Integration of Genetic Algorithms to Engineering Optimization Problems

A Thesis presented to
The Faculty of the College of Engineering and Technology
Ohio University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

by
Jay-Shinn Tsai
March, 1993
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>ABSTRACT</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAPTER 1: INTRODUCTION</td>
<td>2</td>
</tr>
<tr>
<td>CHAPTER 2: LITERATURE REVIEW</td>
<td>5</td>
</tr>
<tr>
<td>2.0 Genetic algorithms</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Genetic algorithms and artificial neural networks</td>
<td>8</td>
</tr>
<tr>
<td>2.2 Genetic algorithms and inverse kinematics for robots with multiple degrees of freedom</td>
<td>11</td>
</tr>
<tr>
<td>2.3 Genetic algorithms and job shop scheduling</td>
<td>13</td>
</tr>
<tr>
<td>CHAPTER 3: METHODS AND APPLICATIONS</td>
<td>21</td>
</tr>
<tr>
<td>3.0 The genetic process in general</td>
<td>21</td>
</tr>
<tr>
<td>3.1 The Exclusive-OR (XOR) problem</td>
<td>25</td>
</tr>
<tr>
<td>3.2 The inverse kinematics problem for robots with multiple degrees of freedom</td>
<td>30</td>
</tr>
<tr>
<td>3.3 The job shop scheduling problem</td>
<td>37</td>
</tr>
<tr>
<td>3.3.1 The genetic process to the job shop scheduling problem</td>
<td>38</td>
</tr>
<tr>
<td>3.3.2 The genetic algorithm against the branch and bound technique</td>
<td>45</td>
</tr>
<tr>
<td>3.3.3 The genetic algorithm against dispatching rules</td>
<td>46</td>
</tr>
<tr>
<td>CHAPTER 4: ANALYSIS OF RESULTS</td>
<td>49</td>
</tr>
<tr>
<td>4.1 The Exclusive-OR (XOR) Problem</td>
<td>49</td>
</tr>
<tr>
<td>4.2 The inverse kinematics problem for robots with multiple degrees of freedom</td>
<td>50</td>
</tr>
<tr>
<td>4.3 The job shop scheduling problem</td>
<td>52</td>
</tr>
<tr>
<td>4.3.1 The genetic process to the job shop scheduling problem</td>
<td>52</td>
</tr>
<tr>
<td>4.3.2 The genetic algorithm against the branch and bound technique</td>
<td>56</td>
</tr>
<tr>
<td>4.3.3 The genetic algorithm against dispatching rules</td>
<td>57</td>
</tr>
<tr>
<td>CHAPTER 5: CONCLUSIONS</td>
<td>64</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>68</td>
</tr>
</tbody>
</table>
APPENDICES:

APPENDIX I: Program Listing
(1) The Exclusive-OR problem............. A- 1
(2) The inverse kinematics for robots
with multiple degrees of freedom.... A-17
(3) The job shop scheduling problem..... A-31

APPENDIX II: Data Description and Listing............. A-53

APPENDIX III: Distributions for All
Performance Measures (25 problems)...... A-59
Abstract

Genetic algorithms (GAs) and their integration with traditional operations have been utilized for solving a variety of optimization problems in engineering. A real-valued code genetic algorithm with stochastic multi-criteria was utilized to solve the decision making of three engineering optimization problems: the Exclusive-OR (XOR) problem, the inverse kinematics problem for robots with multiple degrees of freedom, and the job shop scheduling (JSS) problem. Minimizing RMS error (Root Mean Square Error), arm positioning error with joint angle displacements, and the combination of maximum flow time, mean flow time, maximum tardiness, mean tardiness, work-in process inventory, resource utilization, throughput, and number of tardy jobs were separately employed as the fitness functions in solving these three problems. The results for all three problems showed that real-valued code GAs can clearly represent the corresponding solution set of the problem and accurately find the optimal solutions in a shorter searching time. The genetic algorithm approach for the XOR problem, the inverse kinematics problem for robots with multiple degrees of freedom, and the job shop scheduling problem is proving to be a promising direction for finding near-optimal solutions to engineering optimization problems.
Chapter One

Introduction

In the past three decades, a number of search methods have been used effectively for optimizing the design and operation of engineering systems. However, because today's engineering systems encompass such a broad spectrum, no one optimization method has proven entirely satisfactory by itself. Enumerative search procedures such as dynamic programming enjoy guarantees of globality, but become insufficient when subjected to large problems. Other methods, such as calculus-based gradient search, converge to solutions effectively, but often converge to local optimal solutions when searching noisy functions (Goldberg, 1986b; Karr, 1991). Obviously, finding methods which are capable of overcoming efficiently the problem of local convergence and locate global optima is necessary. One technique which has these characteristics is the genetic algorithm (GA) which has been introduced and examined.

Genetic algorithms (GAs), invented by Dr. John Holland in the 1970s, are a robust search methodology for solving engineering optimization problems and are a discovery heuristic in several machine learning systems. Imitating the process of natural selection and evolution based on Darwin’s theory of survival of the fittest, GAs are related
to "generate and evaluate" search techniques where a candidate is generated and then sent to an evaluator for evaluating (Messa, 1991). GAs use operations such as reproduction, crossover, and mutation found in natural genetics to guide their directions of tour through search spaces (Karr, 1991), and employ an objective function to guide them to find optimal solutions.

GAs, then, are different from many traditional optimization methods in the following ways:

1. GAs use an objective function to guide their search, not derivatives or other knowledge
2. GAs use probabilistic rules to search optimal solutions, not deterministic rules
3. GAs work directly with codings of the parameter set, not the parameters themselves
4. GAs consider many points from a population simultaneously, not a single point

Because GAs search from a population of points simultaneously, there is a reduced chance of converging to local optima. In most traditional search methods, a single point is considered based on some decision rules causing them to converge to local optima in multi-model (many peaked) search spaces. Therefore, GAs produce a more global search by generating entire populations of points randomly,
evaluating each point independently, and combining qualities from existing points to form a new population containing improved points (Goldberg, 1989a; Karr, 1991).

In the past two decades, binary-coded (0's and 1's digit-coded) GAs have been applied to solve the Exclusive-OR (XOR) problem, the inverse kinematics problem for robots with multiple degrees of freedom, and the job shop scheduling problem. However, the binary-coded representation approach is not able to clearly illustrate the corresponding parameter set of the problem it is then asked to solve.

In this research, in place of binary-coded algorithms, real-coded (using real number to represent the parameter set) genetic algorithms using stochastic multi-criteria (combining two or more factors dominating search directions to objective functions) which are different from former methods will be utilized to solve three engineering optimization problems: the Exclusive-OR problem for a neuromorphic structure, the inverse kinematics problem for robots with multiple degrees of freedom, and the job shop scheduling problem. More accurate and practical optimal solutions with shorter search time will be the focus of this research. This will be accomplished by using different encoding methods and different objective functions. The PASCAL language will be utilized for programming.
Chapter Two

Literature Review

During recent years, many search methods have been developed to solve engineering optimization problems. Efforts have been made to modify methods for multi-criteria decision making. Genetic Algorithms (GAs) have been employed as the strategy for solving multi-criteria optimization problems in the 1990's. With the availability of parallel computers, genetic algorithms have been becoming more important.

2.0 Genetic Algorithms

Enumerative optimization methods such as dynamic programming or other techniques such as calculus-based gradient search have been utilized to solve many sophisticated optimization problems in engineering such as the traveling salesman problem (TSP), the job shop scheduling (JSS) problem and the convex or nonconvex optimization problem (Karr, 1991). General enumerative or other optimization methods have become progressively more incapable of being used for optimizing sophisticated engineering systems. To solve the variety of complex engineering optimization problems, genetic algorithms (GAs)
have been used and seem to have additional potential.

Genetic algorithms represent a class of adaptive search techniques that have been intensively studied in the past two decades. Schaffer and Grefenstette (1985) applied GAs to a multi-criteria learning system. The GA successfully perform a learning component between knowledge structures and a multidimensional feedback mechanism. The experimental results of Grefenstette's (1986) research shown that GAs are effective optimization algorithms for multilevel adaptive systems. Grefenstette and Pettey (1986) used two approaches: rule set systems and classifier systems to machine learning based on genetic algorithms. The results indicated that GAs can be applied to solve a navigation task for a simulated robot efficiently. As mentioned in the beginning of this section, the travelling salesman problem (TSP) can be solved by dynamic programming and other traditional operation research methods. However, these methods become insufficient with a large searching space, i.e., 40 cities or more. Nevertheless, Muhlenbein, Schleuter, and Kramer (1988) obtained the best solution of the TSP with 442 cities using genetic algorithms.

In 1989, genetic algorithms to solve NP-complete problems was presented by De Jong and Spears. GAs perform well on the boolean satisfiability problem (SAT). Fogarty (1989a) operated the GA and a rule based system effectively for real-time optimization of combustion on ten simulations
of multiple burner installations. Hajela (1990) used genetic algorithms with a population size of 70 to solve the optimal structural design. The results of this study shown that GAs can find a fully converged optimal design after 6 or 7 generations for locating the area of the optimum. Chang et al. (1990) applied GAs for feature selection and creation in two pattern classifications problems. The results indicated that GAs are able to find the correct solution in both feature selection and feature creation with far fewer evaluations than the number required in a random search procedure. Jenkins (1991) presented a promising result in the optimization of structural design by applying genetic algorithms. Meredith et al. (1991) proposed that GAs represent a potentially efficient and structured technique for improving the performance of a FLC (Fuzzy Logic Controller) system.

Genetic algorithms can be applied effectively for solving many different kinds of optimization problems. A GA was tested using a population size of fifty-two to optimize the parametric design of aircraft by Bramlette and Bouchard (1991). GAs were modified to be different from the general genetic algorithms in this research. The results of this research shown that GAs can find the optimal combinational solution of eight parameters after two iterations with 4680 function evaluations from a search space containing over 136 billion points. Grefenstette (1991) utilized the SAMUEL
system using genetic algorithms to optimize strategies for sequential decision tasks. The results of this study for the Evasive Maneuvers problem presented that GAs can be employed for learning systems to effectively search a space of knowledge structures.

LIEPINS and POTTER (1991) applied genetic algorithms to optimize the multiple-fault diagnosis problem. This research suggested that GAs frequently provide near optimal and even optimal solutions to a variety of multiple-fault diagnosis formulations. Genetic algorithms also are suitable for large-scale chemical optimization problem. LUCASius et al. (1991) indicated that GAs can be applied for the conformation analysis of aqueous DNA molecules effectively.

2.1 Genetic Algorithms and Artificial Neural Networks

In 1987, Smith et al. claimed that genetic algorithms offer a general strategy to calibrating artificial neural systems (ANS). In this study, GAs were utilized to solve the problem of calibrating an ANS that searches optimal paths over a given surface. The results of this research also shown that GAs are able to find a weight vector capable of exactly predicting the optimal path after 73 generations in a searching time within 14 seconds. The speed of
computation and the accuracy of solutions would be difficult to merge into most other network programming approaches. Whitley and Hanson (1989b) pointed out that genetic algorithms can easily encode a neural net to the binary form and provide an objective function, sum-squared error, which is employed in current neural network procedures. Therefore, artificial neural networks would seem to be a suitable application of GAs. In Whitley and Hanson, the GENITOR (GENetic ImplemenTOR), based on genetic algorithms, was applied to the Exclusive-OR (XOR) problem, a 424-encoder, and an adder problem. The results of this experiment shown that the population size of GAs do influence the optimal search and GAs do solve problems that backpropagation can not solve with errors reduced to 0.000000000001 and below.

Harp et al. (1989) implemented genetic algorithms into the GENESYS system to search near-optimal neural network architectures. The results of this application showed that GAs accurately find the best networks. These networks were able to learn in 40 epochs. The training performance was better than a random search algorithm. Montana and Davis (1989) reported excellent results using a genetic algorithm to optimize a neural network for the classification of sonar data. However, instead of the binary encoding, a real-valued encoding approach was employed. The conclusions of their study shown that GAs can be successfully used to train
feedforward networks.

Genetic algorithms were used to evolve neural network architectures for the Exclusive-OR (XOR) Boolean function problem by Miller et al. (1989). GAs employed the binary encoding, the total sum square error (TSSE) as the fitness function, with a population size of 50, a crossover rate of 0.6, and a bitwise mutation rate of 0.005 to solve for the XOR problem. The results were promising. Spears and De Jong (1990) revisited the canonical example of an NP-Complete problem, the boolean satisfiability (SAT) problem, by using neural networks and genetic algorithms. Pitney, Smith, and Greenwood (1990) applied GAs to the design of processing elements (PEs) for recurrently joined optimal path finding networks. Their research implemented ARGOT (Adaptive Representation Genetic Optimizer Technique) incorporated with GAs using Standard Mean Error (SME) as the fitness function to solve this problem. This study found that the GA can increase the search speed and efficiency.

In the past three decades, the Exclusive-OR (XOR) problem has been a topic of research. The XOR is a problem of symbolic logic by which a two-input, one-output system produces a positive result only when the two inputs are of different values. Whitley and Bogart (1990a) utilized a binary-coded GA combined with a backpropagation neural network to solve the XOR problem successfully. In 1990, Austin developed two fitness functions to guide a real-coded
GA to find the optimal solution for the XOR problem. The first was based on a linear piece-wise function mapping the integer value of the output bit pattern. The other performance function was based on the hamming distance of the output pattern from the target pattern. However, the accuracy of optimal solutions and the searching time still required further study. Marshall and Harrison (1991) claimed that the training of feedforward neural networks by backpropagation needs much time-consuming experimentation by the network designer. Therefore, in this study, the binary-coded genetic algorithm with the mean-square-error (MSE) as the fitness function were employed efficiently to optimize the network structures and parameters for the XOR problem.

However, to date, no literature is available in which real-valued genetic algorithms with an objective function, RMS error (Root Mean Square error), have been utilized to solve the XOR problem.

2.2 Genetic Algorithms and Inverse Kinematics

for Robots with Multiple degrees of freedom

Solving the inverse kinematics problem for robots with multiple degrees of freedom has also been an application of genetic algorithms. Parker et al. (1989) successfully utilized a binary-coded GA with multi-criteria to find the optimal robot configuration. The robot's arm positioning
error with an additional constraint based on joint angle displacements from the initial position were employed as the multi-objective fitness function in this application. The simulated robot model was a four-degree-of-freedom redundant robotic arm manipulator. The population size of 50 was used in this research. The results of this research shown that GAs are suitable for off-line programming of a redundant robot in point-to-point positioning tasks. Although the final positioning accuracy is still need improving, the simplicity and ability of GAs makes an interesting contribution to the field of inverse kinematics.

