fixed at 400</td>
</tr>
</tbody>
</table>

Thirty replications are run for every combination. So, totally $30 \times 6 = 180$ experiments are employed for each data set ($180 \times 3 = 540$ total experiments). For 10-job data set, the optimal solution is 27.7498. The results are analyzed in Table 32. The average results are very close to each other, so, it is not very clear which combination works better. On the other hand, when we check the frequency of the optimal solution, 0.1 mutation probability looks a lot better than others.

In Table 33 and Table 34, the results for 20-job and 40-job data sets are given. A mutation probability of 0.1 is still better, but the difference is very small. For 20-job data set, a mutation probability of 0.1 works better with reciprocal exchange mutation while for 40-job data set 0.1 mutation probability works better with insertion mutation.

As a result, we can conclude that when the problem is smaller, lower mutation probability can find the optimal solution easily. On the other hand, bigger mutation probability makes its search in a bigger range which can be concluded from the bigger
standard deviation. The best and worst solutions among the 180 replications (given next to tables) were found when mutation probability is 0.5 for both 20 and 40 jobs problems. This also supports the conclusion.

Table 32

Experiment Results for Cell Loading - 10 Jobs (30 Replicates)

<table>
<thead>
<tr>
<th>Mutation Prob.</th>
<th>Mutation Type</th>
<th>Frequency of Optimal Result</th>
<th>Average Result (Maximize Similarity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>Reciprocal Exchange m.</td>
<td>30</td>
<td>27.74980511</td>
</tr>
<tr>
<td>0.1</td>
<td>Insertion Mutation</td>
<td>30</td>
<td>27.74980511</td>
</tr>
<tr>
<td>0.3</td>
<td>Reciprocal Exchange m.</td>
<td>15</td>
<td>27.60496067</td>
</tr>
<tr>
<td>0.3</td>
<td>Insertion Mutation</td>
<td>17</td>
<td>27.64799871</td>
</tr>
<tr>
<td>0.5</td>
<td>Reciprocal Exchange m.</td>
<td>18</td>
<td>27.66073285</td>
</tr>
<tr>
<td>0.5</td>
<td>Insertion Mutation</td>
<td>16</td>
<td>27.63181473</td>
</tr>
</tbody>
</table>

Table 33

Experiment Results for Cell Loading - 20 Jobs (30 Replicates)

<table>
<thead>
<tr>
<th>Mutation Prob.</th>
<th>Mutation Type</th>
<th>Average Result (Maximize Similarity)</th>
<th>STDEV</th>
<th>MIN (Overall)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>Reciprocal Exchange m.</td>
<td>127.3405776</td>
<td>0.425260819</td>
<td>124.7584</td>
</tr>
<tr>
<td>0.1</td>
<td>Insertion Mutation</td>
<td>126.5377234</td>
<td>0.677363524</td>
<td>MAX (Overall)</td>
</tr>
<tr>
<td>0.3</td>
<td>Reciprocal Exchange m.</td>
<td>126.6704192</td>
<td>1.027418884</td>
<td>130.2075</td>
</tr>
<tr>
<td>0.3</td>
<td>Insertion Mutation</td>
<td>126.6108865</td>
<td>0.912711848</td>
<td>AVERAGE (Overall)</td>
</tr>
<tr>
<td>0.5</td>
<td>Reciprocal Exchange m.</td>
<td>126.5527835</td>
<td>0.847694639</td>
<td>127.0678</td>
</tr>
<tr>
<td>0.5</td>
<td>Insertion Mutation</td>
<td>126.7422246</td>
<td>1.050092798</td>
<td></td>
</tr>
</tbody>
</table>
Table 34

*Experiment Results for Cell Loading - 40 Jobs (30 Replicates)*

<table>
<thead>
<tr>
<th>Mutation Prob.</th>
<th>Mutation Type</th>
<th>Average Result (Maximize Similarity)</th>
<th>STDEV</th>
<th>MIN (Overall)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>Reciprocal Exchange m.</td>
<td>533.30392</td>
<td>2.040469842</td>
<td>527.715</td>
</tr>
<tr>
<td>0.1</td>
<td>Insertion Mutation</td>
<td>534.6609152</td>
<td>2.364799622</td>
<td>MAX (Overall)</td>
</tr>
<tr>
<td>0.3</td>
<td>Reciprocal Exchange m.</td>
<td>532.7855825</td>
<td>2.444164018</td>
<td>541.1272</td>
</tr>
<tr>
<td>0.3</td>
<td>Insertion Mutation</td>
<td>533.2352315</td>
<td>2.706271955</td>
<td>AVERAGE (Overall)</td>
</tr>
<tr>
<td>0.5</td>
<td>Reciprocal Exchange m.</td>
<td>533.2292595</td>
<td>2.407754275</td>
<td>533.3225</td>
</tr>
<tr>
<td>0.5</td>
<td>Insertion Mutation</td>
<td>532.7201397</td>
<td>2.840328031</td>
<td></td>
</tr>
</tbody>
</table>

Fischer test for 40-jobs has been analyzed to see if the difference is significant. LSD is calculated as 1.263484. The 2\textsuperscript{nd} option where mutation probability is 0.1 and mutation type is insertion mutation turned out to be better than others for 95% confidence level as shown in Table 35 (Page 102).
One-way ANOVA: y versus x

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>SS</th>
<th>MS</th>
<th>F</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>5</td>
<td>73.77</td>
<td>14.75</td>
<td>2.40</td>
<td>0.039</td>
</tr>
<tr>
<td>Error</td>
<td>174</td>
<td>1070.63</td>
<td>6.15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>179</td>
<td>1144.41</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S = 2.481   R-Sq = 6.45%   R-Sq(adj) = 3.76%

Individual 95% CIs For Mean Based on Pooled StDev

<table>
<thead>
<tr>
<th>Level</th>
<th>N</th>
<th>Mean</th>
<th>StDev</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30</td>
<td>533.30</td>
<td>2.04</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>534.66</td>
<td>2.36</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>532.79</td>
<td>2.44</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>533.24</td>
<td>2.71</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>533.23</td>
<td>2.41</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>532.72</td>
<td>2.84</td>
<td></td>
</tr>
</tbody>
</table>

Pooled StDev = 2.48

Figure 40: Boxplot for Combinations in Cell Loading.
Determining GA Parameters for Product Sequencing

Once the cell loading is accomplished, product sequencing should be performed for products in each cell. There are two methods proposed in order to accomplish this: 1-) using math model and 2-) using genetic algorithms. In this part, the experiments in order to determine better combinations of parameters are carried out. The experiments are done using GA program on 3 data sets given in APPENDIX D.

There are a total of 864 combinations (2*2*2*4*3*3*3) as calculated in Table 36. There are two mutation methods: reciprocal exchange mutation and insertion mutation; three mutation Probabilities: 0.1, 0.3 and 0.5; two crossover types: position-based crossover and order crossover; three crossover probabilities: 0.6 0.8 and 1; two selection
types: roulette wheel and best; four random population percentages: 0, 10, 20 and 30, and three position-based crossover probabilities: 0.1, 0.3 and 0.5. Other parameters are kept fixed which are manpower level = 20, number of chromosomes = 100 and number of generations = 400.

Table 36

Calculation of Combinations for Product Sequencing

<table>
<thead>
<tr>
<th>Crossover Type</th>
<th>Calculation of Combinations</th>
<th>Mutation Type</th>
<th>Calculation of Combinations</th>
<th>Selection Type</th>
<th>Calculation of Combinations</th>
<th>Manpower</th>
<th>Calculation of Combinations</th>
<th>Random Population Percentage</th>
<th>Calculation of Combinations</th>
<th>Mutation Probability</th>
<th>Calculation of Combinations</th>
<th>Crossover Probability</th>
<th>Calculation of Combinations</th>
<th>Position-based Crossover Probability</th>
<th>Calculation of Combinations</th>
<th>Population size</th>
<th>Calculation of Combinations</th>
<th>Number of Generations</th>
<th>Calculation of Combinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roulette Wheel</td>
<td>2</td>
<td>Roulette Wheel</td>
<td>2</td>
<td>Fixed at 20</td>
<td>1</td>
<td>20</td>
<td>30</td>
<td>4</td>
<td>3</td>
<td>0.5</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>0.5</td>
<td>3</td>
<td>Fixed at 100</td>
<td>1</td>
<td>Fixed at 400</td>
<td>1</td>
</tr>
<tr>
<td>Best</td>
<td>2</td>
<td>Reciprocal Exchange Mutation</td>
<td>2</td>
<td>Fixed at 20</td>
<td>1</td>
<td>20</td>
<td>30</td>
<td>4</td>
<td>3</td>
<td>0.5</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>0.5</td>
<td>3</td>
<td>Fixed at 100</td>
<td>1</td>
<td>Fixed at 400</td>
<td>1</td>
</tr>
<tr>
<td>Fixed at 100</td>
<td>1</td>
<td>Insertion Mutation</td>
<td>2</td>
<td>Fixed at 20</td>
<td>1</td>
<td>20</td>
<td>30</td>
<td>4</td>
<td>3</td>
<td>0.5</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>0.5</td>
<td>3</td>
<td>Fixed at 100</td>
<td>1</td>
<td>Fixed at 400</td>
<td>1</td>
</tr>
<tr>
<td>Fixed at 20</td>
<td>1</td>
<td>Fixed at 20</td>
<td>1</td>
<td>Fixed at 400</td>
<td>1</td>
<td>20</td>
<td>30</td>
<td>4</td>
<td>3</td>
<td>0.5</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>0.5</td>
<td>3</td>
<td>Fixed at 100</td>
<td>1</td>
<td>Fixed at 400</td>
<td>1</td>
</tr>
<tr>
<td>Fixed at 20</td>
<td>1</td>
<td>Fixed at 400</td>
<td>1</td>
<td>Fixed at 400</td>
<td>1</td>
<td>20</td>
<td>30</td>
<td>4</td>
<td>3</td>
<td>0.5</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>0.5</td>
<td>3</td>
<td>Fixed at 100</td>
<td>1</td>
<td>Fixed at 400</td>
<td>1</td>
</tr>
</tbody>
</table>

Ten replications are run for every combination. So, a total of 8640 (864*10) runs are employed for each data set (8640*3 = 25920 total runs). The objective is to minimize the number of periods where available manpower is exceeded.
For 10-job data set, the optimal result is 1. The GA program was able to find the optimal solution in 8348 runs out of 8640 runs. Moreover, GA program was able to find the optimal solution 10 times out of 10 times for 640 combinations out of 864 combinations. The number of optimal results is also given in Table 37 considering GA parameters used. For example, according to Table 37, 4175 optimal solutions are found when crossover type is position-based crossover and 4173 optimal solutions are obtained when crossover type is order crossover.

Table 37

<table>
<thead>
<tr>
<th>Number of Optimal Results for Product Sequencing - 10 Jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>Crossover Type</td>
</tr>
<tr>
<td>Mutation Type</td>
</tr>
<tr>
<td>Selection Type</td>
</tr>
<tr>
<td>Manpower</td>
</tr>
<tr>
<td>Random Population Percentage</td>
</tr>
<tr>
<td>Mutation Probability</td>
</tr>
<tr>
<td>Crossover Probability</td>
</tr>
<tr>
<td>Position-based Crossover Probability</td>
</tr>
<tr>
<td>Population size</td>
</tr>
<tr>
<td>Number of Generations</td>
</tr>
<tr>
<td>Total optimal solution</td>
</tr>
</tbody>
</table>

For 20-job data set, the optimal result is 1. The GA program was not able to find the optimal solution. However, the best solution found is 2. It found the best solution in 27 runs out of 8640 runs. Moreover, GA program was able to find the best solution in 2
times out of 10 times for only two combinations. So, the parameters for these two combinations can be regarded as the best parameters and they are given in Table 38.