In 1990, Davidor employed genetic algorithms to optimize robot trajectories successfully. This application proved that GAs are capable of processing order dependent, varying in length representations to exactly characterize robot trajectories. The GA approach might also be applied to the obstacle avoidance problem. Khoogar and Parker (1991) utilized a GA with a fitness function, minimizing the total error which is combined by the positioning error, avoid the obstacles, and the number of moves, to use the redundancies to plan no collision motions of the robot from an arbitrary initial point to a desired target point. The results of this application are supportive and encouraging.

However, no information has been found which presents real-valued GAs being used to find the more accurate solutions to the inverse kinematics problem for robots with
multiple degrees of freedom.

2.3 Genetic Algorithms and Job Shop Scheduling

Job Shop Scheduling for processing m jobs on n machines is a combinatorial and a classical operations research problem. There are $(m!)^n$ feasible schedules in this hard problem. An optimal solution depending on a specific measure of performance absolutely exists and can theoretically be found in a countable number of computational iterations. However, a lot of computations will be required to find a satisfactory schedule. For example, there are $(8!)^2 = 1625702400$ feasible solutions for a small scheduling problem such as 8 jobs on 2 machines. It will take an interminable time to search an optimal schedule by direct enumeration. Therefore, employing effective theorems, rules, and algorithms to solve this complex problem are necessary.

In the past two decades, genetic algorithms have been used to approach optimal solutions for the job shop scheduling problem. Davis and Ritter (1987) developed a probabilistic search routine, simulated annealing, to achieve scheduling of students in class sections at Harvard University. The routine itself was optimized by the use of a genetic algorithm with some modifications such as average-crossover and creep-value new operators. A real-coded
parameter set was used in this investigation. In 1989, Cleveland and Smith employed a GA as a method of scheduling the release of jobs into an automated manufacturing facility. A population size of 100, recombination rate of 0.6, five operators such as subtour replace, subtour chunking, subtour swap, partially-mapped crossover (PMX), and weighted chunking, and tardiness penalties as the evaluation function were operated in this experiment. Cleveland et al. also compared the schedule evaluation of GAs with three dispatch rules: earliest due date (EDD), shortest processing time (SPT), and least slack time (LST) and claimed that the GA-based technique is more effective than these three rules.

Hou, Hong, and Ansari (1990) proposed an effective approach based on genetic algorithms to solve the multiprocessor scheduling problem. GAs were employed as an operator to find an optimal schedule for a general task graph to be performed on a multiprocessor system in this investigation. Minimizing the schedule length was the objective function for GAs in this application. This approach with a population size of 20 and the crossover rate of 0.6 was able to converge to a near-optimal solution in 40 iterations. Biegel and Davern (1990) used a binary-coded genetic algorithm with a fitness function, evaluating each schedule's mean flow time, a mutation operator, and partially-mapped crossover to generalize the n-tasks on m-
processors (serial) case in the job shop scheduling (JSS)
problem. Biegel et al. (1990) concluded that the GA
scheduler can be employed as a transitive phase among
production engineering (manufacturing process plan), order
entry (work order demand), and factory floor (shop schedule)
in the Computer Integrated Manufacturing (CIM) system.

Genetic algorithms with a new operator called linear order
crossover (LOX) and partially-mapped crossover (PMX) for
minimizing the tardiness for different capacity job shop
scheduling problems was presented by Falkenauer and
Bouffouix (1991). The results of this experiment shown that
GAs represent a near-optimal and reasonable solution in a
fast searching time over other techniques for this problem.

Kanet and Sridharan (1991) applied a computer search
technique called PROGENITOR based on genetic algorithms with
general mating operators to minimize tardiness for
scheduling 16 jobs onto a single machine. A more realistic
data set which adds different job types with different setup
times was employed in this research. Whitley et al. (1991)
developed a computer optimizer called GENITOR based on
genetic algorithms with a new operator, Enhanced Edge
Recombination, to solve the job scheduling problem and then
obtained a dramatic result. This innovative operator also
was efficiently applied to resolve the Travelling Salesman
Problem (TSP) for 30 and 105 cities (Whitley et al., 1989a).

In the study of local search in problem and heuristic space
for job shop scheduling genetic algorithms (Storer et al., 1992), GAs have been employed to develop job shop scheduling in three experiments: using and minimizing the makespan criterion, minimizing average absolute deviation from due dates, solving the "robust scheduling" for job shops. The results of these experiments were compared with other optimization techniques. The conclusions of this research shown that GAs provide significant advantages for combinatorial optimization problems. Starkweather et al. (1992) applied binary-coded GAs to the Travelling Salesman Problem (TSP) and sequence scheduling problems. Success with this problem has led us to applying similar methods to production scheduling on a sequencing problem. In this investigation, four crossover operators: Enhanced Edge Recombination, Order Crossover, Order crossover (with random crossover points), and partially-Mapped Crossover (PMX) were introduced, surveyed, and compared. Stoppler and Bierwirth (1992) applied a parallel genetic algorithm (PGA) to a special case of the general production problem, the so-called n-job m-machine Permutation flowshop n/m/P/Cmax and also surveyed three crossover operators: Partially-Mapped Crossover (PMX), Order Crossover (OR), and Maximal-Preventive Crossover (MPX). This experiment presented that the OX crossover operator performs best.

In addition, Genetic algorithms can be employed as a optimization procedure to the generic architecture for an
intelligent controller (See Figure 2.1) proposed by Davis et al. (1992). This architecture performs four major functions: assessment, optimization, execution, and monitoring. Within that architecture, there are four phases in the optimization function. The GA plays the third phase for further improvement in the optimization procedure (See Figure 2.2) (Rabelo et al., 1993).
Figure 2.1 Generic Control Architecture (Adopted from Davis et al. (1992). A generic architecture for intelligent control systems, Computer Integrated Manufacturing Systems, 5, 2, 105-113)
Therefore, to view the whole situation, most scheduling research is done for optimizing a performance measure (one criterion). However, to date, in the real world, multiple performance measures (multicriteria) in more complicated scheduling models are definitely required. There are many important criteria related to scheduling decisions. Emmons (1975) employed multicriteria of minimizing mean flow time with the minimum number of tardy jobs and demonstrated a branch-and-bound procedure to solve a n jobs on one machine scheduling problem. Sen and Gupta (1983) proposed a branch-and-bound algorithm with a bicriteria, minimizing a linear
combination of total flow time and maximum Tardiness, to achieve an optimal solution for a \( n \) jobs on one machine scheduling problem. In 1986, four innovative algorithms for three bicriteria problems employing mean flow time, maximum tardiness, and number of tardy jobs and a three-criteria problem including all of these criteria were utilized to optimize the job shop scheduling problem by Nelson, Sarin, and Daniels. The computational results gave the potential to encourage research incorporating multicriteria to the scheduling problem. Fry, Armstrong, and Blackstone (1987) used a heuristic pairwise interchange algorithm with linear programming to minimize the total penalty based on earliness and tardiness penalties in single machine scheduling.

In the last decade, multicriteria decision making has obtained a great deal of attention in production scheduling problems. However, most research in this field requires more robust algorithms, more realistic test data, more accurate solutions, and a shorter searching time. Furthermore, no information has indicated that genetic algorithms with a more practical test data set and a multicriteria objective function, minimizing maximum flow time, mean flow time, maximum tardiness, mean tardiness, work-in process inventory, resource utilization, throughput, and number of tardy jobs, have been used to solve the job shop scheduling problem.
Chapter Three
Methods and Applications

3.0 The GA process in general

A genetic algorithm with three operators: reproduction, crossover, and mutation will be utilized to solve three engineering optimization problems: the Exclusive-OR, the inverse kinematics for robots with multiple degrees of freedom, and the job shop scheduling problem in this research. To illustrate the basic concept of the GA process, a simple and interesting example is drawn as follows:

**Objective:** Matching the secret string [0 0 1 0 1 0 1 0]

<table>
<thead>
<tr>
<th>Four random trial strings</th>
<th>Matching Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>A) 0 1 0 1 0 1</td>
<td>1</td>
</tr>
<tr>
<td>B) 1 1 1 1 0 1</td>
<td>1</td>
</tr>
<tr>
<td>C) 0 1 1 0 1 1</td>
<td>4</td>
</tr>
<tr>
<td>D) 1 0 1 1 0 0</td>
<td>3</td>
</tr>
</tbody>
</table>

Taking two of the top scorers and "Reproduction"

C) 0 1 1 0 1 1
D) 1 0 1 1 0 0
C) 0 1 1 0 1 1
D) 1 0 1 1 0 0
### Mating through "Crossover" (cut each at an arbitrary point)

<table>
<thead>
<tr>
<th>Crossover</th>
<th>Offspring</th>
<th>Matching Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>C) 0 1:1 0 1 1</td>
<td>E) 0 1:1 1 0 0</td>
<td>2</td>
</tr>
<tr>
<td>D) 1 0:1 1 0 0</td>
<td>F) 1 0:1 0 1 1</td>
<td>4</td>
</tr>
<tr>
<td>C) 0 1:1 0:1 1</td>
<td>G) 0 1 1 0:0 0</td>
<td>4</td>
</tr>
<tr>
<td>D) 1 0 1 1:0 0</td>
<td>H) 1 0 1 1:1 1</td>
<td>3</td>
</tr>
</tbody>
</table>

### Taking two of the top scorers - Repeat reproduction and crossover

<table>
<thead>
<tr>
<th>Crossover</th>
<th>Offspring</th>
<th>Matching Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>F) 1:0 1 0 1 1</td>
<td>I) 1:1 1 0 0 0</td>
<td>3</td>
</tr>
<tr>
<td>G) 0:1 1 0 0 0</td>
<td>J) 0:0 1 0 1 1</td>
<td>5</td>
</tr>
<tr>
<td>F) 1 0 1:0 1 1</td>
<td>K) 1 0 1:0 0 0</td>
<td>4</td>
</tr>
<tr>
<td>G) 0 1 1:0 0 0</td>
<td>L) 0 1 1:0 1 1</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Crossover</th>
<th>Offspring</th>
<th>Matching Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>J) 0 0 1 0:1 1</td>
<td>M) 0 0 1 0:0 0</td>
<td>5</td>
</tr>
<tr>
<td>K) 1 0 1 0:0 0</td>
<td>N) 1 0 1 0:1 1</td>
<td>4</td>
</tr>
<tr>
<td>J) 0 0 1 0 1:1</td>
<td>O) 0 0 1 0:1 0</td>
<td>6</td>
</tr>
<tr>
<td>K) 1 0 1 0 0:0</td>
<td>P) 1 0 1 0 0:1</td>
<td>3</td>
</tr>
</tbody>
</table>

The "0" string is the optimal solution matching the secret string [0 0 1 0 1 0].

Therefore, the genetic algorithm process can be described as capable of:

1. generating a population of chromosomes (a set of possible solutions to a problem) randomly
2. evaluating each chromosome (each possible solution) in the population according to a fitness criterion
3. choosing parent chromosomes with higher performance values in evaluation
(4) discarding lower performing chromosomes of the population to make room for the new chromosomes

(5) creating new chromosomes by mating parent chromosomes using mutation and crossover and insert them into the population (population size is fixed throughout the entire process)

(6) evaluating the new chromosomes; and

(7) if the terminating criterion is achieved, stopping and returning the optimal chromosome (the optimal solution); if not, going to (3).

Reproduction gives an increasing number of samples to the observed best solutions. Those possible solutions are made of abstractions to the actual problem. Note that GAs consider entire possible solutions in the search space simultaneously while many other methods work from a single point. Therefore, GAs search for the global optimum while other techniques may find local optimum. Moreover, the objective criteria of GAs are stochastic rather than deterministic.

Crossover occurs by first randomly mating the newly produced solutions. Then, the mated pairs of solutions are crossed over based on a random number between 1 and the string length (SL) minus one [1, (SL-1)]. The new strings are generated by exchanging the values between the strings at the all string positions starting with the random number
plus one. For example,

**Binary encoding**

Assume to mate the following two strings, random number $= 3$,

- $A = [0 \ 1 \ 1 \ 0 \ 0]$
- $B = [1 \ 1 \ 0 \ 1 \ 1]$

Crossover

- $A' = [0 \ 1 \ 1 : 1 \ 1]$
- $B' = [1 \ 1 \ 0 : 0 \ 0]$

**Real-Value encoding**

- $A = [127.42 \ 107.85 \ 166.79 \ 87.09 \ 123.58]$
- $B = [107.89 \ 99.78 \ 154.67 \ 66.90 \ 100.56]$

Crossover

- $A' = [127.42 \ 107.85 \ 166.79 : 66.90 \ 100.56]$
- $B' = [107.89 \ 99.78 \ 154.67 : 87.09 \ 123.58]$

Mutation is the random change with a specific probability of a string position value. In the above binary encoding example, mutation is to change a 0 to a 1 or a 1 to a 0 under a specific probability. In the real encoding, however, for each entry in the string, this operator will, with fixed probability, add to it a random number chosen from the initialization probability distribution.

GAs operations depend simply on random number generation, string copying, and partial string exchanging. In theory, GAs may be applied to any problem and could be ideal for the XOR problem, the inverse kinematics of redundant robots problem, and the job shop scheduling problem.
3.1 The Exclusive-OR (XOR) Problem

The much-studied XOR Boolean Function served as the first application task in this research. A simple neural network will be developed to integrate a genetic algorithm to solve the Exclusive-OR (XOR) problem. Table 3.1 shows the inputs and desired output for each of the four possible cases of input and output. The problem is called the "Exclusive-OR" problem, because the output is on "1" when exactly one of two inputs is on "1" (That is, one or the other, but not both).

Table 3.1 The inputs and outputs for each of the four cases of the XOR problem

<table>
<thead>
<tr>
<th>Case</th>
<th>Inputs</th>
<th></th>
<th></th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Input1</td>
<td>Input2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

Neural networks are algorithms based on concepts inspired by research into the nature of the brain. The basic units of neural networks are shown in Figure 3.1. Figure 3.2 shows a simple neural network designed to solve the XOR problem. There are 2 inputs, 1 bias unit, 2 hidden units, 9 weights, and 1 output. The actual calculations for
The output of the trained network for case 3 ($I_1 = 0$, $I_2 = 1$) are shown in Figure 3.3.
Figure 3.2 A simple neural network designed to solve the XOR problem

<table>
<thead>
<tr>
<th>Hidden Layer #1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>I#1 = (-3.6)(1) + (2.5)(1) + (2.2)(0) = -1.1</td>
<td></td>
</tr>
<tr>
<td>T#1 = ( \frac{1}{1+e^{-(-1.1)}} = 0.2497 )</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hidden Layer #2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>I#2 = (-4.7)(1) + (11.0)(1) + (11.1)(0) = 6.3</td>
<td></td>
</tr>
<tr>
<td>T#2 = ( \frac{1}{1+e^{-6.3}} = 0.9982 )</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Io = (-2.8)(1) + (-11.4)(0.2497) + (8.4)(0.9982) = 2.7383</td>
<td></td>
</tr>
<tr>
<td>To = ( \frac{1}{1+e^{-2.7383}} = 0.9392 )</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.3 Calculations for the output of the trained network shown in Figure 3.2.
There are several terms which must be defined before executing the GA process:

(1) Population size: 4 groups of 9-term weights for 4 networks

(2) Initialization procedure: the weights of the initial members of the population are chosen at random with a uniform distribution between -1.0 and 1.0.

(3) The mutation probability: 0.1

(4) Mutation operation: add to the mutated solution a random value chosen between -0.5 and 0.5

(5) The crossover probability: 0.5

(6) The fitness function: RMS error (Root Mean Square Error). The RMS error has the form of

\[ \sqrt{\frac{\sum_{p=1}^{m} \sum_{j=1}^{J} (T_{ipj} - O_{ipj})^2}{mj}} \]

where

- \( T_{ipj} \) is the desired output of the \( j \)th output node in the \( p \)th pattern
- \( O_{ipj} \) is the real output of the \( j \)th output node in the \( p \)th pattern
- \( m \) is the number of training patterns
- \( j \) is the number of the number of the output nodes in the network
The GA process proceeds as follows:

(1) A population of legal weights is randomly generated
(2) Evaluate each group of weights using the fitness function (RMS Error)
(3) Perform the traditional GA operations of reproduction, crossover, and mutation
(4) Evaluate each new group of weights for it's RMS error
(5) Continue with the GA process steps (3) and (4) above until the RMS error of any group of weights is smaller than 0.05
(6) Stop the process, and present the results to users as a group of weights for the simple neural network.

The GA operators: reproduction, crossover, and mutation are performed using the above process. Each new group of weights is produced by evaluation and the "survival of the fittest" concept. The optimal solution is the group of weights providing the minimum RMS error.

The above GA process will be programmed in the PASCAL language on an IBM-386 compatible personal computer.
3.2 The Inverse Kinematics Problem for Robots with Multiple Degrees of Freedom

A robotic manipulator has an infinite number of configuration for the robot end-effector to reach a given point when it has more degrees of freedom than its workspace does. Therefore, it becomes extensively difficult to determine the optimal robot configuration when the robotic manipulator has more and more degrees of freedom (Ma, 1992).

A number of methods have been employed to obtain a set of solution of robot-joint angles for the robotic manipulator with multiple degrees of freedom on the basis of inverse kinematics. These approaches, such as the Lyapunov function and Jacobian pseudo-inverse, have been incapable of determining the optimal robot-joint angles which follows by the minimum displacement of the robot joints. Thus, here, genetic algorithms was utilized as the optimization mechanism to solve this much-studied problem but still has no unsatisfactory solutions to date.

The MITSUBISHI RM-501 robot was employed as the apparatus in this research. This robot, shown in Figure 3.4, is a five degree-of-freedom arm manipulator.
Since the fifth degree of freedom (Wrist Roll) only influences the orientation of the robot end-effector, therefore, this research dealt with four of the five degrees of freedom. The robot is redundant in three dimensional workspace with these four degrees of freedom. The specifications of this manipulator are given in Table 3.2. This table was adopted from the chapter two (specifications) of the manual of the MITSUBISHI RM-501 robot.
Table 3.2 The specifications of the joint angles

<table>
<thead>
<tr>
<th>Items</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waist rotation (θ1)</td>
<td>300 degrees</td>
</tr>
<tr>
<td>Shoulder rotation (θ2)</td>
<td>130 degrees</td>
</tr>
<tr>
<td>Elbow rotation (θ3)</td>
<td>90 degrees</td>
</tr>
<tr>
<td>Wrist pitch (θ4)</td>
<td>-90 ↔ 90 degrees</td>
</tr>
</tbody>
</table>

Furthermore, in this integration of genetic algorithms, θ1 (the waist rotation angle) was not considered as a parameter in the optimization procedure. The waist rotation (θ1) which from the initial position (called the initial XZ plane) to the target position (called the target XZ plane) was fixed, therefore, in this study, θ1 was not a parameter in the genetic process. This angle can be calculated by the coordinates of target point (X, Y, Z) and the initial configuration (θ1, θ2, θ3, θ4) of robot using simple plane geometry (See Figure 3.5).
Given the target position coordinates: \((X_t, Y_t, Z_t)\) and the initial configuration of robot: \((\theta_1, \theta_2, \theta_3, \theta_4)\) then the near-optimal configuration of robot: \((\theta_1', \theta_2', \theta_3', \theta_4')\)

![Diagram](image)

Figure 3.5 The optimizing mechanism for determining the near-optimal configuration of robot

There are several terms which must be defined before executing the GA process:

1. Population size: 16 groups of legal (within its own range of movements) 3 angles \((\theta_2, \theta_3, \theta_4)\)

2. Initialization procedure: the angles of the initial members of the population are chosen at random with a distribution between \((60^\circ-190^\circ)\) for \(\theta_2\), \((90^\circ-180^\circ)\) for \(\theta_3\), and \((90^\circ-270^\circ)\) for \(\theta_4\)

3. The mutation probability: 0.50
(4) Mutation operation: add to the mutated solution a random value by rules depending on the scale of the arm positioning error

(5) The crossover probability: 0.25

(6) The fitness function:

\[ \text{Fitness} = \text{Position Error} + K \times \text{Displacement(maximum)} \]

where

\[ \text{Position Error} = \sqrt{(X_f - X_t)^2 + (Y_f - Y_t)^2} \]

\[ \text{Displacement} = \max \{ |\theta(2,f) - \theta(2,i)|, |\theta(3,f) - \theta(3,i)|, |\theta(4,f) - \theta(4,i)| \} \]

\[ \text{t: the target point} \]

\[ \text{i: the initial point} \]

\[ \text{f: the feasible solution} \]

\[ K = 0.2 \] (the factor scales the contributions from each term)

In this study, several rules were added in the mutation operation to forced the solutions which have the smaller position error to survive in the population. For example, a solution with the position error of 0.5 will be plus or minus a degree between 0.1° and 1°. A solution with the larger position error will be plus or minus a larger range of degrees. The fitness function is to minimize the arm positioning error with an additional term based on the joint angle displacements from the initial point. Considering the
Joint rotational displacement is the concept of the cost. A arm manipulator has to accurately move to the target position in the shortest time. The rotational displacement can dominate the time and cost. Also, if the fitness function only minimizes the term of the arm positioning error, the optimal solution could be the same position as the initial point. Therefore, in this application, the genetic algorithm will minimize the fitness function by forcing the arm positioning error under 0.005 and the maximum angle displacement to a minimum value.

Then, the GA process can be described as follows:

1. A population of 16-group legal ($\theta_2$, $\theta_3$, $\theta_4$) is randomly generated
2. Use the fitness function to evaluate each group
3. Choose the smallest 4 groups according to the value of fitness
4. Reproduction 4 times to keep the population size of 16 groups legal $\theta$s
5. Mutation (probability = 0.50)  
   * mutation rule --> IF ( $? < $Positioning Error $< ?$ ) then +/- a degree in different ranges
6. Crossover (probability = 0.25)
* A = [166.79°, 87.09°, 123.58°]
B = [154.67°, 66.90°, 100.56°]

Crossover

---------->

A' = [166.79°, 66.90°, 100.56°]
B' = [154.67°, 87.09°, 123.58°]

7. Repeat (2) to (6), until the arm positioning Error less than 0.005

The above GA process will be programmed in the PASCAL language on an IBM-386 compatible personal computer. Ma (1992) did a contribution to the inverse kinematics problem for robots with multiple degrees of freedom. Ma employed two steps of artificial neural network systems for planning robot motions with the same four-degree-of-freedom robotic manipulator as this research's. However, the accuracy and training time of artificial neural network systems still need improving. Therefore, an experiment, integrating genetic algorithms and artificial neural network systems, will be considered to reduce the arm positioning error and speed up the searching time.
3.3 The Job Shop Scheduling Problem

In this section, the application of a parallel genetic algorithm to a special case, 10-job on a single machine, of the general production scheduling problem will be investigated. This determines the order of 10 processing jobs to be performed on a single machine. Concentration is paid to permutation of schedules for this problem. Furthermore, feasible schedules are made by dispatching the jobs with an appropriate priority rule according to some certain performance measures.

Genetic algorithms (GAs) will be programmed to find a near-optimal schedule with some certain performance measures in this huge searching space, 10! (3628800) feasible schedules. Then, genetic algorithms will be compared with the much-studied and famous scheduling approach, the Branch and Bound method, under bicriteria (mean flow time and maximum tardiness). Furthermore, pure GAs and GAs with a scheduling simulator will be organized against 16 dispatching rules. A simulator was built to generate the testing data set. This data set, included current time (time unit is changing during the procedure), job type (different setup times), arrival time, due date, and processing time (a uniform distribution), is more practical to match the real world's production system.
3.3.1 The Genetic Process to Job Shop Scheduling Problem

A number of assumptions about the structure of the job shop scheduling problem in this research have to be made before describing the genetic process. There are several assumptions were made as follows:

1. only one job can be processed on a given machine at one time
2. no operation may be interrupted unless an entirely new schedule is created
3. these 10 jobs can be processed at any time during the scheduling period

The genetic algorithm procedures to the job shop scheduling problem can be characterized as follows:

1. A population of 50-group legal schedules is randomly generated
2. Evaluate each schedule using fitness functions
   ▲ Fitness = K1 * Maximum Flow Time
   K2 * Mean Flow Time
   K3 * Maximum Tardiness
   K4 * Mean Tardiness
   K5 * Work-In-Process Inventory
   K6 * Resource Utilization
Throughput

Number of tardy jobs

where $K_i (i=1,2,\ldots,8) = 1$ (very important)

0.5 (important)

0 (not important)

3. Choose the best 25 groups

4. Reproduction

5. Mutation (with a probability of 0.5 to exchange two arbitrary jobs' position)

6. Crossover

* Partially Mapped Crossover

* Order Crossover #1

* Order Crossover #2

* Enhanced Edge Recombination

7. Repeat (2) to (6), until the smallest value of fitness is equal to the previous iteration's fitness

These four crossover methods are described in the following sections:

**Partially Mapped Crossover (PMX):** This operator was introduced and used to recombine permutations of objects for the Traveling Salesman Problem (TSP) by Goldberg and Lingle in 1985. PMX operator produces the legal solutions by choosing the swapping interval between two crossover points
selected randomly. The offspring will inherit the elements from the interval of one of the parents. Then, it is necessary to detect and fix the illegal situations by mapping and exchanging. For example, consider two permutations of 10 elements:

PMX: Position: 1 2 3 4 5 6 7 8 9 10
A: 9 8 4 5 6 7 1 3 2 10
B: 8 7 1 2 3 10 9 5 4 6

The swapping interval: (4 to 6)

Position: 1 2 3 4 5 6 7 8 9 10
A': 9 8 4 2 3 10 1 3 2 10
B': 8 7 1 5 6 7 9 5 4 6

After mapping and exchanging:

Position: 1 2 3 4 5 6 7 8 9 10
A'': 9 8 4 2 3 10 1 6 5 7
B'': 8 10 1 5 6 7 9 2 4 3

Enhanced Edge Recombination: Whitley et al. (1992) developed and used this crossover operator to recombine the links between two cities from two parents to generate a new
solution to the Traveling Salesman Problem. The adjacency
table plays an important roll in the operation. An example
will be illustrated to describe how to use this operator
which is different from past crossover methods. Consider
these two parent tours: \([a\ b\ c\ d\ e\ f\ g\ h\ i\ j]\) and \([c\ f\ a\ j\ h\ d\ i\ g\ b\ e]\). The edge table is as follows:

<table>
<thead>
<tr>
<th>element</th>
<th>Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b, f, j</td>
</tr>
<tr>
<td>b</td>
<td>a, c, g, e</td>
</tr>
<tr>
<td>c</td>
<td>b, d, e, f</td>
</tr>
<tr>
<td>d</td>
<td>c, e, h, i</td>
</tr>
<tr>
<td>e</td>
<td>d, f, b, c</td>
</tr>
<tr>
<td>f</td>
<td>e, g, c, a</td>
</tr>
<tr>
<td>g</td>
<td>f, h, i, b</td>
</tr>
<tr>
<td>h</td>
<td>g, i, j, d</td>
</tr>
<tr>
<td>i</td>
<td>h, j, d, g</td>
</tr>
<tr>
<td>j</td>
<td>i, a, h</td>
</tr>
</tbody>
</table>

Step 1: Select a initial element randomly. Suppose \(a\) is
the initial element to start this offspring procedure.
Step 2: Since $a$ has been selected, $a$ element should be discarded from the right-hand side area of the edge table.

Step 3: Choose one element from $a$’s links which has the least links to be the next element. In this case, $j$ has two links, $b$ and $f$ both have 3 links. Therefore, $j$ is the next element.

Step 4: Repeat step 2 & 3, remove $j$ element from the right-hand side area of the edge table. $i$ & $h$ element both have 3 links, under this condition, $i$ or $h$ will be selected randomly. Suppose $i$ element is selected in this case.