Table 38

*Best Parameters for Product Sequencing - 20 Jobs*

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Position-based C.</td>
<td>Reciprocal Exchange M.</td>
<td>Best</td>
<td>0</td>
<td>0.1</td>
<td>0.6</td>
<td>0.1</td>
</tr>
<tr>
<td>Order C.</td>
<td>Insertion M.</td>
<td>Best</td>
<td>0</td>
<td>0.1</td>
<td>0.6</td>
<td>-</td>
</tr>
</tbody>
</table>

The number of best results is also given in Table 39 considering GA parameters used. For example, according to Table 39, 14 best solutions are found when crossover type is position-based crossover and 13 optimal solutions are obtained when crossover type is order crossover.
Table 39

**Number of Optimal Results for Product Sequencing - 20 Jobs**

<table>
<thead>
<tr>
<th></th>
<th>Number of Best Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Crossover Type</strong></td>
<td>14 (PBC) 13 (OC)</td>
</tr>
<tr>
<td><strong>Mutation Type</strong></td>
<td>10 (REM) 17 (IM)</td>
</tr>
<tr>
<td><strong>Selection Type</strong></td>
<td>4 (RW) 23 (B)</td>
</tr>
<tr>
<td><strong>Manpower</strong></td>
<td>Fixed at 20</td>
</tr>
<tr>
<td><strong>Random Population Percentage</strong></td>
<td>7 (0) 6 (10) 8 (20) 6 (30)</td>
</tr>
<tr>
<td><strong>Mutation Probability</strong></td>
<td>16 (0.1) 11 (0.3) 0 (0.5)</td>
</tr>
<tr>
<td><strong>Crossover Probability</strong></td>
<td>10 (0.6) 10 (0.8) 7 (1)</td>
</tr>
<tr>
<td><strong>Position-based Crossover Probability</strong></td>
<td>7 (0.1) 3 (0.3) 4 (0.5)</td>
</tr>
<tr>
<td><strong>Population size</strong></td>
<td>Fixed at 100</td>
</tr>
<tr>
<td><strong>Number of Generations</strong></td>
<td>Fixed at 400</td>
</tr>
<tr>
<td><strong>Total best solution</strong></td>
<td>27</td>
</tr>
</tbody>
</table>

For 40-job data set, the optimal result equals to 6. The GA program was not able to find the optimal solution since the problem size is too big. The best solution found is 7. The GA program found the best solution in 5 runs out of 8640 runs. They all belong to different combinations. So, the parameters for these five combinations can be regarded as the best parameters and they are given in Table 40.

If better choices are intersected for each parameter, then 2\textsuperscript{nd}, 3\textsuperscript{rd} and 4\textsuperscript{th} combinations look better.
Table 40

*Best Parameters for Product Sequencing - 40 Jobs*

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Order C.</td>
<td>Reciprocal Exchange M.</td>
<td>Roulette Wheel</td>
<td>0</td>
<td>0.5</td>
<td>0.6</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>Order C.</td>
<td>Insertion M.</td>
<td>Best</td>
<td>10</td>
<td>0.1</td>
<td>0.6</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>Order C.</td>
<td>Reciprocal Exchange M.</td>
<td>Best</td>
<td>10</td>
<td>0.1</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>Order C.</td>
<td>Reciprocal Exchange M.</td>
<td>Roulette Wheel</td>
<td>30</td>
<td>0.3</td>
<td>0.6</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>Position Based C.</td>
<td>Insertion M.</td>
<td>Best</td>
<td>30</td>
<td>0.3</td>
<td>1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

The number of best results is also given in Table 41 considering GA parameters used. For example, according to Table 41, only one best solution is found when crossover type is position-based crossover and 4 best solutions are obtained when crossover type is order crossover.
Table 41

*Number of Optimal Results for Product Sequencing - 40 Jobs*

<table>
<thead>
<tr>
<th></th>
<th>Number of Best Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Crossover Type</strong></td>
<td>1 (PBC)</td>
</tr>
<tr>
<td><strong>Mutation Type</strong></td>
<td>3 (REM)</td>
</tr>
<tr>
<td><strong>Selection Type</strong></td>
<td>2 (RW)</td>
</tr>
<tr>
<td><strong>Manpower</strong></td>
<td>Fixed at 20</td>
</tr>
<tr>
<td><strong>Random Population Percentage</strong></td>
<td>1 (0)</td>
</tr>
<tr>
<td><strong>Mutation Probability</strong></td>
<td>2 (0.1)</td>
</tr>
<tr>
<td><strong>Crossover Probability</strong></td>
<td>3 (0.6)</td>
</tr>
<tr>
<td><strong>Position-based Crossover Probability</strong></td>
<td>1 (0.1)</td>
</tr>
<tr>
<td><strong>Population size</strong></td>
<td>Fixed at 100</td>
</tr>
<tr>
<td><strong>Number of Generations</strong></td>
<td>Fixed at 400</td>
</tr>
<tr>
<td><strong>Total best solution</strong></td>
<td>5</td>
</tr>
</tbody>
</table>

As a conclusion, when we look at the entire results of experiments, we see that Best Selection type works better than Roulette Wheel, insertion mutation works slightly better than reciprocal exchange mutation, 0.1 mutation probability is better than others, and lastly, 0.1 position-based probability is better than others.

The statistical analysis is applied only to 40-job experiment. First, only the combinations given in Table 41 are considered. However, there is not a significant difference between them. Therefore, other than looking at best solutions, Fischer test is applied considering averages for the entire set of combinations to see if the difference are significant (for 40-job data set). Therefore, the best combination considering the average is chosen and it is compared against combinations with lower average to see how many combinations are there with no significance between. The combination that has the best
average is given below. The 10 replications of this combination are compared against others starting from the one with better average to the worst ones. It is concluded that there is no significant difference between the first 51 combinations and the best combination. So, all of top 51 combinations can be considered as good combinations.

Table 42

The Combination with the Best Average Result (40 Jobs)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Position-based C.</td>
<td>Reciprocal Exchange M.</td>
<td>Best</td>
<td>10</td>
<td>0.1</td>
<td>0.8</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Determining GA parameters for Cell Loading and Product Sequencing

GA program used for this purpose accomplishes both cell loading and product sequencing at the same time. It was not possible to create a mathematical model in order to do both cell loading and product sequencing. So, there is only GA program for performing cell loading and product sequencing simultaneously. In this part, the experiments in order to determine better parameters for genetic approach are analyzed. The experiments are accomplished using the GA program on 3 data sets given at APPENDIX D.
There are a total of 864 combinations (2*2*2*4*3*3*3). It is the same number of combinations with product sequencing problem which was already shown in Table 36 (page 103).

Ten replications are run for every combination. So, totally 864*10=8640 runs are employed for each data set (8640*3 = 25920 total runs). The objective is to minimize the total number of periods that available manpower is exceeded in two cells.

Since there is not a mathematical model developed to do both cell loading and product sequencing at the same time, the optimal result is not known. For 10-job data set, the best result found by GA is 2. The GA program was able to find that solution in 8639 runs out of 8640 runs. So, GA could not find that result in just 1 run out of 8640 runs. So, 2 is probably the optimal result. The only different solution has the parameters given in Table 43. In that table, we see that mutation probability and crossover probability have their highest values at the same time; also random population percentage is pretty high, so this leads to bigger search space, but worse results. In Table 43, position-based crossover probability is 0.3 which also did not work well in previous experiments.
Table 43

Combination of Parameters for Worst Cell Loading and Product Sequencing Result - 10 Jobs

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Position-based C.</td>
<td>Insertion Mutation</td>
<td>Best</td>
<td>20</td>
<td>0.5</td>
<td>1</td>
<td>0.3</td>
</tr>
</tbody>
</table>

For 20-job data set, the best result found by GA equals to 2. The GA program was able to find the best solution in 1 run out of 8640 runs. So, the parameters for this combination can be regarded as better parameters and they are given in Table 44. It is not surprising to see that this combination leads to the best solution. Because, these parameters worked better in the previous experiments too.

Table 44

Best Combination of Parameters for Cell Loading and Product Sequencing- 20 Jobs

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Order C.</td>
<td>Reciprocal Exchange M.</td>
<td>Best</td>
<td>20</td>
<td>0.1</td>
<td>0.6</td>
<td>-</td>
</tr>
</tbody>
</table>

For 40-job data set, the best result obtained is 8. The GA program found this solution in 13 runs out of 8640 runs. They all belong to different combinations. So, the
parameters for these 13 combinations can be regarded as the best parameters and they are listed in Table 45. According to that table, reciprocal exchange mutation works a lot better than insertion mutation. The preferred selection type is best, mutation probability is 0.1, and position-based crossover probability is 0.1.

If better choices are intersected all together, 4th and 11th choices have them all. So, they may be regarded as the better combinations of parameters. Also, Fischer test is applied to these 13 best combinations to see if there is a significant difference between them (for 40-job data set).

**One-way ANOVA: y versus x**

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>SS</th>
<th>MS</th>
<th>F</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>12</td>
<td>10.569</td>
<td>0.881</td>
<td>0.98</td>
<td>0.474</td>
</tr>
<tr>
<td>Error</td>
<td>117</td>
<td>105.400</td>
<td>0.901</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>129</td>
<td>115.969</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S = 0.9491  R-Sq = 9.11%  R-Sq(adj) = 0.00%

Individual 95% CIs For Mean Based on Pooled StDev

| Level | N   | Mean  | StDev | +---------+---------+---------+---------|
|-------|-----|-------|-------|---------+---------+---------+---------|
| 1     | 10  | 10.000| 0.816 | (---------*---------) |
| 2     | 10  | 10.200| 1.033 | (---------*---------) |
| 3     | 10  | 10.500| 0.972 | (---------*---------) |
| 4     | 10  | 9.900 | 0.876 | (---------*---------) |
| 5     | 10  | 10.300| 0.949 | (---------*---------) |
| 6     | 10  | 9.600 | 0.843 | (---------*---------) |
| 7     | 10  | 10.200| 0.919 | (---------*---------) |
| 8     | 10  | 10.200| 1.033 | (---------*---------) |
| 9     | 10  | 10.500| 0.972 | (---------*---------) |
| 10    | 10  | 10.600| 1.174 | (---------*---------) |
| 11    | 10  | 9.900 | 0.738 | (---------*---------) |
| 12    | 10  | 10.500| 0.972 | (---------*---------) |
| 13    | 10  | 10.400| 0.966 | (---------*---------) |

Pooled StDev = 0.949
LSD is calculated as 0.839822. There is a significant difference between 3 and 6, 6 and 9, 6 and 10, 6 and 12. The difference is not significant for others. So, we can eliminate 3rd, 9th, 10th, and 12th options, so there are 9 options remaining that we can consider as good combinations.
## Table 45

**Best Combination of Parameters for Cell Loading and Product Sequencing- 40 Jobs**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Order C.</td>
<td>Reciprocal Exchange M.</td>
<td>Best</td>
<td>0</td>
<td>0.1</td>
<td>0.8</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>Position Based C.</td>
<td>Reciprocal Exchange M.</td>
<td>Best</td>
<td>0</td>
<td>0.3</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>3</td>
<td>Position Based C.</td>
<td>Reciprocal Exchange M.</td>
<td>Roulette Wheel</td>
<td>0</td>
<td>0.5</td>
<td>0.6</td>
<td>0.3</td>
</tr>
<tr>
<td>4</td>
<td>Position Based C.</td>
<td>Reciprocal Exchange M.</td>
<td>Best</td>
<td>10</td>
<td>0.1</td>
<td>0.6</td>
<td>0.1</td>
</tr>
<tr>
<td>5</td>
<td>Order C.</td>
<td>Reciprocal Exchange M.</td>
<td>Roulette Wheel</td>
<td>10</td>
<td>0.1</td>
<td>0.6</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>Order C.</td>
<td>Reciprocal Exchange M.</td>
<td>Best</td>
<td>10</td>
<td>0.1</td>
<td>0.6</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>Position Based C.</td>
<td>Reciprocal Exchange M.</td>
<td>Best</td>
<td>10</td>
<td>0.1</td>
<td>0.8</td>
<td>0.3</td>
</tr>
<tr>
<td>8</td>
<td>Order C.</td>
<td>Reciprocal Exchange M.</td>
<td>Roulette Wheel</td>
<td>20</td>
<td>0.1</td>
<td>0.8</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>Position Based C.</td>
<td>Reciprocal Exchange M.</td>
<td>Best</td>
<td>20</td>
<td>0.5</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>10</td>
<td>Order C.</td>
<td>Reciprocal Exchange M.</td>
<td>Roulette Wheel</td>
<td>30</td>
<td>0.1</td>
<td>0.6</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>Position Based C.</td>
<td>Reciprocal Exchange M.</td>
<td>Best</td>
<td>30</td>
<td>0.1</td>
<td>0.8</td>
<td>0.1</td>
</tr>
<tr>
<td>12</td>
<td>Order C.</td>
<td>Reciprocal Exchange M.</td>
<td>Best</td>
<td>30</td>
<td>0.3</td>
<td>0.8</td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>Position Based C.</td>
<td>Insertion M.</td>
<td>Best</td>
<td>30</td>
<td>0.3</td>
<td>1</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Population Size vs. Number of Generations

It is critical to know how big we should keep population size compared to the number of generations. In order to find an answer to this question, 5 different combinations are tested. Ten replications are run for every combination. So, totally 50 runs are made. In the first combination, the population size is kept 100 while the number of generations is 2500. In the second combination, the population size is 250 while the number of generations is 1000 and so on as given in Table 46. The important thing is to keep (population size*number of generations) fixed at 250000.