Then, repeat above procedures, a new offspring will be created as follows: $[a \ j \ i \ h \ d \ c \ f \ e \ b \ g]$

**Order Crossover #1:** This crossover operator was developed by Davis (1985b). The new offspring obtains the exactly same elements between two crossover points selected randomly from one of the parents. This operation is similar to the swapping interval of PMX. But the fixing work of illegal solutions is totally different. The remaining elements are obtained from the other parent orderly. The inherited procedure must begin with the position next to the second crossover point and jump all elements already appear in the new offspring. Here is the example:
Order Crossover #2: This crossover operator, developed by Syswerda (1991), is different from the previous operator. In this operation, several crossover points are chosen at random. These selected elements of one of the parents are forced to cover these selected elements’ position of the other parent orderly. An example is introduced as follows:
Position: 1 2 3 4 5 6 7 8 9 10
A: a b c d e f g h i j
B: c f a j h d i g b e

Crossover points: 3, 4, 7, 9

Position: 1 2 3 4 5 6 7 8 9 10
A: a b c d e f g h i j
    * * * *
B: c f a j h d i g b e
    * * * *

Selected Elements (for B): a, j, i, b
Selected Elements' position (in A): 1, 2, 9, 10

Offspring #1:
Position: 1 2 3 4 5 6 7 8 9 10
A': a j c d e f g h i b

Selected Elements (for A): c, d, g, i
Selected Elements' position (in B): 1, 6, 7, 8

Offspring #2:
Position: 1 2 3 4 5 6 7 8 9 10
B': c f a j h d g i b e
The above GA process will be programmed in the PASCAL language on an IBM-386 compatible personal computer. This program was designed to be user-friendly. Users can determine the importance of each performance measure, the crossover method, and the mutation operation.

3.3.2 The Genetic Algorithm Against
The Branch and Bound Technique

Apart from heuristic approaches, the branch and bound method is probably the solution technique most widely used in scheduling. This approach performs very well in simple scheduling problems with one performance measure but becomes insufficient to subject to large scheduling problems with more criteria.

A classical branch and bound strategy with two criteria (mean flow time and maximum tardiness) will be programmed to schedule the 10-job on a single machine scheduling problem. The so-called classical is that there is no dispatching rules such as SPT (shortest processing time) and EDD (earlier due date) will be employed before executing the process of branch and bound. Therefore, the algorithm of the branch and bound method will be operated to explore every level’s branches of the elimination tree and to find the near-optimal schedule with the lowest bound (the value of the summation of mean flow time and maximum tardiness).
Then, the results of the branch and bound method for the 10-job on a single machine scheduling problem will be compared with the genetic algorithm's.

3.3.3 The genetic Algorithm Against Dispatching Rules

In this section, 16 dispatching rules will be introduced and set up to an investigation to compare with genetic algorithms. Dispatching rules are utilized to select the next job to be machined from a set of queue jobs. These 16 scheduling rules are described as follows (Rabelo et al., 1993; Montazeri et al., 1990):

1. SPT (Shortest Processing Time): Select the job with the shortest process time.
2. LPT (Largest Processing Time): Select the job with the largest processing time.
3. FIFO (First In First Out): Select the job which arrived earliest.
4. LIFO (Last In First out): Select the job which arrived latest.
5. SST (Shortest Setup Time): Select the job with the least setup time.
6. LST (Largest Setup Time): Select the job with the largest setup time.
7. SPST (Shortest Processing and Setup Time): Select the job with the least processing and setup time.
8. LPST (Largest Processing and Setup Time): Select the job with the largest processing and setup time.
9. EDD (Earliest Due Date): Select the job with the nearest due date.
10. LDD (Latest Due Date): Select the job with the latest due date.
11. mSlack (Minimum Slack Time): Select the job with the least amount of slack time (available time before due date for remaining operations).
12. MSLACK (Maximum Slack Time): Select the job with the maximum amount of slack time.
13. CR (Critical Ratio): Select the job with the smallest ratio, calculated at time $t$, as follows:
   \[ CR(i) = \frac{\text{due date}(i) - t}{\text{processing time if job}(i)} \]
14. SSLACK (Shortest Static Slack Time): Select the job with the least amount of static slack time.
15. SLK/TP: Select the job with the smallest ratio of the slack time to the total processing time.
16. SLK/RT: Select the job with the smallest ratio of the slack time to the remaining time before due date.

An experiment will be established to compare the performance of genetic algorithms (GAs) and GAs with two
rules with these 16 dispatching rules under different criteria in 100 problems. "GAs with two rules" means that genetic procedures will previously select the two "best schedules" from these 16 dispatching rules under a certain performance measure into the initial population. A program was written to obtain the results of these 16 dispatching rules using the C programming language.
Chapter Four

Analysis of Results

4.1 The Exclusive-OR Problem

Genetic algorithms with a population size of 4 and RMS Error (Root Mean Square Error) as the fitness function successfully find an optimal group of weights with the RMS error under 0.05 to each node for a simple neural network within one minute searching time (See Figure 4.1).

Figure 4.1 The results of the XOR problem
This program was written in the graphics mode of the PASCAL environment. The training time can be improved by changing to the text mode. But, this is not a very important point here. The results of the XOR problem show that a genetic algorithm can be efficiently utilized to combine with a neural network to solve engineering optimization problems. However, there is a low probability for genetic algorithms to be restrained in the local minimum during the optimization procedure. To improve this situation, a population size of 16 and 32 were used to solve this problem. Then, this situation still exists. This puzzle also exists in the next two problems: the genetic algorithm to the inverse kinematics for redundant robots problem and the job shop scheduling problem.

4.2 The Inverse Kinematics Problem for robots with Multiple Degrees of Freedom

Figure 4.2 shows the results of genetic algorithms (GAs) for solving the inverse kinematics of redundant robots problem. Real-coded GAs with a population size of 16 can find a near-optimal robot configuration with an arm positioning error of 0.004 (inch) and a searching time within one minute in the graphic mode environment. This outcome really gives a promising way for using genetic algorithms to resolve engineering optimization problems.
However, the factor $K_t$ scaling the contributions from each term of the fitness function, is the dominant influence to the solution. Figure 4.2 shows that the ranked 1’s solution has the larger positioning error than ranked 2’s solution’s. Although decision makers can according to the their own practical requirements to select one solution, it is very difficult to find a proper factor $K$ to the fitness function for both minimizing the arm positioning error and maximum displacement to obtain the really suitable optimal answer.

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>135.86</td>
<td>166.59</td>
<td>117.19</td>
<td>0.0516</td>
<td>40.99</td>
<td>8.2295</td>
<td>14</td>
</tr>
<tr>
<td>135.99</td>
<td>166.73</td>
<td>117.73</td>
<td>0.0359</td>
<td>40.99</td>
<td>8.2219</td>
<td>12</td>
</tr>
<tr>
<td>136.04</td>
<td>166.67</td>
<td>117.51</td>
<td>0.0245</td>
<td>40.99</td>
<td>8.2203</td>
<td>10</td>
</tr>
<tr>
<td>136.08</td>
<td>166.49</td>
<td>117.53</td>
<td>0.0138</td>
<td>41.02</td>
<td>8.2198</td>
<td>9</td>
</tr>
<tr>
<td>136.11</td>
<td>166.72</td>
<td>117.70</td>
<td>0.0683</td>
<td>41.05</td>
<td>8.2283</td>
<td>16</td>
</tr>
<tr>
<td>135.93</td>
<td>166.60</td>
<td>117.49</td>
<td>0.0289</td>
<td>40.97</td>
<td>8.2299</td>
<td>3</td>
</tr>
<tr>
<td>136.06</td>
<td>166.56</td>
<td>117.39</td>
<td>0.0004</td>
<td>41.00</td>
<td>8.2004</td>
<td>2</td>
</tr>
<tr>
<td>135.89</td>
<td>166.60</td>
<td>117.41</td>
<td>0.0442</td>
<td>40.93</td>
<td>8.2102</td>
<td>6</td>
</tr>
<tr>
<td>136.11</td>
<td>166.59</td>
<td>117.24</td>
<td>0.0155</td>
<td>41.05</td>
<td>8.2255</td>
<td>13</td>
</tr>
<tr>
<td>135.97</td>
<td>166.60</td>
<td>117.80</td>
<td>0.0336</td>
<td>40.91</td>
<td>8.2155</td>
<td>8</td>
</tr>
<tr>
<td>135.89</td>
<td>166.57</td>
<td>117.51</td>
<td>0.0307</td>
<td>40.97</td>
<td>8.2047</td>
<td>5</td>
</tr>
<tr>
<td>135.99</td>
<td>166.39</td>
<td>117.48</td>
<td>0.0628</td>
<td>40.97</td>
<td>8.2368</td>
<td>15</td>
</tr>
<tr>
<td>135.99</td>
<td>166.74</td>
<td>117.35</td>
<td>0.0095</td>
<td>40.94</td>
<td>8.1975</td>
<td>1</td>
</tr>
<tr>
<td>135.94</td>
<td>166.60</td>
<td>117.66</td>
<td>0.0299</td>
<td>40.88</td>
<td>8.2029</td>
<td>4</td>
</tr>
<tr>
<td>135.89</td>
<td>166.73</td>
<td>117.52</td>
<td>0.0284</td>
<td>40.99</td>
<td>8.2114</td>
<td>7</td>
</tr>
<tr>
<td>136.02</td>
<td>166.49</td>
<td>117.33</td>
<td>0.0289</td>
<td>40.96</td>
<td>8.2209</td>
<td>11</td>
</tr>
</tbody>
</table>

Figure 4.2 The results of the inverse kinematics for redundant robots problem
Furthermore, this genetic algorithm program also was combined with an artificial neural network system to optimize this robot final configuration problem. Genetic algorithms only generated 15 groups of legal 8s, the remaining group will be provided by the results of this artificial neural network system. Since the results of that system are almost the near-optimal configuration, thus, genetic algorithms can find a more accurate solution in a shorter searching time. Averagely, combining with this artificial neural network optimization system can reduce searching iterations within 50%.

4.3 The Job Shop Scheduling Problem

4.3.1 The Genetic Process to Job Shop Scheduling Problem

One group of data set (See Table 4.1) is represented to explain the results of genetic algorithms for solving the job shop scheduling problem. This result was found by focusing the performance measures on mean flow time and maximum tardiness (K2,K3=1; K1,K4,K5,K6,K7,K8=0). Figure 4.3 shows the genetic algorithm can find a near-optimal schedule with the minimum summation value of mean flow time and maximum tardiness in 15 iterations within 5 seconds.
Table 4.1 The represented data set

Current Time = 863  
Previous Job Type executed: 3  
Queue Size = 10

<table>
<thead>
<tr>
<th>Job Number</th>
<th>Job Type</th>
<th>Arrival Time</th>
<th>Processing Time</th>
<th>Due Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>789</td>
<td>Normal(8,0.4)</td>
<td>890</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>805</td>
<td>Normal(8,0.4)</td>
<td>911</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>809</td>
<td>Normal(10,0.6)</td>
<td>910</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>826</td>
<td>Normal(4,0.2)</td>
<td>886</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>830</td>
<td>Normal(6,0.3)</td>
<td>905</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>832</td>
<td>Normal(15,0.75)</td>
<td>1009</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>847</td>
<td>Normal(8,0.4)</td>
<td>956</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>848</td>
<td>Normal(5,0.2)</td>
<td>919</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>855</td>
<td>Normal(4,0.2)</td>
<td>919</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>860</td>
<td>Normal(3,0.1)</td>
<td>920</td>
</tr>
</tbody>
</table>

Table 4.2 and Figure 4.4 show the relationship between mean flow time and maximum tardiness of this data set by evaluating all 10! (3628800) feasible schedules.
Figure 4.3 The results of genetic algorithms for scheduling the 10 jobs on a single machine

Table 4.2 The relationship between mean flow time and maximum tardiness of the represented data set

<table>
<thead>
<tr>
<th>Data Group</th>
<th>XX</th>
<th>Evaluating 10! (3628800) schedules</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Tardiness (maximum)</td>
</tr>
<tr>
<td>Mean Flow Time</td>
<td></td>
<td>2.0&lt;=T&lt;5.7</td>
</tr>
<tr>
<td>69.7&lt;=MFT&lt;71.0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>71.0&lt;=MFT&lt;72.2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>72.2&lt;=MFT&lt;73.5</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>73.5&lt;=MFT&lt;75.0</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>75.0&lt;=MFT&lt;76.0</td>
<td>17</td>
<td>38</td>
</tr>
<tr>
<td>76.0&lt;=MFT&lt;94.9</td>
<td>51</td>
<td>1240</td>
</tr>
</tbody>
</table>
Figure 4.4 The relationship chart of mean flow time and maximum tardiness for the represented data set

There are 33 \((17+16)\) near-optimal solutions in this testing data set. Then, genetic algorithms can accurately find a near-optimal schedule from this huge searching space in a very short searching time (within 5 seconds). The probability to search a near-optimal solution can be easily calculated as \(33/3628800 = 0.0000091\). This is a very promising result for using genetic algorithms to solve the job shop scheduling problem.

From analyzing the results of these three problems, it is obviously to understand that the optimization problem with multiple objective is a surprisingly hard topic in the
field of engineering. However, genetic algorithms actually can overcome these difficulties if the population size, mutation rate, crossover rate, and factor $K$ can be set up appropriately.

### 4.3.2 The Genetic Algorithm Against The Branch and Bound Technique

Table 4.3 shows the results of comparing the performance between the genetic algorithm and the branch and bound method with bicriteria: mean flow time and maximum tardiness in 25 different problems.

<table>
<thead>
<tr>
<th></th>
<th>Genetic Algorithms</th>
<th>Branch and Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lowest frequency</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>* Mean Flow Time</td>
<td>84.28</td>
<td>90.58</td>
</tr>
<tr>
<td>* Max. Tardiness</td>
<td>77.80</td>
<td>86.04</td>
</tr>
</tbody>
</table>

* average total performance.

It is obvious that the genetic algorithm is superior to the branch and bound in scheduling the 10 jobs on a single machine with a bicriteria objective function. The branch and bound technique depends on the lower bound to explore
the necessary branches of the elimination tree. Sometimes this approach will miss the near-optimal schedule by exploring wrong branches. On the contrary, genetic algorithms consider many feasible solutions simultaneously in the whole searching space, therefore, GAs are liable to find the near-optimal solution.

4.3.3 The Genetic Algorithm Against Dispatching Rules

Tables 4.4, 4.5, 4.6, and table 4.7 show the results of comparing the performance of genetic algorithms and 16 dispatching rules with different criteria in 100 different problems. The results show that the performance of pure genetic algorithms or genetic algorithms with the two best rules selected from a larger list of such rules is superior to these dispatching rules in any criterion or bicriteria decision.
Table 4.4 Genetic Algorithms against 16 rules with one criterion
(lowest frequency and average total performance)

<table>
<thead>
<tr>
<th>10 jobs fixed during this simulation — 100 problems</th>
<th>Maximum Flow Time</th>
<th>Mean Flow Time</th>
<th>Maximum Tardiness</th>
<th>Mean Tardiness</th>
<th>WIP Inventory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Criteria</td>
<td>Rules</td>
<td>No.</td>
<td>Mean</td>
<td>No.</td>
<td>Mean</td>
</tr>
<tr>
<td>Rules</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPT</td>
<td>475.4</td>
<td>2</td>
<td>298.6</td>
<td>87</td>
<td>319.6</td>
</tr>
<tr>
<td>LPT</td>
<td>487.9</td>
<td>46</td>
<td>311.6</td>
<td>2</td>
<td>345.2</td>
</tr>
<tr>
<td>FIFO</td>
<td>457.0</td>
<td>53</td>
<td>311.4</td>
<td>2</td>
<td>305.9</td>
</tr>
<tr>
<td>LIFO</td>
<td>564.8</td>
<td>0</td>
<td>304.0</td>
<td>36</td>
<td>410.1</td>
</tr>
<tr>
<td>SST</td>
<td>491.2</td>
<td>23</td>
<td>305.7</td>
<td>20</td>
<td>344.0</td>
</tr>
<tr>
<td>LST</td>
<td>464.3</td>
<td>33</td>
<td>308.2</td>
<td>9</td>
<td>310.5</td>
</tr>
<tr>
<td>SPST</td>
<td>475.3</td>
<td>2</td>
<td>298.6</td>
<td>95</td>
<td>319.5</td>
</tr>
<tr>
<td>LPST</td>
<td>487.9</td>
<td>46</td>
<td>312.0</td>
<td>2</td>
<td>344.8</td>
</tr>
<tr>
<td>EDD</td>
<td>459.3</td>
<td>51</td>
<td>308.5</td>
<td>2</td>
<td>302.3</td>
</tr>
<tr>
<td>LDD</td>
<td>563.4</td>
<td>0</td>
<td>306.9</td>
<td>10</td>
<td>421.0</td>
</tr>
<tr>
<td>mSLACK</td>
<td>459.3</td>
<td>51</td>
<td>309.0</td>
<td>2</td>
<td>302.3</td>
</tr>
<tr>
<td>MSLACK</td>
<td>563.3</td>
<td>0</td>
<td>306.8</td>
<td>12</td>
<td>420.9</td>
</tr>
<tr>
<td>CR</td>
<td>462.3</td>
<td>51</td>
<td>307.6</td>
<td>2</td>
<td>302.4</td>
</tr>
<tr>
<td>SSLACK</td>
<td>511.8</td>
<td>0</td>
<td>300.4</td>
<td>44</td>
<td>334.9</td>
</tr>
<tr>
<td>SLK/TP</td>
<td>462.3</td>
<td>51</td>
<td>307.6</td>
<td>2</td>
<td>302.4</td>
</tr>
<tr>
<td>SLK/RT</td>
<td>511.5</td>
<td>0</td>
<td>306.1</td>
<td>4</td>
<td>354.8</td>
</tr>
<tr>
<td>* GA</td>
<td>456.0</td>
<td>94</td>
<td>298.6</td>
<td>87</td>
<td>302.3</td>
</tr>
<tr>
<td>** GA</td>
<td>456.0</td>
<td>94</td>
<td>298.5</td>
<td>98</td>
<td>302.3</td>
</tr>
</tbody>
</table>

* Pure Genetic Algorithms.
** Genetic Algorithms or Genetic Algorithms with rules.
Table 4.5: Genetic Algorithms against 16 rules with one criterion (lowest frequency and average total performance)

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Resource Util.</th>
<th>Through-put</th>
<th>No.s of Tardy Job</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>No.</td>
<td>Mean</td>
</tr>
<tr>
<td>Rules</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPT</td>
<td>0.942</td>
<td>37</td>
<td>0.078</td>
</tr>
<tr>
<td>LPT</td>
<td>0.943</td>
<td>56</td>
<td>0.078</td>
</tr>
<tr>
<td>FIFO</td>
<td>0.903</td>
<td>25</td>
<td>0.074</td>
</tr>
<tr>
<td>LIFO</td>
<td>0.900</td>
<td>26</td>
<td>0.074</td>
</tr>
<tr>
<td>SST</td>
<td>0.949</td>
<td>91</td>
<td>0.079</td>
</tr>
<tr>
<td>LST</td>
<td>0.888</td>
<td>2</td>
<td>0.073</td>
</tr>
<tr>
<td>SPST</td>
<td>0.943</td>
<td>37</td>
<td>0.078</td>
</tr>
<tr>
<td>LPST</td>
<td>0.933</td>
<td>52</td>
<td>0.077</td>
</tr>
<tr>
<td>EDD</td>
<td>0.906</td>
<td>7</td>
<td>0.075</td>
</tr>
<tr>
<td>LDD</td>
<td>0.910</td>
<td>7</td>
<td>0.075</td>
</tr>
<tr>
<td>mSLACK</td>
<td>0.902</td>
<td>8</td>
<td>0.074</td>
</tr>
<tr>
<td>MSLACK</td>
<td>0.910</td>
<td>9</td>
<td>0.075</td>
</tr>
<tr>
<td>CR</td>
<td>0.907</td>
<td>5</td>
<td>0.075</td>
</tr>
<tr>
<td>SSLACK</td>
<td>0.928</td>
<td>28</td>
<td>0.077</td>
</tr>
<tr>
<td>SLK/TP</td>
<td>0.907</td>
<td>5</td>
<td>0.075</td>
</tr>
<tr>
<td>SLK/RT</td>
<td>0.904</td>
<td>2</td>
<td>0.074</td>
</tr>
<tr>
<td>* GA</td>
<td>0.949</td>
<td>89</td>
<td>0.078</td>
</tr>
<tr>
<td>** GA</td>
<td>0.949</td>
<td>93</td>
<td>0.079</td>
</tr>
</tbody>
</table>

* Pure Genetic Algorithms.
** Genetic Algorithms or Genetic Algorithms with rules.
Table 4.6 Genetic Algorithms against 16 rules with bicriteria (lowest frequency and average total performance)

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Maximum Flow Time</th>
<th>Maximum Tardiness</th>
<th>Mean Flow Time</th>
<th>Mean Tardiness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rules</td>
<td>Mean</td>
<td>No.</td>
<td>Mean</td>
<td>No.</td>
</tr>
<tr>
<td>SPT</td>
<td>475.4</td>
<td>2</td>
<td>319.6</td>
<td>298.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>LPT</td>
<td>487.9</td>
<td>46</td>
<td>345.2</td>
<td>311.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>FIFO</td>
<td>457.0</td>
<td>82</td>
<td>305.9</td>
<td>311.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>LIFO</td>
<td>564.8</td>
<td>0</td>
<td>410.1</td>
<td>304.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>SST</td>
<td>491.2</td>
<td>23</td>
<td>344.0</td>
<td>305.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>LST</td>
<td>464.3</td>
<td>34</td>
<td>310.5</td>
<td>308.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21</td>
</tr>
<tr>
<td>SPST</td>
<td>475.3</td>
<td>3</td>
<td>319.5</td>
<td>298.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>LPST</td>
<td>487.9</td>
<td>46</td>
<td>344.8</td>
<td>312.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>EDD</td>
<td>459.3</td>
<td>89</td>
<td>302.3</td>
<td>308.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>LDD</td>
<td>563.4</td>
<td>0</td>
<td>421.0</td>
<td>306.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>mSLACK</td>
<td>459.3</td>
<td>89</td>
<td>302.3</td>
<td>309.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>MSLACK</td>
<td>563.3</td>
<td>0</td>
<td>420.9</td>
<td>306.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>CR</td>
<td>462.3</td>
<td>72</td>
<td>302.4</td>
<td>307.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>SSLACK</td>
<td>511.8</td>
<td>0</td>
<td>334.9</td>
<td>300.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>SLK/TP</td>
<td>462.3</td>
<td>72</td>
<td>302.4</td>
<td>307.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>SLK/RT</td>
<td>511.5</td>
<td>0</td>
<td>354.8</td>
<td>306.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>* GA</td>
<td>457.8</td>
<td>91</td>
<td>304.2</td>
<td>301.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>75</td>
</tr>
<tr>
<td>** GA</td>
<td>458.3</td>
<td>92</td>
<td>302.7</td>
<td>301.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>75</td>
</tr>
</tbody>
</table>

* Pure Genetic Algorithms.
** Genetic Algorithms or Genetic Algorithms with rules.
Table 4.7 Genetic Algorithms against 16 rules with bicriteria (lowest frequency and average total performance)

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Mean Flow Time Mean and No. of Tardy jobs</th>
<th>Maximum and No. of Tardiness Mean</th>
<th>No. of Tardy jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rules</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPT</td>
<td>298.6 25 7.1</td>
<td>319.6 3</td>
<td>7.1</td>
</tr>
<tr>
<td>LPT</td>
<td>311.6 0 8.5</td>
<td>345.2 1</td>
<td>8.5</td>
</tr>
<tr>
<td>FIFO</td>
<td>311.4 0 8.6</td>
<td>305.9 0</td>
<td>8.6</td>
</tr>
<tr>
<td>LIFO</td>
<td>304.0 25 6.7</td>
<td>410.1 0</td>
<td>6.7</td>
</tr>
<tr>
<td>SST</td>
<td>305.7 3 7.7</td>
<td>344.0 2</td>
<td>7.7</td>
</tr>
<tr>
<td>LST</td>
<td>308.2 1 8.0</td>
<td>310.5 13</td>
<td>8.0</td>
</tr>
<tr>
<td>SPST</td>
<td>298.6 33 7.1</td>
<td>319.5 3</td>
<td>7.1</td>
</tr>
<tr>
<td>LPST</td>
<td>312.0 0 8.6</td>
<td>344.8 1</td>
<td>8.6</td>
</tr>
<tr>
<td>EDD</td>
<td>308.5 0 8.3</td>
<td>302.3 14</td>
<td>8.3</td>
</tr>
<tr>
<td>LDD</td>
<td>306.9 8 6.8</td>
<td>421.0 0</td>
<td>6.8</td>
</tr>
<tr>
<td>mSLACK</td>
<td>309.0 0 8.4</td>
<td>302.3 13</td>
<td>8.4</td>
</tr>
<tr>
<td>MSLACK</td>
<td>306.8 9 6.8</td>
<td>420.9 0</td>
<td>6.8</td>
</tr>
<tr>
<td>CR</td>
<td>307.6 0 8.4</td>
<td>302.4 7</td>
<td>8.4</td>
</tr>
<tr>
<td>SSLACK</td>
<td>300.4 21 6.8</td>
<td>334.9 3</td>
<td>6.8</td>
</tr>
<tr>
<td>SLK/TP</td>
<td>307.6 0 8.4</td>
<td>302.4 7</td>
<td>8.4</td>
</tr>
<tr>
<td>SLK/RT</td>
<td>306.1 6 5.9</td>
<td>354.8 4</td>
<td>5.9</td>
</tr>
<tr>
<td>* GA</td>
<td>298.6 85 6.2</td>
<td>303.2 76</td>
<td>6.7</td>
</tr>
<tr>
<td>** GA</td>
<td>298.6 85 6.2</td>
<td>302.3 76</td>
<td>6.8</td>
</tr>
</tbody>
</table>

* Pure Genetic Algorithms.
** Genetic Algorithms or Genetic Algorithms with rules.
Furthermore, Table 4.8 shows the results of an experiment for comparing the searching iterations of pure genetic algorithms and genetic algorithms with the two best dispatching rules selected from a larger list of such rules under certain criteria. Averagely, the iterations will be decreased by adding the two best rules within 40%. These encouraging results and the outcomes shown in the section 4.2 (integrated genetic algorithms to an artificial neural
network system) indeed give a promising direction to implement genetic algorithms to be the compromise analyst role (phase 3) in the optimization function of the generic control architecture described in the chapter 2.3.
Chapter Five

Conclusions

In this research, genetic algorithms were implemented to solve three engineering optimization problems: the Exclusive-OR (XOR) problem, the inverse kinematics for redundant robots problem, and the job shop scheduling (JSS) problem. The results for all these three problems showed that real-coded genetic algorithms can clearly represent the corresponding solution set of the problem and accurately find the near-optimal solutions in a shorter searching time. This study is proving that genetic algorithms are a powerful and robust technique for solving engineering optimization problems.

Furthermore, this research also provides an encouraging result for integrating genetic algorithms with other techniques to solve engineering optimization problems. For example, in the first part of this research, genetic algorithms can be successfully utilized to the problem of training a feedforward network. The scaling problem in backpropagation (a gradient decent procedure) does not exist in genetic algorithms. One reason for this is that GAs improve the current best candidate monotonically and keep the current best individual as part of their population while they search for better candidates. Secondly, the mutation and crossover operators can make GAs keep far away
from local minima.

Second, in the section of the inverse kinematics problem for robots with multiple degrees of freedom, genetic algorithms can be integrated with artificial neural networks to optimize the robot final configuration. The near-optimal solution of this neural network system was selected as a group of the initial population of GAs in this integration. By way of this combination, GAs were able to find a more near-optimal robot configuration with a more accurate arm positioning error in a shorter searching time.

Moreover, in the section of the job shop scheduling problem, genetic algorithms performed very well in scheduling 10 jobs on a single machine. The performance of GAs is superior to other scheduling techniques such as the branch and bound method and 16 dispatching rules. When genetic algorithms were combined with two of the best rules selected from 16 dispatching rules, the more near-optimal schedules can be obtained in fewer searching iterations. These promising results are proving that genetic algorithms can be integrated with other techniques for improvement in engineering optimization problems.

However, the genetic algorithm itself actually is a multiple objective optimization problem. Users have to organize the appropriate combination of the population size, the initialization distribution, the mutation rate, and the crossover rate, then, the performance might be increased.
Furthermore, the factor (K) for scaling the contributions from each term of the multiple objective fitness function are the dominate influence to the optimal solution. To date, no information shows that what exactly the population should be, or the mutation rate should be, etc., and even how to set the proper Ks for the multiple objective optimization. According to the related former researches, experiences, and trial and error is the way to overcome these difficulties.