For this experiment, cell loading and product sequencing program is used with 40-job data set. The results given in Table 47 are obtained. It is interesting to see that when the number of generations and population size are close to each other, the results are better. Also, keeping population size higher than the number of generations is not a better idea than keeping number of generations higher than population size.

Table 46

*Population size vs. Number of Generations*

<table>
<thead>
<tr>
<th>Population Size</th>
<th>Number of Generations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st option</td>
<td>100</td>
</tr>
<tr>
<td>2nd option</td>
<td>250</td>
</tr>
<tr>
<td>3rd option</td>
<td>500</td>
</tr>
<tr>
<td>4th option</td>
<td>1000</td>
</tr>
<tr>
<td>5th option</td>
<td>2500</td>
</tr>
</tbody>
</table>
Table 47

*Results for Population size vs. Number of Generations*

<table>
<thead>
<tr>
<th></th>
<th>Average Result</th>
<th>Best Result</th>
<th>Frequency of Best (5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st option</td>
<td>8</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>2nd option</td>
<td>6.4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>3rd option</td>
<td>6.1</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>4th option</td>
<td>7.3</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>5th option</td>
<td>8.7</td>
<td>8</td>
<td>0</td>
</tr>
</tbody>
</table>

Fischer test is also applied to see if the results are significantly different. LSD equals to 0.727531 for 95% confidence level. The 2nd and 3rd options are significantly better than others as seen in Table 48. However, they are not significantly different than each other.
One-way ANOVA: y versus x

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>SS</th>
<th>MS</th>
<th>F</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>4</td>
<td>47.000</td>
<td>11.750</td>
<td>17.92</td>
<td>0.000</td>
</tr>
<tr>
<td>Error</td>
<td>45</td>
<td>29.500</td>
<td>0.656</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>49</td>
<td>76.500</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S = 0.8097  R-Sq = 61.44%  R-Sq(adj) = 58.01%

Individual 95% CIs For Mean Based on Pooled StDev

| Level | N    | Mean | StDev | ----+---------+---------+---------+-----|
|-------|------|------|-------|-----+---------+---------+---------+-----|
| 1     | 10   | 8.0000| 0.8165| (-----*----)
| 2     | 10   | 6.4000| 0.8433| (-----*----)
| 3     | 10   | 6.1000| 0.9944| (-----*----)
| 4     | 10   | 7.3000| 0.8233| (-----*----)
| 5     | 10   | 8.7000| 0.4830|                            (-----*----)

Pooled StDev = 0.8097

Figure 41: Boxplot for Number of Generations vs. Population Size.
Table 48

*Fischer Test for Population Size vs. Number of Generations*

<table>
<thead>
<tr>
<th>Level 1</th>
<th>Level 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>not significant</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>significant</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>significant</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>significant</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>significant</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>significant</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>significant</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>not significant</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>significant</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>not significant</td>
</tr>
</tbody>
</table>

**Effect of Phases**

The total number of generations is divided into 5 phases as shown in Figure 42. It is possible to run a lot of experiments on these phases. However, in this paper, it is only compared with no-phase case in order to show how effective it is.

*Figure 42: Effect of Phases.*
In the no-phase case, every chromosome goes thorough crossover and mutation in each generation. However, in the 5-phase case, chromosomes go through just mutation, just crossover, or both. This is the primary difference. For example, in Figure 42, there are 400 generations which are divided into 5 phases. Every phase has 20% of 400 generations which equals to 80. In the first phase, chromosomes just go through crossover for 80 generations. Then, in the second phase, half of the population goes through crossover and mutation while other half goes through just crossover. This procedure continues like this as shown in Figure 42 until the program reaches 400 generations.

In order to compare 5-phase case and no-phase case, the divisions of the phases given in Figure 42 are applied on 40-job data set using cell loading and product sequencing program. All parameters are kept fixed. Just no-phase case is clicked or unclicked. Ten replications are run and the results are given in Table 49. It is easy to see that there is a significant difference between no-phase case and 5-phase case. The worst result obtained using 5-phase case is even better than the best result obtained using no-phase case. So, 5-phase case is a lot better.

Table 49

Effect of Phases

<table>
<thead>
<tr>
<th></th>
<th>Average Result (# of periods that available manpower is exceeded)</th>
<th>Best Result (# of periods that available manpower is exceeded)</th>
<th>Worst Result (# of periods that available manpower is exceeded)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No-Phase</td>
<td>9.5</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>5-Phase</td>
<td>6.1</td>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>
It is also possible to show that this difference is significant using Fisher Test. LSD was calculated as 0.598057 for 95% confidence level. The difference between means is 3.4, so, the difference between two approaches is significantly different.

One-way ANOVA: y versus x

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>SS</th>
<th>MS</th>
<th>F</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
<td>57.800</td>
<td>57.800</td>
<td>140.59</td>
<td>0.000</td>
</tr>
<tr>
<td>Error</td>
<td>18</td>
<td>7.400</td>
<td>0.411</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>19</td>
<td>65.200</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S = 0.6412  R-Sq = 88.65%  R-Sq(adj) = 88.02%

Individual 95% CIs For Mean Based on Pooled StDev

| Level | N  | Mean | StDev | ---+---------+---------+---------+------|
|-------|----|------|-------|---+---------+---------+---------+------|
| 1     | 10 | 6.1000 | 0.7379 | ---+---------+---------+---------+------| (--*--) |
| 2     | 10 | 9.5000 | 0.5270 | ---+---------+---------+---------+------| (---*--) |

---+---------+---------+---------+------|
6.0       7.2       8.4       9.6       

Pooled StDev = 0.6412
Comparison of Math Model and Genetic Algorithms for Cell Loading

Three data sets given in APPENDIX D are applied to the mathematical model and the results are compared with the results of GA. Below, codes used for mathematical model are given for 10-job problem. The mathematical model and genetic algorithm programs have different objectives. So, “execute” command is added at the end of the code in order to convert the objective of mathematical model into the objective of genetic algorithms so that both can be compared.

Figure 43: Boxplot for 5-Phase vs. No-Phase.
//CELL LOADING FOR 10 JOBS

int NbProducts = 10;
range Products = 1..NbProducts;
dvar int+ x[Products][Products] in 0..1;
int convertedObjective;
int cellArray[Products];

//similarity coefficients
float SimilarityCoefficients[Products][Products] =
    [ [ 0 , 0.666666667 , 0.6 , 0.666666667 ,
      0.739130435 , 0.666666667 , 0.6 , 0.739130435 ,
      0.739130435 , 0.538461538 ] ,
      [ 0.666666667 , 0 , 0.6 , 0.6 , 0.6 , 0.739130435 , 0.818181818 , 0.904761905 , 0.818181818 , 0.739130435 ] ,
      [ 0.6 , 0.6 , 0 , 0.6 , 0.666666667 , 0.538461538 , 0.6 , 0.666666667 , 0.538461538 ] ,
      [ 0.666666667 , 0.6 , 0.6 , 0 ] ,
      [ 0.481481481 , 0.739130435 , 0.739130435 , 0.666666667 , 0.6 , 0.481481481 ] ,
      [ 0.739130435 , 0.6 , 0.666666667 , 0.481481481 ] ,
      [ 0.666666667 , 0.666666667 , 0.538461538 ] ,
      [ 0.666666667 , 0.739130435 , 0.538461538 , 0.538461538 ] ,
      [ 0.739130435 , 0.6 , 0 , 0.739130435 ] ,
      [ 0.739130435 , 0.818181818 , 0.538461538 ] ,
      [ 0.6 , 0.818181818 , 0.6 , 0.739130435 , 0.739130435 , 0.666666667 ] ,
      [ 0.481481481 , 0.739130435 , 0 , 0.739130435 ] ,
      [ 0.739130435 , 0.904761905 , 0.666666667 , 0.666666667 , 0.666666667 , 0.739130435 , 0.739130435 , 0 , 0.666666667 , 0.666666667 ] ,
      [ 0.739130435 , 0.818181818 , 0.538461538 , 0.6 , 0.666666667 , 0.666666667 , 0.666666667 , 0.666666667 , 0.666666667 ] ,
      [ 0.481481481 , 0.538461538 , 0.538461538 , 0.538461538 , 0.666666667 , 0.666666667 , 0 ] ]

maximize
sum (k in Products)
sum (p in Products)
x[p][k]*SimilarityCoefficients[p][k];

subject to
// all products should be assigned to a cell
forall ( p in Products )
sum ( k in Products )
x[p][k]==1;

// upper limit for the number of products assigned to a cell is 5
forall ( c in Products )
sum( p in Products )
x[p][c]<=5;

forall ( p in Products )
forall ( c in Products )
x[c][p]<=x[p][p];

// total number of cells equals to 2
sum (k in Products)
x[k][k]==2;

}

execute
{
    for (var k in Products)
    for (var s in Products)
    {
        if (x[k][s] == 1)
        {
            cellArray[k] = s;
        }
    }

    for (var l in Products)
    for (var m in Products)
    {
        if (cellArray[l] ==cellArray[m])
        {
            convertedObjective = convertedObjective +
            Similaritycoefficients[l][m];
        }
    }

    writeln("convertedObjective = ", convertedObjective);
}

When this code is entered to the OPL, the results obtained are given below.

According to these results, we see that there are 2 cells opened (cell 1 and cell 2).
Products 1, 3, 4, 5, 6 are assigned to the first cell and products 2, 7, 8, 9, 10 are assigned to the second cell. Then, the objective (total similarity) is maximized with a value of 5.952719745. Execute command finds the objective of genetic algorithms when products 1, 3, 4, 5, 6 are assigned to the first cell and products 2, 7, 8, 9, 10 are assigned to the second cell. So, the converted objective is 27.7498 (which add all pairwise similarities for the products in the same family). The conversion is discussed in APPENDIX G.

```cpp
// solution (optimal) with objective 5.952719745
x = [[1 0 0 0 0 0 0 0 0 0]
     [0 1 0 0 0 0 0 0 0 0]
     [1 0 0 0 0 0 0 0 0 0]
     [1 0 0 0 0 0 0 0 0 0]
     [1 0 0 0 0 0 0 0 0 0]
     [1 0 0 0 0 0 0 0 0 0]
     [0 1 0 0 0 0 0 0 0 0]
     [0 1 0 0 0 0 0 0 0 0]
     [0 1 0 0 0 0 0 0 0 0]
     [0 1 0 0 0 0 0 0 0 0]];  
```

```cpp
// solution (optimal) with objective 5.952719745
convertedObjective = 27.74980511
```

The codes for 20-job and 40-job data sets and results are given in APPENDIX E.
The GA was able to find the optimal result for 10 jobs. However, it did not give the optimal result for 20 and 40-job data sets. On the other hand, Mathematical Model finds the optimal result. The comparison is given in Table 50 considering both duration of execution and the results. It is possible to see that modified version of P-median model works really efficiently. It can still find the optimal solution in less than one second for 40-job problem. On the other side, GA is pretty fast too, however, it cannot catch mathematical model in duration or in accuracy of the results.