This research has implemented genetic algorithms with the appropriate combination of genetic operation factors and Ks for solving engineering optimization problems very well. However, these combinations are focused on these three problems, it could be insufficient and inadequate to subject to other optimization problems. Khoogar and Parker (1991) noted that the searching time is largely dependent on the population-size and the mutation rate in the research of solving the obstacle avoidance of redundant manipulators. Goldberg (1989b) suggested that populations ranging in size from 30 to 200 are typically selected and implemented for some fixed number of fitness functions. Goldberg also concluded that relatively small populations are fit for serial implementations and large populations are appropriate for parallel genetic algorithms.

Fogarty (1989b), in the study of varying the probability of mutation in the genetic algorithm, indicated
that varying mutation over generations significantly
improves the performance of the GA on this experiment. Sen
and Gupta (1983) recommended that Ks in the fitness function
could have the linear relationship. A formulation for the
objective function for minimizing the total flow time and
maximum tardiness to a scheduling problem in this research
is described as: Minimize $Z = pF(S) + qt(S)$, where $p+q=1$, and
$p,q \geq 0$. The results of this study shown that $p$ and $q$ with
the linear relationship sometimes are capable of finding a
near-optimal solution but sometimes can not.

There are many studies focusing on solving these
difficulties to find appropriate genetic operation factors
and Ks. These approaches provide the directions for further
research in the field of integrating genetic algorithms to
solve engineering optimization problems.
References


Appendices
Appendix I: 
Program Listing

(1) The Exclusive-OR problem

(2) The Inverse Kinematics for Redundant Robots

(3) The Job Shop Scheduling problem
program XORDEMO; {This program solves the XOR problem}

Uses Crt,Graph;
type msg_type=array[1..4] of string[16];
  weiarr_type=array[1..32,1..9] of string[5];
  weight_type=array[1..32,1..9] of real;
  rms_type=array[1..32] of string[9];
  rms_error_type=array[1..32] of real;
  p_fit_type=array[1..32] of integer;
  math_type=array[1..32,1..4] of real;

var graphdriver,graphmode:integer;
  x,y,i:integer;
  exit_flag,key_press:boolean;
  ran_num:weiarr_type;
  rms:rams_type;
  weight:weight_type;
  p_fit:p_fit_type;
  p_temp:integer;
  temp_fit:rms_error_type;
  t_fit:real;

procedure main_menu(var exit_flag:boolean);
  var ch:char; {Draw the main menu}
  begin
    detectgraph(graphdriver,graphmode);
    initgraph(graphdriver,graphmode,"’");
    setvisualpage(0); setactivepage(0);
    setbkcolor(lightblue); setviewport(0,0,640,450,true);
    clearviewport;
    setcolor(white);
    x:=getmaxx; y:=getmaxy;
    line(round(x/3),20,10,20);
    line(10,20,10,y-20);
    line(10,y-20,x-10,y-20);
    line(x-10,y-20,x-10,20);
    line(x-10,20,2*round(x/3),20);
    line(round(x/3),24,15,24);
    line(15,24,15,y-24);
    line(15,y-24,x-15,y-24);
    line(x-15,y-24,x-15,24);
    line(x-15,24,2*round(x/3),24);
    settextrstyle(1,0,2);
    outtextxy(round(x/3)+4,12,’Exclusive - OR (XOR)’);
    settextrstyle(1,0,1);
    outtextxy(140,40,’--- SPRING 1992, JAY-SHINN TSAI ---’);
    settextrstyle(3,0,1);
    outtextxy(40,80,’Welcome! This program is to demonstrate a basic artificial’);
    outtextxy(140,100,’neural network using GENETIC ALGORITHMS included’);
    outtextxy(140,120,’BIASED-MUTATE WEIGHTS and CROSSOVER WEIGHTS’);
    outtextxy(140,140,’to solve the EXCLUSIVE-OR (XOR) problem.’);
    outtextxy(140,160,’There are 2 inputs, 2 hidden units and 1 output’);
    outtextxy(140,180,’in this network. The following table
is the sample');
outtextxy(140,200,'data.'); settextstyle(2,0,6);
outtextxy(200,240,'Inputs
Output');
line(250,260,465,260); outtextxy(270,265,'0');
outtextxy(330,265,'0'); outtextxy(420,265,'0');
outtextxy(270,285,'1'); outtextxy(332,285,'1');
outtextxy(422,285,'1'); outtextxy(272,305,'1');
outtextxy(330,305,'0'); outtextxy(422,305,'1');
outtextxy(272,325,'1'); outtextxy(332,325,'1');
outtextxy(420,325,'0'); line(370,240,370,343);
settextstyle(3,0,1);
outtextxy(80,370,'Press
ENTER to start to create a set of
random number');
outtextxy(80,390,'for initial weights or ESC to exit !');
sound(550); delay(50); nosound; setlinestyle(0,0,3);
key_press:=false;
while not key_press do
begin
  for i:=1 to 2 do
  begin
    setcolor(white); line(380,410,388,410);
delay(150); setcolor(lightblue);
    line(380,410,388,410); delay(150);
  end;
  if keypressed then ch:=readkey;
  if ((ch=#13) or (ch=#27)) then key_press:=true;
end;
setlinestyle(0,0,1);
if ch=#27 then exit_flag:=true;
end;

procedure win_1(text_type:integer; mess_arr:msg_type);
{window_1 tells user what
 step we are doing now}
var i:integer;
begin
  setviewport(242,198,392,278,true); clearviewport;
  if text_type=1 then
  begin
    settextstyle(1,0,4);
    outtextxy(25-(length(mess_arr[1]))*2,20,mess_arr[1])
  end;
  if text_type=0 then
  begin
    settextstyle(2,0,6);
    for i:=1 to 4 do
      outtextxy(3,8+(i-1)*17,mess_arr[i]);
  end;
  settextstyle(0,0,1);
end;
procedure gra_init;  
{Draw 4 networks}

var msg:msg_type;
begin
  setcolor(green);
  for i:=0 to 500 do
    line(0,i,640,i); setcolor(lightblue);
  for i:=0 to 500 do
    line(0,i,640,i); setcolor(white);
  setbckcolor(lightblue); setfillstyle(0,0);
  setlinestyle(0,3); rectangle(0,0,639,479);
  line(0,239,639,239); line(318,0,318,479);
  setlinestyle(0,1);
  circle(30,80,15); circle(30,140,15);
  rectangle(60,5,85,25); line(140,65,140,95);
  line(140,65,170,80); line(140,95,170,80);
  line(140,125,140,155); line(140,125,170,140);
  line(140,155,170,140); circle(285,110,15);
  line(45,80,140,80); line(45,80,140,140);
  line(45,140,140,80); line(45,140,140,140);
  line(170,80,270,110); line(170,140,270,110);
  line(70,25,140,140); line(85,25,140,80);
  line(85,15,270,110); settextstyle(2,0,3);
  outtextxy(18,77,'Input 1'); outtextxy(18,137,'Input 2');
  outtextxy(142,77,'Hidden'); outtextxy(142,137,'Hidden');
  outtextxy(274,107,'Output'); outtextxy(64,8,'Bias');
  outtextxy(64,15,'= 1'); rectangle(20,200,220,230);
  settextstyle(1,0,1); outtextxy(21,204,'RMS ERROR=');
  circle(30+318,80,15); rectangle(60+318,5,403,25);
  line(140+318,65,140+318,95); line(140+318,65,170+318,80);
  line(140+318,95,170+318,80); line(140+318,125,140+318,155);
  line(140+318,125,170+318,140);
  line(140+318,155,170+318,140);
  circle(285+318,110,15); line(45+318,80,140+318,80);
  line(45+318,80,140+318,140); line(45+318,140,140+318,140);
  line(45+318,140,140+318,140);
  line(170+318,80,270+318,110);
  line(170+318,140,270+318,110);
  line(70+318,25,140+318,140); line(85+318,25,140+318,80);
  line(85+318,15,270+318,110); settextstyle(2,0,3);
  outtextxy(18+318,77,'Input 1');
  outtextxy(18+318,137,'Input 2');
  outtextxy(142+318,77,'Hidden');
  outtextxy(142+318,137,'Hidden');
  outtextxy(274+318,107,'Output');
  outtextxy(64+318,8,'Bias'); outtextxy(64+318,15,'= 1');
  rectangle(20+318+72,200,220+318+72,230);
  settextstyle(1,0,1);
  outtextxy(21+318+72,204,'RMS ERROR=');  
{network3}
circle(30, 80+239+80, 15); circle(30, 140+239+80, 15);
rectangle(60, 5+239+80, 85, 25+239+80);
line(140, 65+239+80, 140, 95+239+80);
line(140, 65+239+80, 170, 80+239+80);
line(140, 95+239+80, 170, 80+239+80);
line(140, 125+239+80, 140, 155+239+80);
line(140, 125+239+80, 170, 140+239+80);
line(140, 155+239+80, 170, 140+239+80);
circle(285, 110+239+80, 15);
line(45, 80+239+80, 140, 80+239+80);
line(45, 80+239+80, 140, 140+239+80);
line(45, 140+239+80, 140, 80+239+80);
line(45, 140+239+80, 140, 140+239+80);
line(170, 80+239+80, 270, 110+239+80);
line(170, 140+239+80, 270, 110+239+80);
line(70, 25+239+80, 140, 140+239+80);
line(85, 25+239+80, 140, 80+239+80);
line(85, 15+239+80, 270, 110+239+80);
settextstyle(2, 0, 3);
outtextxy(18, 77+239+80, 'Input 1');
outtextxy(18, 137+239+80, 'Input 2');
outtextxy(142, 77+239+80, 'Hidden');
outtextxy(142, 137+239+80, 'Hidden');
outtextxy(274, 107+239+80, 'Output');
outtextxy(64, 8+239+80, 'Bias');
outtextxy(64, 15+239+80, '=' 1');
rectangle(20, 200+49, 220, 230+49);
settextstyle(1, 0, 1);
outtextxy(21, 204+49, 'RMS ERROR');
{network4}
outtextxy(142+318,77+239+80,'Hidden');
outtextxy(142+318,137+239+80,'Hidden');
outtextxy(274+318,107+239+80,'Output');
outtextxy(64+318,8+239+80,'Bias');
outtextxy(64+318,15+239+80,'= 1');
rectangle(20+318+72,200+49,220+318+72,230+49);
settextstyle(1,0,1);
outtextxy(21+318+72,204+49,'RMS ERROR=');

setviewport(240,197,394,279,true);
clearviewport; rectangle(0,0,154,82);
settextstyle(1,0,4);
msg[1]:=Welcome!;
win_1(1,msg); sound(550); delay(50); nosound;
delay(1500); setcolor(lightblue);
win_1(1,msg);
setcolor(white);
settextstyle(0,0,1);
end;

procedure win_11(ran:weiarr_type);
{window_11 puts weight11, weight12, weight13, weight14}
var i:integer;
beginn
setviewport(54,76,83,142,true);
clearviewport; settextstyle(2,0,4);
for i:=1 to 4 do
outtextxy(2,(i-1)*19,ran_num[1,i]);
end;

procedure win_12(ran:weiarr_type);
{window_12 puts weight15, weight16}
var i:integer;
beginn
setviewport(175,77,204,142,true);
clearviewport; settextstyle(2,0,4);
for i:=1 to 2 do
outtextxy(2,(i-1)*50,ran_num[1,i+4]);
end;

procedure win_13(ran:weiarr_type);
{window_13 puts weight17, weight18, weight19}
var i:integer;
beginn
setviewport(62,40,158,52,true);
clearviewport; settextstyle(2,0,4);
for i:=1 to 3 do
outtextxy((i-1)*32+2,0,ran_num[1,i+6]);
end;
procedure win_14(ran:weiarr_type);
{window_14 puts chromosome weight[1,j]}
var i:integer;
begin
    setviewport(10,170,308,185,true);
clearviewport; setcolor(white);
setbkcolor(lightblue); rectangle(0,0,298,15);
for i:=1 to 9 do
    line(((i-1)*33+33,0, (i-1)*33+33,15);
settextstyle(2,0,4);
for i:=1 to 9 do
    outtextxy(((i-1)*33+4,2,ran\_[\_num[1,i])]
end;

procedure win_15(rms:rms_type);
{window_15 puts rms_error1}
begin
    setviewport(140,204,216,226,true);
clearviewport; setcolor(white);
setbkcolor(lightblue); settextstyle(0,0,1);
outtextxy(2,7,rms[1]);
end;

procedure win_21(ran:weiarr_type);
{window_21 puts weight21, weight22, weight23, weight24}
var i:integer;
begin
    setviewport(54+318,76,83+318,142,true);
clearviewport; settextstyle(2,0,4);
for i:=1 to 4 do
    outtextxy(2,(i-1)*19,ran\_[\_num[2,i])]
end;

procedure win_22(ran:weiarr_type);
{window_22 puts weight25, weight26}
var i:integer;
begin
    setviewport(175+318,77,204+318,142,true);
clearviewport; settextstyle(2,0,4);
for i:=1 to 2 do
    outtextxy(2,(i-1)*50,ran\_[\_num[2,i+4])]
end;
procedure win_23(ran:weiarr_type);
{window_23 puts weight27, weight28, weight29}

var i:integer;
begin
  setviewport(62+318,40,158+318,52,true);
  clearviewport; settextstyle(2,0,4);
  for i:=1 to 3 do
    outtextxy((i-1)*32+2,0,ran_num[2,i+6]);
end;

procedure win_24(ran:weiarr_type);
{window_24 puts chromosome2 weight[2,j]}

var i:integer;
begin
  setviewport(10+318,170,308+318,185,true);
  clearviewport;
  setcolor(white);
  setbkcolor(lightblue);
  rectangle(0,0,298,15);
  for i:=1 to 9 do
    line((i-1)*33+33,0,(i-1)*33+33,15);
  settextstyle(2,0,4);
  for i:=1 to 9 do
    outtextxy((i-1)*33+4,2,ran_num[2,i]);
end;

procedure win_25(rms:rms_type);
{window_25 puts rms_error2}

begin
  setviewport(140+318+72,204,216+318+72,226,true);
  clearviewport;
  setcolor(white);
  setbkcolor(lightblue);
  settextstyle(0,0,1);
  outtextxy(2,7,rms[2]);
end;

procedure win_31(ran:weiarr_type);
{window_31 puts weight31, weight32, weight33, weight34}

var i:integer;
begin
  setviewport(54,76+239+80,83,142+239+80,true);
  clearviewport; settextstyle(2,0,4);
  for i:=1 to 4 do
    outtextxy(2,(i-1)*19,ran_num[3,i]);
end;