Table 50

<table>
<thead>
<tr>
<th>GA vs. Math Modeling for Cell Loading</th>
<th>10 JOBS</th>
<th>20 JOBS</th>
<th>40 JOBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Best Result &amp; Average Result</td>
<td>27.7498 &amp; 27.6756</td>
<td>130.2075 &amp; 127.0678</td>
<td>541.1272 &amp; 533.3225</td>
</tr>
<tr>
<td>Average Duration (second)</td>
<td>0.2576 sec.</td>
<td>0.936 sec.</td>
<td>2.48 sec.</td>
</tr>
<tr>
<td>MATHEMATICAL MODEL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Result</td>
<td>27.7498</td>
<td>130.9992</td>
<td>563.9968</td>
</tr>
<tr>
<td>Duration (second)</td>
<td>0.01 sec.</td>
<td>0.01 sec.</td>
<td>0.06 sec.</td>
</tr>
</tbody>
</table>

Comparison of Math Model and Genetic Algorithms for Product Sequencing

Three data sets given in APPENDIX D are applied to the mathematical model for product sequencing and the results are compared with the results obtained in GA. The codes entered in the mathematical model for 10-job problem is given below:
//10 JOBS PRODUCT SEQUENCING

int NbPeriods = 10;
int NbProducts = 10;
int NbTotalperiods = 14;
int NbStages = 5;
int NbVariable = 6;

range Periods = 1..NbPeriods;
range Products = 1..NbProducts;
range Totalperiods = 1..NbTotalperiods;
range Stages = 1..NbStages;
range Variable = 1..NbVariable;

//manpower levels for jobs in all stages
int m[Stages][Products] =
[ 3, 5, 3, 2, 3, 2, 5, 4, 4, 8 ],
[ 5, 3, 8, 4, 6, 3, 3, 4, 3, 3 ],
[ 4, 4, 2, 2, 6, 5, 2, 4, 5, 3 ],
[ 6, 4, 3, 6, 3, 5, 5, 4, 3, 3 ],
[ 2, 4, 4, 6, 2, 5, 5, 4, 3, 3 ];

//sum of extra manpower levels
float E[Totalperiods] =
[15, 11, 7, 4, 0, 0, 0, 0, 0, 0, 3.9, 8.08181818181818, 11.9984848484848, 16.3061771561772];

dvar int+ x[Products][Periods] in 0..maxint;
dvar float+ t[Totalperiods] in 0..maxint;
dvar int+ s[Totalperiods];

minimize
sum (j in Totalperiods)
s[j];

subject to
{
forall ( j in Products )
sum ( p in Periods )
x[ j ][ p ] == 1;
forall( p in Periods)
sum ( j in Products)
x[ j ][ p ] == 1;

sum ( p in Periods )
m[ 1 ][ p ] * x[ p ][ 1 ] + E[1] == t[1];

sum ( p in Periods )
m[ 2 ][ p ] * x[ p ][ 1 ]
+ sum (p in Periods)

sum ( p in Periods )
m[ 3 ][ p ] * x[ p ][ 1 ]
+ sum (p in Periods)
m[ 2 ][ p ] * x[ p ][ 2 ]
+ sum (p in Periods)

sum ( p in Periods )
m[ 4 ][ p ] * x[ p ][ 1 ]
+ sum (p in Periods)
m[ 3 ][ p ] * x[ p ][ 2 ]
+ sum (p in Periods)
m[ 2 ][ p ] * x[ p ][ 3 ]
+ sum (p in Periods)

forall(k in Variable)
sum ( p in Periods )
m[ 5 ][ p ] * x[ p ][ k ]
+ sum ( p in Periods )
m[ 4 ][ p ] * x[ p ][ k+1 ]
+ sum ( p in Periods )
m[ 3 ][ p ] * x[ p ][ k+2 ]
+ sum (p in Periods)
m[ 2 ][ p ] * x[ p ][ k+3 ]
+ sum (p in Periods)
m[ 1 ][ p ] * x[ p ][ k+4 ] + E[k+4] == t[ k+4 ];

sum ( p in Periods )
m[ 5 ][ p ] * x[ p ][ 7 ]
+
\[
\begin{align*}
\text{sum ( p in Periods )} & \quad \text{m}[4][ p ] \times x[ p ][8] + \\
\text{sum ( p in Periods )} & \quad \text{m}[3][ p ] \times x[ p ][9] + \\
\text{sum ( p in Periods )} & \quad \text{m}[2][ p ] \times x[ p ][10] + \text{E}[11] == \text{t}[11]; \\
\text{sum ( p in Periods )} & \quad \text{m}[5][ p ] \times x[ p ][8] + \\
\text{sum ( p in Periods )} & \quad \text{m}[4][ p ] \times x[ p ][9] + \\
\text{sum ( p in Periods )} & \quad \text{m}[3][ p ] \times x[ p ][10] + \text{E}[12] == \text{t}[12]; \\
\text{sum ( p in Periods )} & \quad \text{m}[5][ p ] \times x[ p ][9] + \\
\text{sum ( p in Periods )} & \quad \text{m}[4][ p ] \times x[ p ][10] + \text{E}[13] == \text{t}[13]; \\
\text{sum ( p in Periods )} & \quad \text{m}[5][ p ] \times x[ p ][10] + \text{E}[14] == \text{t}[14]; \\
\forall (j \text{ in Totalperiods} ) & \quad 1000 \times s[j] \geq t[j]-20;
\end{align*}
\]

According to these results, the optimal value is 1. The sequence should be 8 – 2 – 6 – 7 – 4 – 1 – 3 – 10 – 5 – 9. Then the total manpower level is 19 for the first period, 20
for the second period, 16 for the third period and so on as given in t matrix. The codes for 20-job, 40-job and 80-job results are given in APPENDIX F.

For the GA part, better parameters are used while keeping population size and the number of generations at 500. 10 replications are employed for each data set. So, a total of 40 experiments are run. The GA was able to find the optimal result for 10-job problem. However, it did not give the optimal result for 20-job data set. While, Mathematical Model finds the optimal results for 10-job and 20-job problems, the computer was out of memory after 2 hours for 40-job data set. GA found the same result for this data set (40 Jobs). However, when we increase the problem size to 80 jobs, GA could find better results compared to the Mathematical Model. The comparison is given in Table 51 considering duration of execution and the results.

Table 51

*GA vs. Math Modeling for Product Sequencing*

<table>
<thead>
<tr>
<th></th>
<th>10 JOBS</th>
<th>20 JOBS</th>
<th>40 JOBS</th>
<th>80 JOBS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GA</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Best Result &amp; Average Result</td>
<td>1 &amp; 1</td>
<td>2 &amp; 2.3</td>
<td>4 &amp; 5.1</td>
<td>10 &amp; 12.2</td>
</tr>
<tr>
<td>Average Duration (second)</td>
<td>0.686 sec.</td>
<td>9.1875 sec.</td>
<td>18.5625 sec.</td>
<td>37.33 sec.</td>
</tr>
<tr>
<td><strong>MATHEMATICAL MODEL</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duration (second)</td>
<td>0.40 sec.</td>
<td>1212.27 sec.</td>
<td>Out of memory after 1.6 hours</td>
<td>Out of Memory after 4.7 hours</td>
</tr>
</tbody>
</table>
Best Method for Cell Loading + Product Sequencing

There are two methods for cell loading and two methods for product sequencing and one method for both cell loading and product sequencing. It should be determined which one will work better among all of them. These methods are summarized in Table 52.

Table 52

| Possible Options in Order to Accomplish Cell Loading and Product Sequencing |
|---------------------------------|-----------------|
| **Cell Loading**                | **Product Sequencing** |
| 1st option                      | GA              | GA              |
| 2nd option                      | GA              | Math model      |
| 3rd option                      | Math model      | Math model      |
| 4th option                      | Math model      | GA              |
| 5th option                      | GA              |

Additionally, for the experimentation purpose, two more objectives are added into the GA program for cell loading. The first objective was to maximize similarity. Moreover, minimizing similarity and random cell loading are also added. So, the number of combinations increased to 9 as given in Table 53.
Table 53

**Possible Options with Additional Options in Order to Accomplish Cell Loading and Product Sequencing**

<table>
<thead>
<tr>
<th>Option</th>
<th>Cell Loading</th>
<th>Product Sequencing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st option</td>
<td>GA - 1st objective</td>
<td>GA</td>
</tr>
<tr>
<td>2nd option</td>
<td>GA - 1st objective</td>
<td>Math model</td>
</tr>
<tr>
<td>3rd option</td>
<td>GA - 2nd objective</td>
<td>GA</td>
</tr>
<tr>
<td>4th option</td>
<td>GA - 2nd objective</td>
<td>Math model</td>
</tr>
<tr>
<td>5th option</td>
<td>GA - 3rd objective</td>
<td>GA</td>
</tr>
<tr>
<td>6th option</td>
<td>GA - 3rd objective</td>
<td>Math model</td>
</tr>
<tr>
<td>7th option</td>
<td>Math model</td>
<td>Math model</td>
</tr>
<tr>
<td>8th option</td>
<td>Math model</td>
<td>GA</td>
</tr>
<tr>
<td>9th option</td>
<td></td>
<td>GA</td>
</tr>
</tbody>
</table>

To accomplish the experiment, 40-job data set is used. For GA, the population size is kept at 500 with 500 generations. The manpower level is 20 and 10 replications are run. Better parameters which were determined in previous sections are used. These parameters are given in Table 54, Table 55 and Table 56.

Since 40-job data set is used, 40 jobs are considered for cell loading and 20 jobs are considered in product sequencing per cell (number of cells=2).
The results obtained from 9 options are given in Table 57. First of all, for this 40-job cell loading problem, splitting the entire procedure into two phases is a good idea. Because, the worst results are obtained from the 9th option in which cell loading and product sequencing were accomplished in one GA program. The best result is 2 and this result is obtained from GA-2nd Objective (minimizing total similarity) in cell loading and
math modeling in product sequencing. It was seen in previous experiments that math modeling in product sequencing works better in 20-job problem since the problem size is still small enough for math model. So, it is not surprising to get better results from the methods which use math modeling in product sequencing part of the problem (Check 2nd, 4th, 6th, 7th options in the table).

However, it is interesting that maximizing similarity in cell loading does not improve results in product sequencing. Just compare 4th and 7th options where they both use math modeling in product sequencing however, 4th option minimizes similarity in the cell loading part using GA, on the other hand, 7th option uses math model. It is so obvious that minimizing total similarity works a lot better in this case. It is even possible to say that a random cell loading approach works better than maximizing similarity.

Table 57

*Overall Results for 9 options for Cell Loading and Product Sequencing (40 Jobs)*

<table>
<thead>
<tr>
<th>Option</th>
<th>Best Result</th>
<th>Average Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st option</td>
<td>5</td>
<td>5.7</td>
</tr>
<tr>
<td>2nd option</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3rd option</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4th option</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>5th option</td>
<td>4</td>
<td>5.1</td>
</tr>
<tr>
<td>6th option</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>7th option</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>8th option</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>9th option</td>
<td>5</td>
<td>6.2</td>
</tr>
</tbody>
</table>
CHAPTER 6: CONCLUSIONS AND FUTURE WORK

Genetic Algorithm can be considered as organization of randomization as inspired by natural evolutionary process. It has been shown that traditional GA gives better results than random solutions. Later, better parameters for GA were determined (They are summarized in Table 59). Then, the best population size and number of generation combination was determined. It has been observed that when population size and number of generations are closer, it gives better results. Later, a newly generated GA approach (5-Phase Approach) was compared against traditional GA Approach and it was shown that it is a lot more efficient than traditional GA. At the end, GA was ready to be compared with mathematical modeling. Mathematical modeling and GA is compared for both cell loading and also for product sequencing problems.

In short, there are two methods used. The first one is mathematical modeling and the second one is genetic algorithm. It has been observed that both of them may be powerful problem solving tools. For example modified version of p-median model gives the optimal result in a very short time and it does not depend on the problem size much. So, GA was not even a match against math modeling for cell loading problem. The advantage of mathematical modeling is that it always gives optimal solution unless computer runs out of memory.

However, it has also been observed that sometimes math modeling spends a lot of time trying different solutions when the problem size is big and cannot give a good solution and computer runs out of memory after a while. This case was observed in product sequencing problem. GA gave better results in a very short time.
About Genetic Algorithms, setting the experimental conditions is really important. The effect of parameters on the result should be observed during experimentation and better parameters must be used in order to improve efficiency of the genetic algorithm approach. One type of mutation may work well on one problem while another mutation type works well on another problem. The better working parameters are given in Table 59. Some shortcuts have been used which are given in Table 58. Mutation probability of 0.1 and best selection type works better for all cases.

Table 58

<table>
<thead>
<tr>
<th>Shortcuts</th>
<th>Crossover Types</th>
<th>Mutation Types</th>
<th>Selection Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order Crossover</td>
<td>OC</td>
<td>Reciprocal Exchange Mutation</td>
<td>REM</td>
</tr>
<tr>
<td>Position Based Crossover</td>
<td>PBC</td>
<td>Insertion Mutation</td>
<td>IM</td>
</tr>
<tr>
<td>Roulette Wheel Selection</td>
<td>RWS</td>
<td>Best Selection</td>
<td>BS</td>
</tr>
<tr>
<td>Random Population Percentage</td>
<td>RPP</td>
<td>Mutation Probability</td>
<td>MP</td>
</tr>
<tr>
<td>Crossover Probability</td>
<td>CP</td>
<td>Position Based Crossover Probability</td>
<td>PBCP</td>
</tr>
</tbody>
</table>
It has been observed that for maximizing similarity in cell loading, modified version of p-median model is very efficient considering the duration of execution and also the results obtained. It gives the optimal result in a very short time. Genetic approach is not a good method compared to math modeling for that problem. However, while performing experimentation, it has been observed that maximizing product similarity for each cell does not make product sequencing easier. Instead, minimizing product similarity in each cell turned out to be the best option to find better product sequence. It is like filling up two bowls with stones. When big stones (maximize similarity) are put all together in the first bowl and smaller stones (maximize similarity) in the second bowl, it may not be possible to fit them all. However, when both bowls are filled with small and big stones together (minimize similarity), then it will be possible to fit them to two bowls.
The program that handles both cell loading and product sequencing at the same time did not work well. Separating the problem into two stages gave better results. So, it can also be concluded that dividing a big problem into small pieces (cell loading + product sequencing) is a good idea.

Lastly, as a future work, more experimentation with more data sets (and even problems with more jobs) can be solved in order to see the effect of problem size and also to see the effect of parameters better. Product sequencing objective (minimize sum of periods where available manpower is exceeded) may be changed to minimize total deviation (sum of manpower levels where available manpower is exceeded given as objective 1 below) and also to minimize maximum deviation (maximum of manpower levels where available manpower is exceeded given as objective 2 below), and results can be analyzed. It is also possible to minimize a couple objectives together at the same time which will lead to multi-objective optimization.

\[
\text{Objective } 1 = \sum_{t=1}^{T} \max (0, \text{manpower required}_{t} - \text{manpower available}_{t})
\]

\[
\text{Objective } 2 = \max(d_1, d_2, d_3 ... d_T) \text{ for periods } 1, 2, ... T
\]

where \(d_t = \max(0, \text{manpower required}_{t} - \text{manpower available}_{t})\)
REFERENCES


Group technology and cellular manufacturing: A state-of-the-art synthesis of research
and practice / editors, p 1-6, Nallan C. Suresh and John M. Kay (c1998). In

Scheduling System. Production and Inventory Management. (pp 47-51).


Ohno, T., 1912-. (c1988). Toyota seisan hoshiki. English; Toyota production system :
Beyond large-scale production / Taiichi Ohno ; foreword by Norman Bodek.

manufacturing. Colombus, OH: Group technology/cellular manufacturing world
symposium.


Rico. In Suresh N. C. & Kay J. M. (Eds.). Group technology and cellular
manufacturing: A state-of-the-art synthesis of research and practice. Boston :
Kluwer Academic.


APPENDIX A: Opl Codes for Example Manpower Allocation Problem

/****************************************************
* OPL 5.5 Model
* Author: Fatih
* Creation Date: 2/23/2008 at 6:08 PM
****************************************************/
int NBOperations = 5;
range Operations = 1..NBOperations;

//mi is the number of machines needed for operation operation i
dvar int+ m[Operations] in 0..maxint;

//Operation Times
float OperationTimes[Operations] = [0.6 , 1 , 0.8 , 1.2 , 0.4];

//R is the production rate
dvar int+ R in 0..maxint;

maximize R;

subject to
{
  forall (o in Operations) 
  m[o]/OperationTimes[o] >= R;

  //8 is the upper limit for the number of workers for all operations.
  forall (o in Operations) 
  m[o] <= 8;

  //20 is the total number of workers we have in a cell
  sum (o in Operations) 
  m[o] <= 20;
}

Final solution with objective = 5:
R = 5;
m = [3 5 4 6 2];
APPENDIX B: Opl Codes for Example Cell Loading Problem

/*********************************************
* OPL 5.5 Model
* Author: Fatih
* Creation Date: 3/2/2008 at 3:18 PM
*********************************************/

int NbProducts = 10;

range Products = 1..NbProducts;

dvar int+ x[Products][Products] in 0..1;

//similarity coefficients
float SimilarityCoefficients[Products][Products] =
[ [ 0 , 0.666666667 , 0.6 , 0.695652174 , 0.625 , 0.6 , 0.625 , 0.739130435 , 0.772727273 , 0.538461538 ] ,
  [ 0.666666667 , 0 , 0.6 , 0.695652174 , 0.6 , 0.695652174 , 0.904761905 , 0.772727273 , 0.739130435 ] ,
  [ 0.6 , 0.6 , 0 , 0.392857143 , 0.695652174 , 1 , 0.695652174 , 0.6 , 0.625 , 0.481481481 ] ,
  [ 0.695652174 , 0.56 , 0.392857143 , 0 , 0.583333333 , 0.392857143 , 0.583333333 , 0.625 , 0.652173913 , 0.5 ] ,
  [ 0.625 , 0.695652174 , 0.695652174 , 0.583333333 , 0 , 0.695652174 , 1 , 0.695652174 , 0.80952381 , 0.5 ] ,
  [ 0.6 , 0.6 , 1 , 0.392857143 , 0.695652174 , 0 , 0.695652174 , 0.6 , 0.625 , 0.481481481 ] ,
  [ 0.625 , 0.695652174 , 0.695652174 , 0.583333333 , 1 , 0.695652174 , 0 , 0.695652174 , 0.80952381 , 0.5 ] ,
  [ 0.739130435 , 0.904761905 , 0.6 , 0.625 , 0.695652174 , 0.6 , 0.695652174 , 0.80952381 , 0.625 ,
    0.80952381 , 0.772727273 , 0.625 , 0.652173913 , 0.80952381 , 0.625 , 0.80952381 , 0.772727273 , 0 , 0.625 ] ,
  [ 0.538461538 , 0.739130435 , 0.481481481 , 0.5 , 0.5 , 0.481481481 , 0.5 , 0.666666667 , 0.625 , 0 ] ];

maximize
sum (k in Products)
sum (p in Products)
x[p][k]*SimilarityCoefficients[p][k];

subject to
{
  //all products should be assigned to a cell
forall (p in Products) 
sum (k in Products) 
x[p][k]==1;

//upper limit for the number of products assigned to a cell is 5
forall (c in Products) 
forall (p in Products) 
x[p][c]<=5;

forall (p in Products) 
forall (c in Products) 
x[c][p]<=x[p][p];

forall (k in Products) 
x[k][k]==2;

Final solution with objective = 6.1364:
x = [[0 0 0 0 0 0 1 0 0]
     [0 0 0 0 0 0 1 0 0]
     [0 0 0 1 0 0 0 0 0]
     [0 0 0 0 0 1 0 0 0]
     [0 0 0 1 0 0 0 0 0]
     [0 0 0 1 0 0 0 0 0]
     [0 0 0 1 0 0 0 0 0]
     [0 0 0 0 0 0 1 0 0]
     [0 0 0 0 1 0 0 0 0]
     [0 0 0 0 0 0 1 0 0]]
APPENDIX C: Opl Codes for Example Product Sequencing Problem

/*********************************************
* OPL 5.5 Model
* Author: Fatih
* Creation Date: 2/15/2008 at 1:37 AM
*********************************************/

int NbPeriods = 10;
int NbProducts = 10;
int NbTotalperiods = 14;
int NbStages = 5;
int NbVariable = 6;

range Periods = 1..NbPeriods;
range Products = 1..NbProducts;
range Totalperiods = 1..NbTotalperiods;
range Stages = 1..NbStages;
range Variable = 1..NbVariable;

//manpower levels for jobs in all stages
int m[Stages][Products] =[
   [3, 5, 3, 2, 2, 3, 2, 4, 3, 8],
   [5, 3, 2, 6, 2, 2, 4, 3, 3],
   [4, 4, 2, 6, 5, 2, 5, 4, 5, 3],
   [6, 4, 8, 3, 5, 8, 5, 4, 5, 3],
   [2, 4, 5, 2, 5, 5, 4, 3, 3]]; 

//total of extra manpower levels
float E[Totalperiods] = [15, 11, 7, 4, 0, 0, 0, 0, 0, 0, 3.5, 6.77, 10.9, 15.8];

dvar int+ x[Products][Periods] in 0..maxint;
dvar float+ t[Totalperiods] in 0..maxint;
dvar int+ s[Totalperiods];

minimize
sum (j in Totalperiods)
s[j];

subject to
{
forall ( j in Products )
sum ( p in Periods )
x[ j ][ p ] == 1;
forall ( p in Periods)
sum ( j in Products)
x[j][p ] == 1;

sum ( p in Periods )
m[ 1 ][ p ] * x[ p ][ 1 ] + E[1] == t[1];

sum ( p in Periods )
m[ 2 ][ p ] * x[ p ][ 1 ]
+  
sum ( p in Periods)

sum ( p in Periods )
m[ 3 ][ p ] * x[ p ][ 1 ]
+  
sum ( p in Periods)
m[ 2 ][ p ] * x[ p ][ 2 ]
+  
sum ( p in Periods)

sum ( p in Periods )
m[ 4 ][ p ] * x[ p ][ 1 ]
+  
sum ( p in Periods )
m[ 3 ][ p ] * x[ p ][ 2 ]
+  
sum ( p in Periods)
m[ 2 ][ p ] * x[ p ][ 3 ]
+  
sum ( p in Periods)

forall( k in Variable)
sum ( p in Periods )
m[ 5 ][ p ] * x[ p ][ k ]
+  
sum ( p in Periods )
m[ 4 ][ p ] * x[ p ][ k+1 ]
+  
sum ( p in Periods )
m[ 3 ][ p ] * x[ p ][ k+2 ]
+  
sum ( p in Periods)
m[ 2 ][ p ] * x[ p ][ k+3 ]
\[
\sum (p \in \text{Periods}) \quad m[1][p] \times x[p][k+4] + E[k+4] = t[k+4];
\]
\[
\sum (p \in \text{Periods}) \quad m[5][p] \times x[p][7]
\]
\[
+ \quad \sum (p \in \text{Periods}) \quad m[4][p] \times x[p][8]
\]
\[
+ \quad \sum (p \in \text{Periods}) \quad m[3][p] \times x[p][9]
\]
\[
+ \quad \sum (p \in \text{Periods}) \quad m[2][p] \times x[p][10] + E[11] = t[11];
\]
\[
\sum (p \in \text{Periods}) \quad m[5][p] \times x[p][8]
\]
\[
+ \quad \sum (p \in \text{Periods}) \quad m[4][p] \times x[p][9]
\]
\[
+ \quad \sum (p \in \text{Periods}) \quad m[3][p] \times x[p][10] + E[12] = t[12];
\]
\[
\sum (p \in \text{Periods}) \quad m[5][p] \times x[p][9]
\]
\[
+ \quad \sum (p \in \text{Periods}) \quad m[4][p] \times x[p][10] + E[13] = t[13];
\]
\[
\sum (p \in \text{Periods}) \quad m[5][p] \times x[p][10] + E[14] = t[14];
\]

forall (j in Total periods)
\[
1000 * s[j] >= t[j] - 20;
\]