procedure win_32(ran:weiarr_type);
{window_32 puts weight35, weight36, weight36}

var i:integer;
begin
setviewport(175,77+239+80,204,142+239+80, true);
clearviewport; settextstyle(2,0,4);
for i:=1 to 2 do
outtextxy(2,(i-1)*50,ran_num[3,i+4]);
end;

procedure win_33(ran:weiarr_type);
{window_33 puts weight37, weight38, weight39}
var i:integer;
begin
setviewport(62,40+239+80,158,52+239+80, true);
clearviewport; settextstyle(2,0,4);
for i:=1 to 3 do
outtextxy((i-1)*32+2,0,ran_num[3,i+6]);
end;

procedure win_34(ran:weiarr_type);
{window_34 puts chromosome3 weight[3,j]}
var i:integer;
begin
setviewport(10,170+123,308,185+123, true);
clearviewport; setcolor(white);
setbkcolor(lightblue); rectangle(0,0,298,15);
for i:=1 to 9 do
line((i-1)*33+33,0,(i-1)*33+33,15);
settextstyle(2,0,4);
for i:=1 to 9 do
outtextxy((i-1)*33+4,2,ran_num[3,i]);
end;

procedure win_35(rms:rms_type);
{window_35 puts rms_error3}
begin
setviewport(140,204+49,216,226+49, true);
clearviewport; setcolor(white);
setbkcolor(lightblue); settextstyle(0,0,1);
outtextxy(2,7,rms[3]);
end;

procedure win_41(ran:weiarr_type);
{window_41 puts weight41, weight42, weight43, weight44}
var i:integer;
begin
setviewport(54+318,76+239+80,83+318,142+239+80, true);
clearviewport; settextstyle(2,0,4);
for i:=1 to 4 do
outtextxy(2,(i-1)*19,ran_num[4,i]);
end;
procedure win_42(ran:weiarr_type);
{window_42 puts weight45, weight46}
var i:integer;
begin
  setviewport(175+318,77+239+80,204+318,142+239+80,true);
  clearviewport; settextrstyle(2,0,4);
  for i:=1 to 2 do
    outtextxy(2,(i-1)*50,ran_num[4,i+4]);
end;

procedure win_43(ran:weiarr_type);
{window_43 puts weight47, weight48, weight49}
var i:integer;
begin
  setviewport(62+318,40+239+80,158+318,52+239+80,true);
  clearviewport;
  settextrstyle(2,0,4);
  for i:=1 to 3 do
    outtextxy((i-1)*32+2,0,ran_num[4,i+6]);
end;

procedure win_44(ran:weiarr_type);
{window_44 puts chromosome4 weight[4,j]}
var i:integer;
begin
  setviewport(10+318,170+123,308+318,185+123,true);
  clearviewport;
  settextrstyle(2,0,4);
  setcolor(white);
  setbkcolor(lightblue);
  rectangle(0,0,298,15);
  for i:=1 to 9 do
    line((i-1)*33+33,0,(i-1)*33+33,15);
  settextrstyle(2,0,4);
  for i:=1 to 9 do
    outtextxy((i-1)*33+4,2,ran_num[4,i]);
end;

procedure win_45(rms:rms_type);
{window_45 puts rme_error4}
begin
  setviewport(140+318+72,204+49,216+318+72,226+49,true);
  clearviewport;
  settextrstyle(0,0,1);
  outtextxy(2,7,rms[4]);
end;

procedure initial; {initialization for variables}
var i,j:integer;
begin
  for i:=1 to 32 do
    begin
for j:=1 to 9 do
weight[i,j]:=0;
end;
end;

procedure set_random(var weight:weight_type);
{creates initial weights}
var msg:msg_type;
i,j,k,m,c:integer;
begin
msg[1]:='Creating initial';
msg[2]:='weights!';
msg[3]:='Please wait!';
msg[4]:='';
sound(550); delay(50); nosound;
win_1(0,msg);
setlinestyle(0,0,3);
for i:=1 to 10 do
begin
setcolor(white); line(125,60,133,60);
delay(150); setcolor(lightblue);
line(125,60,133,60); delay(150);
end;
setlinestyle(0,0,1); setcolor(white);
for i:=1 to 32 do
begin
for j:=1 to 9 do
begin
k:=random(101);
m:=random(2);
if m=0 then m:=m-1;
weight[i,j]:=k*m*0.01;
str(weight[i,j]:3:1,ran_num[i,j]);
end;
end;
for i:=1 to 32 do
begin
for j:=1 to 9 do
begin
val(ran_num[i,j],weight[i,j],c);
if weight[i,j]>=0 then ran_num[i,j]:='+'
end;
end;
end;
{send random value to each window}
win_11(ran_num); win_12(ran_num);
win_13(ran_num); win_14(ran_num);
win_21(ran_num); win_22(ran_num);
win_23(ran_num); win_24(ran_num);
win_31(ran_num); win_32(ran_num);
win_33(ran_num); win_34(ran_num);
win_41(ran_num); win_42(ran_num);
win_43(ran_num); win_44(ran_num);

msg[1]:=‘Completed !!!’;
msg[2]:=‘press any key’;
msg[3]:=‘to start’;
msg[4]:=‘training !!!’;
win_1(0,msg); sound(550);
delay(50); nosound; setlinestyle(0,0,3);
repeat
begin
    setcolor(white); line(100,75,108,75);
delay(150); setcolor(lightblue);
    line(100,75,108,75); delay(150);
end
until keypressed;
setlinestyle(0,0,1); setcolor(white);
end;

procedure multiplication(var weight: weight_type);
{multiplication operator}

var i,j:integer;
    junk: weight_type;

begin
    for i:=1 to 8 do
    begin
        for j:=1 to 9 do
        begin
            junk[i,j]:=weight[p_fit[i],j];
            junk[i+8,j]:=weight[p_fit[i],j];
            junk[i+16,j]:=weight[p_fit[i],j];
            junk[i+24,j]:=weight[p_fit[i],j];
        end;
    end;
    for i:=1 to 32 do
    begin
        for j:=1 to 9 do
        begin
            weight[i,j]:=junk[i,j];
        end;
    end;

procedure mutation(var weight: weight_type);
{Biased Mutation Weights operator}

var i,j,s,m,n:integer;
t:real;
begin
    for i:=3 to 32 do
begin
  for j:=1 to 9 do
  begin
    s:=random(5);          {probability=0.5}
    if s=0 then
      begin
        m:=random(51); n:=random(2);
        if n=0 then n:=n-1;
        t:=m*n*0.01;
        weight[i,j]:=weight[i,j]+t;
      end;
  end;
end;

procedure crossover(var weight:weight_type);
  {Crossover Weights operator}
var i,j,w,v:integer;
  temp_x:weight_type;
begin
  for i:=1 to 32 do
  begin
    for j:=1 to 9 do
      temp_x[i,j]:=weight[i,j];
  end;
  for i:=2 to 16 do
  begin
    for j:=1 to 9 do
      begin
        w:=random(3);
        if w=0 then
          begin
            weight[2*i-1,j]:=weight[2*i,j];
            weight[2*i,j]:=temp_x[2*i-1,j];
          end;
      end;
  end;
end;

procedure train_wei(weight:weight_type);
  {training operator}
var rms_error:rms_error_type;
  sam_data:array[1..4,1..2] of real;
  target:array[1..4] of real;
  output,out:array[1..32,1..4] of real;
  msg:msg_type;
  hidden1,hidden2,net1,net2:array[1..32,1..4] of real;
data: text;
i, j, c: integer;
ch: char;
start, train, jump_out: boolean;

begin{train_wei}
for i:=1 to 4 do msg[i] := ‘’;
msg[1] := 'Training!';
win_1(1, msg);
assign(data, 'sample.dat');
reset(data);
for i:=1 to 4 do
begin
  for j:=1 to 2 do
  begin
    read(data, sam_data[i, j]);
    end;
    readln(data, target[i]);
  end;
end;
close(data);
start := true;
while start do do
begin{start}
for i:=1 to 32 do
begin
  for j:=1 to 4 do
  begin
    net1[i, j] := weight[i, 8] + sam_data[j, 1] * weight[i, 1]
      + sam_data[j, 2] * weight[i, 3];
    net2[i, j] := weight[i, 7] + sam_data[j, 1] * weight[i, 2]
      + sam_data[j, 2] * weight[i, 4];
    end;
  end;
for i:=1 to 32 do
begin
  for j:=1 to 4 do
  begin
    hidden1[i, j] := 1 / (1 + exp(-net1[i, j]));
    hidden2[i, j] := 1 / (1 + exp(-net2[i, j]));
    end;
  end;
for i:=1 to 32 do
begin
  for j:=1 to 4 do
  begin
      + hidden2[i, j] * weight[i, 6];
  end;
end;
for i:=1 to 32 do
begin
  for j:=1 to 4 do
begin
  output[i,j]:=1/(1+exp(-out[i,j]));
end;
end;
for i:=1 to 32 do
begin
  rms_error[i]:=sqrt((sqr(target[1]-output[i,1])
                     +sqr(target[2]-output[i,2])
                     +sqr(target[3]-output[i,3])
                     +sqr(target[4]-output[i,4]))/4);
end;

{rank depending on RMS Error}
for i:=1 to 32 do
begin
  temp_fit[i]:=rms_error[i];
  p_fit[i]:=i;
end;
for i:=1 to 31 do
begin
  for j:=1 to 31 do
  begin
    if temp_fit[j] > temp_fit[j+1] then
    begin
      t_fit:=temp_fit[j];
      p_temp:=p_fit[j];
      temp_fit[j]:=temp_fit[j+1];
      p_fit[j]:=p_fit[j+1];
      temp_fit[j+1]:=t_fit;
      p_fit[j+1]:=p_temp;
    end;
  end;
end;
for i:=1 to 32 do
begin
  for j:=1 to 9 do
  begin
    rms_error[i]:=rms_error[p_fit[i]];
    weight[i,j]:=weight[p_fit[i],j];
  end;
end;
for i:=1 to 32 do
begin
  str(rms_error[i]:9:7,rms[i]);
end;

win_15(rms); win_25(rms); win_35(rms); win_45(rms);
for i:=1 to 32 do
  val(rms[i],rms_error[i],c);
if (rms_error[p_fit[1]] <= 0.05) then
begin
    train:=false;
    start:=false;
end
else
    train:=true;

while train do
begin

    {multiplication}
    multiplication(weight);

    {biased-mutation of weights}
    mutation(weight);

    {crossover weights}
    crossover(weight);

    for i:=1 to 32 do
    begin
        for j:=1 to 9 do
            str(weight[i,j]:3:1,ran_num[i,j]);
    end;

    for i:=1 to 32 do
    begin
        for j:=1 to 9 do
        begin
            val(ran_num[i,j],weight[i,j],c);
            if weight[i,j] >= 0 then ran_num[i,j] :=
                ' + ran_num[i,j];
        end;
    end;

    win_11(ran_num); win_12(ran_num);
    win_13(ran_num); win_14(ran_num);
    win_21(ran_num); win_22(ran_num);
    win_23(ran_num); win_24(ran_num);
    win_31(ran_num); win_32(ran_num);
    win_33(ran_num); win_34(ran_num);
    win_41(ran_num); win_42(ran_num);
    win_43(ran_num); win_44(ran_num);

    train:=false; jump_out:=false;
    if keypressed then ch:=readkey;
    if ch=#27 then
    begin
        train:=false; start:=false; jump_out:=true;
    end;
end; {while train=true}
end; {while start=true}
while not jump_out do begin
  msg[1]:='Success !';
  msg[2]:='RMS ERROR<=0.05';
  msg[3]:='Press any key';
  msg[4]:='to main manu !';
  win 1(0,msg);
  sound(550); delay(50); nosound; delay(100);
  if keypressed then
    repeat ch:=readkey until not keypressed;
    setlinestyle(0,0,3);
    repeat
      begin
        setcolor(white); line(134,75,142,75);
        delay(150); setcolor(lightblue);
        line(134,75,142,75); delay(150);
      end;
      until keypressed;
    if keypressed then begin
      ch:=readkey; jump_out:=true;
    end;
    setlinestyle(0,0,1); setcolor(white);
  end;
end; {train_wei}

Begin {main program}
  exit_flag:=false;
  main_manu(exit_flag);
  if not exit_flag then repeat
    begin
      gra_init;
      initial;
      randomize;
      set_random(weight);
      train_wei(weight);
      main_manu(exit_flag);
    end
    until exit_flag;
  closegraph;
  textmode(lastmode);
End. {main}
program GAFPR; {This program solves the inverse kinematics for robots with multiple degrees of freedom}