Final solution with objective = 0:
\[
s = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0];
\]
\[
x = [[0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]
\]
\[
[0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0]
\]
\[
[0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]
\]
\[
[0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0]
\]
\[
[0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0]
\]
\[
[0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0]
\]
\[
[0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0]
\]
\[
[0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0]
\]
\[
[0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0]
\]

[0 0 0 0 0 0 0 1]
[1 0 0 0 0 0 0 0]
[0 0 0 0 0 1 0 0 0]];

\[ \text{t} = [18 \ 17 \ 20 \ 17 \ 20 \ 20 \ 20 \ 20 \ 20 \ 20 \ 19.5 \ 19.77 \ 18.9 \ 19.8] \]
APPENDIX D: Datasets

Table 60

Data Set 1

<table>
<thead>
<tr>
<th>MANPOWER LEVELS</th>
<th>Stage 1</th>
<th>Stage 2</th>
<th>Stage 3</th>
<th>Stage 4</th>
<th>Stage 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product 0</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Product 1</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Product 2</td>
<td>3</td>
<td>8</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Product 3</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Product 4</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Product 5</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Product 6</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Product 7</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Product 8</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Product 9</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
Table 61

Data Set 2

<table>
<thead>
<tr>
<th>Product</th>
<th>Stage 1</th>
<th>Stage 2</th>
<th>Stage 3</th>
<th>Stage 4</th>
<th>Stage 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>8</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>8</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>14</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>17</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>18</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>19</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
### Table 62

Data Set 3

<table>
<thead>
<tr>
<th>Product</th>
<th>Stage 1</th>
<th>Stage 2</th>
<th>Stage 3</th>
<th>Stage 4</th>
<th>Stage 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>8</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>8</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>8</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>8</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>19</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>21</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>22</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>23</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>24</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>25</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>26</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>27</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>28</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>29</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>30</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>31</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>32</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>33</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>34</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>35</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>36</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>37</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>38</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>39</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
### Table 63

Data Set 4

<table>
<thead>
<tr>
<th>DATA SET 4 MANPOWER LEVELS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage 1</td>
</tr>
<tr>
<td>-----------------------------</td>
</tr>
<tr>
<td>Product 0</td>
</tr>
<tr>
<td>Product 1</td>
</tr>
<tr>
<td>Product 2</td>
</tr>
<tr>
<td>Product 3</td>
</tr>
<tr>
<td>Product 4</td>
</tr>
<tr>
<td>Product 5</td>
</tr>
<tr>
<td>Product 6</td>
</tr>
<tr>
<td>Product 7</td>
</tr>
<tr>
<td>Product 8</td>
</tr>
<tr>
<td>Product 9</td>
</tr>
<tr>
<td>Product 10</td>
</tr>
<tr>
<td>Product 11</td>
</tr>
<tr>
<td>Product 12</td>
</tr>
<tr>
<td>Product 13</td>
</tr>
<tr>
<td>Product 14</td>
</tr>
<tr>
<td>Product 15</td>
</tr>
<tr>
<td>Product 16</td>
</tr>
<tr>
<td>Product 17</td>
</tr>
<tr>
<td>Product 18</td>
</tr>
<tr>
<td>Product 19</td>
</tr>
<tr>
<td>Product 20</td>
</tr>
<tr>
<td>Product 21</td>
</tr>
<tr>
<td>Product 22</td>
</tr>
<tr>
<td>Product 23</td>
</tr>
<tr>
<td>Product 24</td>
</tr>
<tr>
<td>Product 25</td>
</tr>
<tr>
<td>Product 26</td>
</tr>
<tr>
<td>Product 27</td>
</tr>
<tr>
<td>Product 28</td>
</tr>
<tr>
<td>Product 29</td>
</tr>
<tr>
<td>Product 30</td>
</tr>
<tr>
<td>Product 31</td>
</tr>
<tr>
<td>Product 32</td>
</tr>
<tr>
<td>Product 33</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>Product 34</td>
</tr>
<tr>
<td>Product 35</td>
</tr>
<tr>
<td>Product 36</td>
</tr>
<tr>
<td>Product 37</td>
</tr>
<tr>
<td>Product 38</td>
</tr>
<tr>
<td>Product 39</td>
</tr>
<tr>
<td>Product 40</td>
</tr>
<tr>
<td>Product 41</td>
</tr>
<tr>
<td>Product 42</td>
</tr>
<tr>
<td>Product 43</td>
</tr>
<tr>
<td>Product 44</td>
</tr>
<tr>
<td>Product 45</td>
</tr>
<tr>
<td>Product 46</td>
</tr>
<tr>
<td>Product 47</td>
</tr>
<tr>
<td>Product 48</td>
</tr>
<tr>
<td>Product 49</td>
</tr>
<tr>
<td>Product 50</td>
</tr>
<tr>
<td>Product 51</td>
</tr>
<tr>
<td>Product 52</td>
</tr>
<tr>
<td>Product 53</td>
</tr>
<tr>
<td>Product 54</td>
</tr>
<tr>
<td>Product 55</td>
</tr>
<tr>
<td>Product 56</td>
</tr>
<tr>
<td>Product 57</td>
</tr>
<tr>
<td>Product 58</td>
</tr>
<tr>
<td>Product 59</td>
</tr>
<tr>
<td>Product 60</td>
</tr>
<tr>
<td>Product 61</td>
</tr>
<tr>
<td>Product 62</td>
</tr>
<tr>
<td>Product 63</td>
</tr>
<tr>
<td>Product 64</td>
</tr>
<tr>
<td>Product 65</td>
</tr>
<tr>
<td>Product 66</td>
</tr>
<tr>
<td>Product 67</td>
</tr>
<tr>
<td>Product 68</td>
</tr>
<tr>
<td>Product 69</td>
</tr>
<tr>
<td>Product 70</td>
</tr>
<tr>
<td>Product 71</td>
</tr>
<tr>
<td>Product 72</td>
</tr>
<tr>
<td>Product 73</td>
</tr>
<tr>
<td>Product 74</td>
</tr>
<tr>
<td>Product 75</td>
</tr>
<tr>
<td>Product 76</td>
</tr>
<tr>
<td>Product 77</td>
</tr>
<tr>
<td>Product 78</td>
</tr>
<tr>
<td>Product 79</td>
</tr>
</tbody>
</table>
APPENDIX E: Opl Codes for 10, 20, 40 Jobs Cell Loading Problems

/*********************************************
* OPL 6.0 Model
* Author: fy298906
* Creation Date: Nov 13, 2008 at 3:02:56 PM
*********************************************/

//CELL LOADING FOR 10 JOBS
int NbProducts = 10;
range Products = 1..NbProducts;
dvar int+ x[Products][Products] in 0..1;
int convertedObjective;
int cellArray[Products];

//similarity coefficients
float SimilarityCoefficients[Products][Products] =
[  [ 0 ,  0.666666667 ,  0.6    ,  0.666666667    ,
  0.739130435 ,  0.666666667 ,  0.6    ,  0.739130435    ,
  0.739130435    ,  0.538461538 ]   ,
  [ 0.666666667 ,  0      ,  0.6    ,  0.6    ,  0.6    ,
  0.739130435    ,  0.818181818 ,  0.904761905    ,  0.818181818    ,
  0.739130435 ]   ,
  [ 0.6    ,  0.6    ,  0      ,  0.6    ,  0.666666667    ,
  0.538461538    ,  0.6    ,  0.666666667    ,  0.538461538    ,
  0.538461538 ]   ,
  [ 0.666666667 ,  0.6    ,  0.6    ,  0      ,
  0.481481481    ,  0.739130435    ,  0.739130435    ,  0.666666667    ,
  0.6    ,  0.481481481 ]   ,
  [ 0.739130435 ,  0.6    ,  0.666666667    ,
  0.481481481    ,  0      ,  0.6    ,  0.481481481    ,
  0.666666667    ,  0.666666667    ,  0.538461538 ]   ,
  [ 0.666666667 ,  0.739130435    ,  0.538461538    ,
  0.739130435    ,  0.6    ,  0      ,  0.739130435    ,
  0.739130435    ,  0.538461538 ]   ,
  [ 0.6    ,  0.818181818 ,  0.6    ,  0.739130435    ,
  0.481481481    ,  0.739130435    ,  0      ,  0.739130435    ,
  0.739130435 ]   ,
  [ 0.739130435 ,  0.904761905    ,  0.666666667    ,
  0.666666667    ,  0.666666667    ,  0.739130435    ,  0.739130435    ,
  0.739130435 ]   ,
  [ 0.818181818 ,  0.538461538 ,  0.666666667    ,
  0.666666667    ,  0.666666667    ,  0.739130435    ,  0.739130435    ,
  0.739130435 ]   ,
  [ 0.739130435 ,  0.818181818 ,  0.538461538    ,
  0.6    ,  0.666666667    ,  0.818181818    ,  0.739130435    ,
  0.818181818 ]   ,
  [ 0.538461538    ,  0.739130435    ,  0.538461538    ,
  0.481481481    ,  0.538461538    ,  0.538461538    ,  0.666666667    ,
  0.666666667    ,  0.666666667    ,  0 ]   ]   ;

maximize
sum (k in Products)
sum (p in Products)
x[p][k]*SimilarityCoefficients[p][k];

subject to
{
    // all products should be assigned to a cell
    forall ( p in Products)
    sum ( k in Products )
    x[p][k]==1;

    // upper limit for the number of products assigned to a cell is 5
    forall ( c in Products )
    sum( p in Products )
    x[p][c]<=5;

    forall ( p in Products )
    forall ( c in Products )
    x[c][p]<=x[p][p];

    // total number of cells equals to 2
    sum (k in Products)
    x[k][k]==2;
}

execute
{
    for (var k in Products)
    for (var s in Products)
    {
        if (x[k][s] == 1)
        {
            cellArray[k] = s;
        }
    }

    for (var l in Products)
    for (var m in Products)
    {
        if (cellArray[l] ==cellArray[m])
        {
            convertedObjective = convertedObjective + SimilarityCoefficients[l][m];
        }
    }

    writeln("convertedObjective = ", convertedObjective);
}
\[ x = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}; \\
\]

// solution (optimal) with objective 5.952719745
convertedObjective = 27.74980511

iscriminated 

/********************************************
* OPL 6.0 Model
* Author: fy298906
* Creation Date: Nov 13, 2008 at 3:35:01 PM
********************************************/

//CELL LOADING FOR 20 JOBS
int NbProducts = 20;
range Products = 1..NbProducts;
dvar int+ x[Products][Products] in 0..1;
int convertedObjective;
int cellArray[Products];

//similarity coefficients
float SimilarityCoefficients[Products][Products] =
\[
\begin{bmatrix}
0 & 1 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 \\
0.6 & 0.6 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 \\
0.739130435 & 0.739130435 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 \\
0.6 & 0.6 & 0.739130435 & 0.739130435 & 0.739130435 & 0.739130435 & 0.739130435 & 0.739130435 & 0.739130435 & 0.739130435 \\
0.739130435 & 0.739130435 & 0.739130435 & 0.538461538 & 0.538461538 & 0.538461538 & 0.538461538 & 0.538461538 & 0.538461538 & 0.538461538 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 0 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 \\
0.6 & 0.6 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 \\
0.739130435 & 0.739130435 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 & 0.66666667 \\
0.6 & 0.6 & 0.739130435 & 0.739130435 & 0.739130435 & 0.739130435 & 0.739130435 & 0.739130435 & 0.739130435 & 0.739130435 \\
0.739130435 & 0.739130435 & 0.739130435 & 0.538461538 & 0.538461538 & 0.538461538 & 0.538461538 & 0.538461538 & 0.538461538 & 0.538461538 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.66666667 & 0.66666667 & 0 & 1 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\
0.739130435 & 0.739130435 & 0.818181818 & 0.818181818 & 0.818181818 & 0.818181818 & 0.818181818 & 0.818181818 & 0.818181818 & 0.818181818 \\
\end{bmatrix}
\]
maximize
sum (k in Products)
sum (p in Products) x[p][k]*SimilarityCoefficients[p][k];

subject to
{
// all products should be assigned to a cell
forall ( p in Products)
sum ( k in Products )
x[p][k]==1;

// upper limit for the number of products assigned to a cell is 10
forall ( c in Products )
sum( p in Products )
x[p][c]<=10;
forall ( p in Products )
forall ( c in Products )
x[c][p]<=x[p][p];

// total number of cells equals to 2
sum (k in Products)
x[k][k]==2;

} execute
{
    for (var k in Products)
    for (var s in Products)
    {
        if (x[k][s] == 1)
        {
            cellArray[k] = s;
        }
    }

    for (var l in Products)
    for (var m in Products)
    {
        if (cellArray[l] ==cellArray[m])
        {
            convertedObjective = convertedObjective + SimilarityCoefficients[l][m];
        }
    }
    writeln("cellArray = ", cellArray);
    writeln("convertedObjective = ", convertedObjective);
}

// solution (optimal) with objective 13.90543949
x = [[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
     [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
     [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
     [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
     [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
     [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
// solution (optimal) with objective 13.90543949
CellArray = [1 1 3 3 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3]
ConvertedObjective = 130.9992204400002

/*********************************************/
* OPL 6.0 Model
* Author: fy298906
* Creation Date: Nov 13, 2008 at 4:02:06 PM
*********************************************/

//CELL LOADING FOR 40 JOBS
int NbProducts = 40;
range Products = 1..NbProducts;
dvar int+ x[Products][Products] in 0..1;
int ConvertedObjective;
int CellArray[Products];

//similarity coefficients
float SimilarityCoefficients[Products][Products] = 
[ [ 0 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 3 3 3 3 3 3 3 3 3 3 3 ] ];
\[
\text{maximize} \\
\sum (k \text{ in Products}) \\
\sum (p \text{ in Products}) \\
x[p][k]*\text{SimilarityCoefficients}[p][k];
\]
subject to
{

//all products should be assigned to a cell
forall ( p in Products)
sum ( k in Products )
x[p][k]==1;

//upper limit for the number of products assigned to a cell is 20
forall ( c in Products )
sum( p in Products )
x[p][c]<=20;

forall ( p in Products )
forall ( c in Products )
x[c][p]<=x[p][p];

// total number of cells equals to 2
sum (k in Products)
x[k][k]==2;
}

execute
{

for (var k in Products)
for (var s in Products)
{
  if (x[k][s] == 1)
  {
    cellArray[k] = s;
  }
}

for (var l in Products)
for (var m in Products)
{
  if (cellArray[l] ==cellArray[m])
  {
    convertedObjective = convertedObjective +
    SimilarityCoefficients[l][m];
  }
}

writeln("cellArray = ", cellArray);
writeln("convertedObjective = ", convertedObjective);
}

// solution (optimal) with objective 29.81087898
// solution (optimal) with objective 29.81087898
cellArray = [3 3 3 3 6 6 6 6 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6 6 6 6]
convertedObjective = 563.9968817599995
APPENDIX F: Opl Codes for 10, 20, 40, 80 Jobs Product Sequencing Problems

/*****************************************************/
* OPL 6.0 Model
* Author: fy298906
* Creation Date: Nov 13, 2008 at 5:07:58 PM
*****************************************************/

//10 JOBS PRODUCT SEQUENCING

int NbPeriods = 10;
int NbProducts = 10;
int NbTotalperiods = 14;
int NbStages = 5;
int NbVariable = 6;

range Periods = 1..NbPeriods;
range Products = 1..NbProducts;
range Totalperiods = 1..NbTotalperiods;
range Stages = 1..NbStages;
range Variable = 1..NbVariable;

//manpower levels for jobs in all stages
int m[Stages][Products] = 
    [ 3 , 5 , 3 , 2 , 3 , 2 , 5 , 4 , 4 , 8 ] ,
    [ 5 , 3 , 8 , 4 , 6 , 3 , 3 , 4 , 3 , 3 ] ,
    [ 4 , 4 , 2 , 2 , 6 , 5 , 2 , 4 , 5 , 3 ] ,
    [ 6 , 4 , 3 , 6 , 3 , 5 , 5 , 4 , 3 , 3 ] ,
    [ 2 , 4 , 4 , 6 , 2 , 5 , 5 , 4 , 3 , 3 ] ] ;

//sum of extra manpower levels
float E[Totalperiods] = [15, 11, 7, 4, 0, 0, 0, 0, 0, 0, 3.9, 
    8.08181818181818, 11.9984848484848, 16.3061771561772] ;

dvar int+ x[Products][Periods] in 0..maxint;
dvar float+ t[Totalperiods] in 0..maxint;
dvar int+ s[Totalperiods];

minimize
    sum (j in Totalperiods)
    s[j];

subject to
    {
    forall( j in Products )
    sum ( p in Periods )
    x[ j ][ p ] == 1;
    

forall (p in Periods)
sum (j in Products)
x[j][p] == 1;

sum (p in Periods)
m[1][p] * x[p][1] + E[1] == t[1];

sum (p in Periods)
m[2][p] * x[p][1] + sum (p in Periods)

sum (p in Periods)
m[3][p] * x[p][1] + sum (p in Periods)
m[2][p] * x[p][2] + sum (p in Periods)

sum (p in Periods)
m[4][p] * x[p][1] + sum (p in Periods)
m[3][p] * x[p][2] + sum (p in Periods)
m[2][p] * x[p][3] + sum (p in Periods)

forall (k in Variable)
sum (p in Periods)
m[5][p] * x[p][k] + sum (p in Periods)
m[4][p] * x[p][k+1] + sum (p in Periods)
m[3][p] * x[p][k+2] + sum (p in Periods)
m[2][p] * x[p][k+3] + sum (p in Periods)
m[1][p] * x[p][k+4] + E[k+4] == t[k+4];

sum (p in Periods)
m[5][p] * x[p][7]
\[\begin{align*}
&\text{sum (p in Periods)} \quad m[4][p] \times x[p][8] \\
&+ \text{sum (p in Periods)} \quad m[3][p] \times x[p][9] \\
&+ \text{sum (p in Periods)} \quad m[2][p] \times x[p][10] + E[11] = t[11]; \\
&\text{sum (p in Periods)} \quad m[5][p] \times x[p][8] \\
&+ \text{sum (p in Periods)} \quad m[4][p] \times x[p][9] \\
&+ \text{sum (p in Periods)} \quad m[3][p] \times x[p][10] + E[12] = t[12]; \\
&\text{sum (p in Periods)} \quad m[5][p] \times x[p][9] \\
&+ \text{sum (p in Periods)} \quad m[4][p] \times x[p][10] + E[13] = t[13]; \\
&\text{sum (p in Periods)} \quad m[5][p] \times x[p][10] + E[14] = t[14]; \\
&\text{forall (j in Totalperiods)} \quad 1000 \times s[j] \geq t[j] - 20;
\end{align*}\]

// solution (optimal) with objective 1
s = [0 0 0 0 0 0 0 1 0 0 0 0 0 0];
x = [[0 0 0 0 0 1 0 0 0 0 0 0] \\
[0 1 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 1 0 0 0 0] \\
[0 0 0 0 1 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0] \\
[0 0 1 0 0 0 0 0 0 0] \\
[0 0 0 1 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0] \\
[1 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 1] \\
[0 0 0 0 0 0 0 0 1 0 0]];
t = [19 20 16 20 18 20 31 20 18 19.9 19.082 18.998 19.306];

//****************************************************************************** 
* OPL 6.0 Model
* Author: fy298906
* Creation Date: Nov 13, 2008 at 6:44:41 PM
******************************************************************************

//20 JOBS PRODUCT SEQUENCING
int NbPeriods = 20;
int NbProducts = 20;
int NbTotalperiods = 24;
int NbStages = 5;
int NbVariable = 16;

range Periods = 1..NbPeriods;
range Products = 1..NbProducts;
range Totalperiods = 1..NbTotalperiods;
range Stages = 1..NbStages;
range Variable = 1..NbVariable;

//manpower levels for jobs in all stages
int m[Stages][Products] =
[ [ 3, 3, 5, 5, 3, 3, 2, 2, 3, 3, 2, 2, 5, 5, 4, 4, 4, 4, 8, 8 ],
[ 5, 5, 3, 3, 8, 8, 4, 4, 6, 6, 3, 3, 3, 3, 4, 4, 3, 3, 3, 3 ],
[ 4, 4, 4, 4, 2, 2, 2, 2, 6, 6, 5, 5, 2, 2, 4, 4, 5, 5, 3, 3 ],
[ 6, 6, 4, 4, 3, 3, 6, 6, 3, 3, 5, 5, 5, 5, 4, 4, 5, 5, 3, 3 ],
[ 2, 2, 4, 4, 4, 4, 4, 4, 6, 6, 2, 2, 5, 5, 5, 5, 4, 4, 3, 3 ] ];

//total of extra manpower levels
float E[Totalperiods] = [15, 11, 7, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.9, 8.09047619047619, 11.908658008658, 16.2564840956145 ];

dvar int+ x[Products][Periods] in 0..maxint;
dvar float+ t[Totalperiods] in 0..maxint;
dvar int+ s[Totalperiods];

minimize
sum (j in Totalperiods)
s[j];

subject to
{
forall( j in Products )
sum ( p in Periods )
x[ j ][ p ] == 1;
forall( p in Periods)
sum ( j in Products)
x[ j ][ p ] == 1;

sum ( p in Periods )
m[ 1 ][ p ] * x[ p ][ 1 ] + E[1] == t[1];

sum ( p in Periods )
m[ 2 ][ p ] * x[ p ][ 1 ]
+
sum ( p in Periods )

sum ( p in Periods )
m[ 3 ][ p ] * x[ p ][ 1 ]
+
sum ( p in Periods )
m[ 2 ][ p ] * x[ p ][ 2 ]
+
sum ( p in Periods )

sum ( p in Periods )
m[ 4 ][ p ] * x[ p ][ 1 ]
+
sum ( p in Periods )
m[ 3 ][ p ] * x[ p ][ 2 ]
+
sum ( p in Periods )
m[ 2 ][ p ] * x[ p ][ 3 ]
+
sum ( p in Periods )

forall (k in Variable)
sum ( p in Periods )
m[ 5 ][ p ] * x[ p ][ k ]
+
sum ( p in Periods )
m[ 4 ][ p ] * x[ p ][ k+1 ]
+
sum ( p in Periods )
m[ 3 ][ p ] * x[ p ][ k+2 ]
+
sum ( p in Periods )
m[ 2 ][ p ] * x[ p ][ k+3 ]
+
sum ( p in Periods )
m[ 1 ][ p ] * x[ p ][ k+4 ] + E[k+4] == t[ k+4 ];

sum ( p in Periods )
m[ 5 ][ p ] * x[ p ][ 17 ]
+
sum ( p in Periods )
m[ 4 ][ p ] * x[ p ][ 18 ]
\[
+ \sum_{p \in \text{Periods}} m[3][p] \times x[p][19] \\
+ \sum_{p \in \text{Periods}} m[2][p] \times x[p][20] + E[21] = t[21]; \\
\sum_{p \in \text{Periods}} m[5][p] \times x[p][18] \\
+ \sum_{p \in \text{Periods}} m[4][p] \times x[p][19] \\
+ \sum_{p \in \text{Periods}} m[3][p] \times x[p][20] + E[22] = t[22]; \\
\sum_{p \in \text{Periods}} m[5][p] \times x[p][19] \\
+ \sum_{p \in \text{Periods}} m[4][p] \times x[p][20] + E[23] = t[23]; \\
\sum_{p \in \text{Periods}} m[5][p] \times x[p][20] + E[24] = t[24]; \\
\forall (j \in \text{Totalperiods}) 1000 \times s[j] \geq t[j] - 20;
\]

// solution (optimal) with objective 1
s = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0];
x = [[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] \\
[0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0] \\
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] \\
};
t = [18 19 20 20 20 20 19 20 19 20 18 20 18 20 33 20 17 20 18.9
19.09 18.909 19.256];

/*********************************************
* OPL 6.0 Model
* Author: fy298906
* Creation Date: Nov 13, 2008 at 7:14:30 PM
*********************************************/

// 40 JOBS PRODUCT SEQUENCING
int NbPeriods = 40;
int NbProducts = 40;
int NbTotalperiods = 44;
int NbStages = 5;
int NbVariable = 36;

range Periods = 1..NbPeriods;
range Products = 1..NbProducts;
range Totalperiods = 1..NbTotalperiods;
range Stages = 1..NbStages;
range Variable = 1..NbVariable;

//manpower levels for jobs in all stages
int m[Stages][Products] =
[ [ 3, 3, 3, 3, 5, 5, 5, 5, 3, 3,
2, 2, 5, 5, 5, 5, 4, 4, 4, 4,
4, 4, 4, 8, 8, 8, 8 ] ,
[ 5, 5, 5, 5, 3, 3, 3, 3, 8, 8,
8, 4, 4, 4, 4, 6, 6, 6, 3,
3, 3, 3, 3, 3, 3, 4, 4, 4, 4,
3, 3, 3, 3, 3, 3, 3 ] ,
[ 4, 4, 4, 4, 4, 4, 4, 4, 2, 2,
2, 2, 2, 2, 2, 6, 6, 6, 5,
5, 5, 2, 2, 2, 2, 4, 4, 4, 4,
5, 5, 5, 3, 3, 3, 3 ] ,
[ 6, 6, 6, 6, 4, 4, 4, 4, 3, 3,
3, 6, 6, 6, 6, 6, 3, 3, 3, 3,
5, 5, 5, 5, 5, 4, 4, 4, 4,
5, 5, 5, 5, 5, 3, 3, 3 ] ,
[ 2, 2, 2, 2, 4, 4, 4, 4, 4, 4,
4, 4, 6, 6, 6, 6, 6, 2, 2, 2,
5, 5, 5, 5, 5, 4, 4, 4, 4,
3, 3, 3, 3, 3, 3, 3 ] ];

//total of extra manpower levels
float E[Totalperiods] = [15, 11, 7, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.9, 8.09512195121951, 11.8570267131243, 16.2291197363801];

dvar int+ x[Products][Periods] in 0..maxint;
dvar float+ t[Totalperiods] in 0..maxint;
dvar int+ s[Totalperiods];

minimize
sum (j in Totalperiods)
s[j];

subject to
{
forall( j in Products )
sum ( p in Periods )
x[ j ][ p ] == 1;
forall( p in Periods )
sum ( j in Products )
x[ j ][ p ] == 1;

sum ( p in Periods )
m[ 1 ][ p ] * x[ p ][ 1 ] + E[1] == t[1];

sum ( p in Periods )
m[ 2 ][ p ] * x[ p ][ 1 ]
+ sum ( p in Periods )

sum ( p in Periods )
m[ 3 ][ p ] * x[ p ][ 1 ]
+ sum ( p in Periods )
m[ 2 ][ p ] * x[ p ][ 2 ]
+ sum ( p in Periods )

sum ( p in Periods )
m[ 4 ][ p ] * x[ p ][ 1 ]
+ sum ( p in Periods )
m[ 3 ][ p ] * x[ p ][ 2 ]
+ sum ( p in Periods )
m[ 2 ][ p ] * x[ p ][ 3 ]
+ sum ( p in Periods )
forall(k in Variable)
sum ( p in Periods )
m[ 5 ][ p ] * x[ p ][ k ]
+ sum ( p in Periods )
m[ 4 ][ p ] * x[ p ][ k+1 ]
+ sum ( p in Periods )
  m[ 3 ][ p ] * x[ p ][ k+2 ]
+ sum ( p in Periods )
  m[ 2 ][ p ] * x[ p ][ k+3 ]
+ sum ( p in Periods )
  m[ 1 ][ p ] * x[ p ][ k+4 ] +E[k+4] == t[k+4 ];

sum ( p in Periods )
  m[ 5 ][ p ] * x[ p ][ 37 ]
+ sum ( p in Periods )
  m[ 4 ][ p ] * x[ p ][ 38 ]
+ sum ( p in Periods )
  m[ 3 ][ p ] * x[ p ][ 39 ]
+ sum ( p in Periods )
  m[ 2 ][ p ] * x[ p ][ 40 ] + E[41] == t[41];

sum ( p in Periods )
  m[ 5 ][ p ] * x[ p ][ 38 ]
+ sum ( p in Periods )
  m[ 4 ][ p ] * x[ p ][ 39 ]
+ sum ( p in Periods )
  m[ 3 ][ p ] * x[ p ][ 40 ] + E[42] == t[42];

sum ( p in Periods )
  m[ 5 ][ p ] * x[ p ][ 39 ]
+ sum ( p in Periods )
  m[ 4 ][ p ] * x[ p ][ 40 ] + E[43] == t[43];

sum ( p in Periods )
  m[ 5 ][ p ] * x[ p ][ 40 ] + E[44] == t[44];

forall (j in Totalperiods)
  1000*s[j]>=t[j]-20;
}

Program is out of memory after 1.6 hours. Best result found is 4.

/******************************************
* OPL 6.0 Model
* Author: fy298906
* Creation Date: Nov 18, 2008 at 4:12:15 PM
******************************************/
182

//80 JOBS PRODUCT SEQUENCING

int NbPeriods = 80;
int NbProducts = 80;
int NbTotalperiods = 84;
int NbStages = 5;
int NbVariable = 76;

range Periods = 1..NbPeriods;
range Products = 1..NbProducts;
range Totalperiods = 1..NbTotalperiods;
range Stages = 1..NbStages;
range Variable = 1..NbVariable;

//manpower levels for jobs in all stages
int m[Stages][Products] =
[
  [ 3, 3, 3, 3, 3, 3, 3, 5, 5,
    5, 5, 5, 5, 5, 5, 3, 3, 3,
    3, 3, 2, 2, 2, 2, 2, 2, 2,
    3, 3, 3, 3, 3, 3, 2, 2, 2,
    2, 2, 2, 2, 2, 5, 5, 5, 5,
    5, 5, 4, 4, 4, 4, 4, 4, 4,
    4, 4, 4, 4, 4, 4, 8, 8, 8,
    8, 8, 8, 8 ] ,
  [ 5, 5, 5, 5, 5, 5, 8, 8, 8,
    3, 3, 3, 3, 3, 3, 3, 3, 3,
    8, 8, 4, 4, 4, 4, 4, 4, 4,
    6, 6, 6, 6, 6, 6, 6, 3, 3,
    3, 3, 4, 4, 4, 4, 4, 4, 4,
    3, 3, 3, 3, 3, 3, 3, 3, 3,
    3, 3, 3, 3, 3, 3, 3, 3, 3,
    3, 3, 3, 3 ] ,
  [ 4, 4, 4, 4, 4, 4, 4, 4, 4,
    4, 4, 4, 4, 4, 4, 2, 2, 2,
    2, 2, 2, 2, 2, 2, 2, 2, 2,
    6, 6, 6, 6, 6, 6, 6, 5, 5,
    5, 5, 5, 5, 5, 2, 2, 2, 2,
    2, 2, 4, 4, 4, 4, 4, 4, 4,
    5, 5, 5, 5, 5, 5, 5, 3, 3,
    3, 3, 3, 3 ] ,
  [ 6, 6, 6, 6, 6, 6, 6, 6, 4,
    4, 4, 4, 4, 4, 4, 3, 3, 3,
    3, 3, 6, 6, 6, 6, 6, 6, 6,
    3, 3, 3, 3, 3, 3, 3, 3, 3,
    5, 5, 5, 5, 5, 5, 5, 5, 5,
    5, 5, 4, 4, 4, 4, 4, 4, 4,
    5, 5, 5, 5, 5, 5, 3, 3, 3,
    3, 3, 3, 3 ] ,
  [ 2, 2, 2, 2, 2, 2, 2, 2, 4,
    4, 4, 4, 4, 4, 4, 4, 4, 4,
    4, 4, 4, 4, 4, 4, 4, 4, 4,
    2, 2, 2, 2, 2, 2, 2, 2, 2,
    5, 5, 5, 5, 5, 5, 5, 5, 5,
    5, 5, 4, 4, 4, 4, 4, 4, 4,
    5, 5, 4, 4, 4, 4, 4, 4, 4,
    3, 3, 3, 3 ] ,
];
//total of extra manpower levels
float E[Totalperiods] = [15, 11, 7, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.9, 8.097531, 11.82924, 16.21478];

dvar int+ x[Products][Periods] in 0..maxint;
dvar float+ t[Totalperiods] in 0..maxint;
dvar int+ s[Totalperiods];

minimize
sum (j in Totalperiods) s[j];

subject to
{
forall( j in Products )
sum ( p in Periods ) x[j][p] == 1;

dforall( p in Periods )
sum ( j in Products ) x[j][p] == 1;

sum ( p in Periods ) m[1][p] * x[p][1] + E[1] == t[1];

sum ( p in Periods ) m[2][p] * x[p][1] +

sum ( p in Periods ) m[3][p] * x[p][1] +
   sum ( p in Periods ) m[2][p] * x[p][2] +
   sum ( p in Periods ) m[1][p] * x[p][3] + E[3] == t[3];

sum ( p in Periods ) m[4][p] * x[p][1] +
   sum ( p in Periods ) m[3][p] * x[p][2] +
   sum ( p in Periods ) m[2][p] * x[p][3]
\[ \sum_{p \in \text{Periods}} m[1][p] \cdot x[p][4] + E[4] = t[4]; \]

\[ \forall (k \in \text{Variable}) \]
\[ \sum_{p \in \text{Periods}} m[5][p] \cdot x[p][k] + \sum_{p \in \text{Periods}} m[4][p] \cdot x[p][k+1] + \sum_{p \in \text{Periods}} m[3][p] \cdot x[p][k+2] + \sum_{p \in \text{Periods}} m[2][p] \cdot x[p][k+3] + \sum_{p \in \text{Periods}} m[1][p] \cdot x[p][k+4] + E[k+4] = t[k+4]; \]

\[ \sum_{p \in \text{Periods}} m[5][p] \cdot x[p][77] + \sum_{p \in \text{Periods}} m[4][p] \cdot x[p][78] + \sum_{p \in \text{Periods}} m[3][p] \cdot x[p][79] + \sum_{p \in \text{Periods}} m[2][p] \cdot x[p][80] + E[81] = t[81]; \]

\[ \sum_{p \in \text{Periods}} m[5][p] \cdot x[p][78] + \sum_{p \in \text{Periods}} m[4][p] \cdot x[p][79] + \sum_{p \in \text{Periods}} m[3][p] \cdot x[p][80] + E[82] = t[82]; \]

\[ \sum_{p \in \text{Periods}} m[5][p] \cdot x[p][79] + \sum_{p \in \text{Periods}} m[4][p] \cdot x[p][80] + E[83] = t[83]; \]

\[ \sum_{p \in \text{Periods}} m[5][p] \cdot x[p][80] + E[84] = t[84]; \]

\[ \forall (j \in \text{Totalperiods}) \]
\[ 1000 \cdot s[j] \geq t[j] - 20; \]
APPENDIX G: Calculation of Converted Objective for Cell Loading

Products 1, 3, 4, 5, 6 are assigned to the first cell and products 2, 7, 8, 9, 10 are assigned to the second cell. Converted objective is calculated below.

Table 64

**Calculation of Similarities in Two Cells**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell1/cell2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.6</td>
<td>0.67</td>
<td>0.74</td>
<td>0.67</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.6</td>
<td>0</td>
<td>0</td>
<td>0.67</td>
<td>0.54</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0.67</td>
<td>0</td>
<td>0.6</td>
<td>0.48</td>
<td>0.74</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0.74</td>
<td>0</td>
<td>0.67</td>
<td>0.48</td>
<td>0</td>
<td>0.6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0.67</td>
<td>0</td>
<td>0.54</td>
<td>0.74</td>
<td>0.6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 64 - a: Similarity Calculations in Cell 1

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell1/cell2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.82</td>
<td>0.9</td>
<td>0.82</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0.82</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.74</td>
<td>0.74</td>
<td>0.74</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0.9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.74</td>
<td>0</td>
<td>0.82</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0.82</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.74</td>
<td>0.82</td>
<td>0</td>
<td>0.67</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0.74</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.67</td>
<td>0.67</td>
<td>0.67</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 64 - b: Similarity Calculations in Cell 2

<table>
<thead>
<tr>
<th></th>
<th>SUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUM</td>
<td>27.74981</td>
</tr>
</tbody>
</table>