Uses Crt,Graph,Dos;

type msg_type=array[1..1] of string;
  input_type=array[1..8] of string[1];
  theta_str_type=array[1..3,1..16] of string[8];
  theta_num_type=array[1..3,1..16] of real;
  function_str_type=array[1..16] of string[8];
  function_num_type=array[1..16] of real;
  temp_num_type=array[1..3] of real;

var graphdriver,graphmode:integer;
  x,y,i,j,k,l,m,n:integer;

  exit_flag,key_press,cursor,out_flag,out_graph,out_retrain:boolean;
  not_do_1:boolean;
  Theta2,Theta3,Theta4,Xf,Yf:input_type;
  Theta2_str,Theta3_str,Theta4_str:string[8];
  Xf_str,Yf_str:string[8];
  Theta1_str:string;
  Theta2_num,Theta3_num,Theta4_num:real;
  xi1,yi1,xi2,yi2,xi3,yi3,Xf_num,Yf_num,Theta1_num:real;
  theta_str:theta_str_type;
  theta_num:theta_num_type;
  r_num,p_fit,p_err,p_disp:array[1..16] of integer;
  error_num,disp_num,fit_num:function_num_type;
  error_str,disp_str,fit_str,r_str:function_str_type;

procedure main_menu(var exit_flag,out_flag,out_graph,out_retrain:boolean);
{Draw main menu}

var ch:char;

begin{main_menu}
  detectgraph(graphdriver,graphmode);
  initgraph(graphdriver,graphmode,'');
  setvisualpage(0); setactivepage(0);
  setbkcolor(lightblue);
  setviewport(0,0,640,450,true);
  clearviewport; setcolor(white);
  x:=getmaxx; y:=getmaxy;
  line(round(x/3),20,10,20); line(10,20,10,y-20);
  line(10,y-20,x-10,y-20); line(x-10,y-20,x-10,20);
  line(x-10,20,2*round(x/3),20);
  line(round(x/3),24,15,24);
  line(15,24,15,y-24); line(15,y-24,x-15,y-24);
  line(x-15,y-24,x-15,24); line(x-15,24,2*round(x/3),24);
  settextstyle(1,0,2); outtextxy(round(x/3)+4,12,'ISE589 TERMPROJECT');
  settextstyle(1,0,1);
  outtextxy(140,40,'--- SPRING 1992, JAY-SHINN TSAI ---');
settextstyle(3,0,1);
outtextxy(40,80,'Welcome! This program is to demonstrate
a technique by');
outtextxy(140,100,'using the GENETIC ALGORITHMS included
three');
outtextxy(140,120,'operators: REPRODUCTION, MUTATION, and
CROSSOVER');
outtextxy(140,140,'to solve the FLOATING POINT problem of
MITSUBISHI');
outtextxy(140,160,'RM-501 ROBOT.');
outtextxy(140,200,'Users have to know six definitions as
follows:');
settextstyle(2,0,6);
line(140,240,600,240);
outtextxy(140,250,'Initial Point: original position of
d-end-effector.');
outtextxy(140,270,'Final Point: final position of
d-end-effector.');
outtextxy(140,290,'Theta 1: the angle of waist rotation.
[0-300]');
outtextxy(140,310,'Theta 2: the angle of shoulder
rotation. [60-190]');
outtextxy(140,330,'Theta 3: the angle of elbow rotation.
[90-180]');
outtextxy(140,350,'Theta 4: the angle of wrist pitch.
[90-270]');
line(140,380,600,380);
settextstyle(3,0,1);
outtextxy(140,405,'1. Initial training 2. Retraining or
ESC to exit !');
setlinestyle(0,0,3);
key_press:=false;
while not key_press do
begin
  for i:=1 to 2 do
  begin
    setcolor(white); line(557,425,565,425);
    delay(100); setcolor(lightblue);
    line(557,425,565,425); delay(100);
    end;
    if keypressed then ch:=readkey;
    if ((ch=#49) or (ch=#50) or (ch=#27)) then
      key_press:=true;
  end;
  setlinestyle(0,0,1);
  if ch=#49 then out_flag:=false;
  if ch=#27 then
  begin
    exit_flag:=true; out_flag:=true;
    out_graph:=true; out_retrain:=true;
  end;
if ch=#50 then 
begin 
  out_flag:=true; out_graph:=true; 
  out_retrain:=false; 
end; {main_manu}

procedure tell_box(mess_arr:msg_type);
{tell_box tells users what step we are doing now}
begin {tell_box}
  setviewport(3,3,636,42,true);
  setbkcolor(lightblue);
  clearviewport;
  setlinestyle(0,0,3);
  setcolor(white);
  line(0,38,638,38);
  settextrstyle(1,0,3);
  outtextxy(5,4,mess_arr[1]);
end; {tell_box}

procedure initial_input;
{initial initializes variables' array}
var i,j:integer;
begin {initial_input}
  for i:=1 to 8 do
  begin
    Theta2[i]:='';
    Theta3[i]:='';
    Theta4[i]:='';
    Xf[i]:='';
    Yf[i]:='';
  end;
end; {initial_input}

procedure input_manu(var
out_flag,out_graph,out_retrain:boolean;
var Theta2,Theta3,Theta4,Xf,Yf:input_type);

{input_manu to entry the pertinent data for training}
var ch:char;
  msg:msg_type;
begin {input_manu}
  while not out_flag do
  begin
    initial_input;
    setcolor(green);
    for i:=0 to 500 do
line(0,i,640,i); setcolor(lightblue);
for i:=0 to 500 do
line(0,i,640,i); setcolor(white);
setbkcolor(lightblue); setfillstyle(0,0);
setlinestyle(0,0,3); rectangle(1,1,638,478);
settextstyle(1,0,3);
msg[1]:='Please entry the pertinent data for training !';
tell_box(msg);
setviewport(3,42,636,477,true);
clearviewport; settextstyle(1,0,3);
setcolor(white);
outtextxy(15,15,'The Initial Point ---');
outtextxy(40,55,'Theta 2 :');
setlinestyle(0,0,1);
rectangle(150,61,240,81);
outtextxy(40,95,'Theta 3 :');
rectangle(150,101,240,121);
outtextxy(40,135,'Theta 4 :');
rectangle(150,141,240,161);
outtextxy(15,195,'The Final Point ----');
outtextxy(40,235,'X Coordinates :');
rectangle(220,241,310,261);
outtextxy(40,275,'Y Coordinates :');
rectangle(220,281,310,301);
delay(1200); sound(550); delay(50); nosound;
msg[1]:='Please entry Theta 2 of Link #2 ! [60 - 190]';
tell_box(msg);
setviewport(3,42,636,477,true);
settextstyle(2,0,6);
setlinestyle(0,0,1);
rectangle(330,110,608,220);
outtextxy(338,125,'* After entering all data , ');
outtextxy(338,145,' if your inputs are wrong , ');
outtextxy(338,165,' please go back to main');
outtextxy(338,185,' menu to start again !');
msg[1]:='O.K. ! Press RETURN or ESC ?';
tell_box(msg);
setviewport(3,42,636,477,true);
sound(550); delay(50); nosound;
settextstyle(1,0,3);
outtextxy(125,365,'Press ENTER to start training or');
outtextxy(195,395,'ESC to main menu !');
setlinestyle(0,0,3);
key_press:=true;
while key_press do
begin
for i:=1 to 2 do
begin
setcolor(white); line(436,415,444,415);
delay(100); setcolor(lightblue);
line(436,415,444,415); delay(100);
end;
if keypressed then ch:=readkey;
if ((ch=#13) or (ch=#27)) then key_press:=false;
end;
if ch=#27 then
begin
  msg[1]:='Go back to main menu !';
  tell box(msg);
  setviewport(3,42,636,477,true);
  setcolor(green);
  for i:=0 to 500 do
    line(0,i,640,i);
  out_flag:=true; out_graph:=true; out_retrain:=true;
end;
if ch=#13 then
begin
  out_flag:=true; out_graph:=false;
end;
end;{input_menu}

procedure set_random(var theta_str:theta_str_type);
{set_random to create random values for theta 2,3,4}
var i,j,k:integer;
begin{set_random}
  for i:=1 to 16 do
  begin
    k:=random(101);
    theta_num[1,i]:=60+130*k*0.01;
    str(theta_num[1,i]:6:2,theta_str[1,i]);
  end;
  for i:=1 to 16 do
  begin
    k:=random(101);
    theta_num[2,i]:=90+90*k*0.01;
    str(theta_num[2,i]:6:2,theta_str[2,i]);
  end;
  for i:=1 to 16 do
  begin
    k:=random(101);
    theta_num[3,i]:=90+180*k*0.01;
    str(theta_num[3,i]:6:2,theta_str[3,i]);
  end;
end;{set_random}
procedure training(var Theta2,Theta3,Theta4,Xf,Yf: input_type;
    var theta_str:theta_str_type; var Theta1_str:string;
    var not_do_1:boolean);

{***** training to train *****}
var msg:msg_type;
    train_out,GA_out:boolean;
    x,y,x1,x2,x3,y1,y2,y3,t1,t2,t3,f:real;
    c,s,m,n,w,v,p_temp:integer;
    temp_fit,temp_err,temp_disp:function_str_type;
    temp_media:theta_num_type;
    t_fit,t_err,tdisp:string;
    esc_key, ch:char;
    data1,data2,input:text;

begin(training)

    Theta2_str:=concat(Theta2[1],Theta2[2],Theta2[3],Theta2[4],
                        Theta2[5],Theta2[6],Theta2[7],Theta2[8]);
    Theta3_str:=concat(Theta3[1],Theta3[2],Theta3[3],Theta3[4],
                        Theta3[5],Theta3[6],Theta3[7],Theta3[8]);
    Theta4_str:=concat(Theta4[1],Theta4[2],Theta4[3],Theta4[4],
                        Theta4[5],Theta4[6],Theta4[7],Theta4[8]);
    Xf_str:=concat(Xf[1],Xf[2],Xf[3],Xf[4],Xf[5],Xf[6],Xf[7],
                    Xf[8]);
    Yf_str:=concat(Yf[1],Yf[2],Yf[3],Yf[4],Yf[5],Yf[6],Yf[7],
                    Yf[8]);
    val(Theta2_str,Theta2_num,c);
    val(Theta3_str,Theta3_num,c);
    val(Theta4_str,Theta4_num,c);
    val(Xf_str,Xf_num,c);
    val(Yf_str,Yf_num,c);
    assign(input,’result.txt’);
    rewrite(input);
    writeln(input,Theta2_num:6:2,’ ’,Theta3_num:6:2,’
               ’,Theta4_num:6:2);
    writeln(input,Xf_num:8:4,’ ’,Yf_num:8:4);
    close(input);
    xi1:=8.6614*cos((theta2_num-90)*(3.14159/180));
    yi1:=8.6614*sin((theta2_num-90)*(3.14159/180));
    xi2:=xi1+6.2992*cos((theta2_num+theta3_num-270)*
                        (3.14159/180));
    yi2:=yi1+6.2992*sin((theta2_num+theta3_num-270)*
                        (3.14159/180));
    xi3:=xi2+5.5118*cos((theta2_num+theta3_num+theta4_num-450)*
                        (3.14159/180));
    yi3:=yi2+5.5118*sin((theta2_num+theta3_num+theta4_num-450)*
                        (3.14159/180));
    x:=(xi3*Xf_num+yi3*Yf_num)/(sqrt(xi3*xi3+yi3*yi3))
        *sqrt(Xf_num*Xf_num+Yf_num*Yf_num));
    y:=(xi3*Yf_num-Xf_num*yi3)/(sqrt(xi3*xi3+yi3*yi3))
\[ \text{sqrt}(Xf_{\text{num}}*Xf_{\text{num}}+Yf_{\text{num}}*Yf_{\text{num}})) \]

\[ \text{Theta1}_{\text{num}}:=\text{arctan}(y/x)*(180/3.14159) \]

assign(input,'result.txt');
append(input); writeln(input,Theta1_{\text{num}}:6:2);
close(input);

\text{train \_out}:=\text{false};
while not train \_out do {training procedure}
begin{train \_out}
 \text{msg}[1]:' training !';
tell box(msg);
setviewport(3,42,636,477,true);
for \text{i}=1 to 3 do
begin
 \text{for } \text{j}=1 to 16 do
 \text{val(theta \_str[i,j],theta \_num[i,j],c)};
end;
\text{for } \text{i}=1 to 16 do
begin
 \text{x1}=8.6614*cos((theta \_num[1,i]-90)*(3.14159/180));
 \text{y1}=8.6614*sin((theta \_num[1,i]-90)*(3.14159/180));
 \text{x2}=x1+6.2992*cos((theta \_num[1,i]+theta \_num[2,i]-270)*(3.14159/180));
 \text{y2}=y1+6.2992*sin((theta \_num[1,i]+theta \_num[2,i]-270)*(3.14159/180));
 \text{x3}=x2+5.5118*cos((theta \_num[1,i]+theta \_num[2,i])
 \quad+theta \_num[3,i]-450)*(3.14159/180));
 \text{y3}=y2+5.5118*sin((theta \_num[1,i]+theta \_num[2,i])
 \quad+theta \_num[3,i]-450)*(3.14159/180));
 \text{error \_num[i]}:=sqrt((Xf_{\text{num}}-x3)*(Xf_{\text{num}}-x3)
 \quad+(Yf_{\text{num}}-y3)*(Yf_{\text{num}}-y3));
\text{str(error \_num[i]:7:4,error \_str[i])};
end;
\text{for } \text{i}=1 to 16 do
begin
 \text{t1}=abs(theta \_num[1,i]-Theta2\_num);
 \text{t2}=abs(theta \_num[2,i]-Theta3\_num);
 \text{t3}=abs(theta \_num[3,i]-Theta4\_num);
 \text{disp \_num[i]}:=t1;
 if (t2-d\_num[i]) > 0 then disp \_num[i]:=t2;
 if (t3-d\_num[i]) > 0 then disp \_num[i]:=t3;
\text{str(disp \_num[i]:6:2,disp \_str[i])};
end;
\text{for } \text{i}=1 to 16 do
begin
 \text{val(error \_str[i],error \_num[i],c)};
 \text{val(disp \_str[i],disp \_num[i],c)};
 \text{fit \_num[i]}:=error \_num[i]+0.2*disp \_num[i];
\text{str(fit \_num[i]:8:4,fit \_str[i])};
end;

{****** rank depending on "Fitness" ******}
for i:=1 to 16 do
begin
  temp_fit[i]:=fit_str[i];
  p_fit[i]:=i;
end;
for i:=1 to 15 do
begin
  for j:=1 to 15 do
begin
    if temp_fit[j] > temp_fit[j+1] then
begin
      t_fit:=temp_fit[j];
      p_temp:=p_fit[j];
      temp_fit[j]:=temp_fit[j+1];
      p_fit[j]:=p_fit[j+1];
      temp_fit[j+1]:=t_fit;
      p_fit[j+1]:=p_temp;
    end;
  end;
end;
for i:=1 to 16 do
begin
  r_num[p_fit[i]]:=i;
end;
for i:=1 to 16 do
begin
  str(r_num[i]:2,r_str[i]);
end;
for i:=1 to 16 do
begin
  setviewport(295,120+(i-1)*23,375,130+(i-1)*23,true);
  clearviewport;
  settextstyle(0,0,1);
  outtextxy(15,0,error_str[i]);
end;
for i:=1 to 16 do
begin
  setviewport(390,120+(i-1)*23,470,130+(i-1)*23,true);
  clearviewport;
  settextstyle(0,0,1);
  outtextxy(15,0,disp_str[i]);
end;
for i:=1 to 16 do
begin
  setviewport(485,120+(i-1)*23,565,130+(i-1)*23,true);
  clearviewport;
  settextstyle(0,0,1);
  outtextxy(9,0,fit_str[i]);
end;
for i:=1 to 16 do
begin
  setviewport(580,120+(i-1)*23,630,130+(i-1)*23,true);
  clearviewport;
  settextstyle(0,0,1);
  outtextxy(17,0,r_str[i]);
end;
{hit ESC to jump to DOS to use Neural Network}
{and save current data to "status.txt" and "jay.txt"}

if keypressed then esc_key:=readkey;
if esc_key=#27 then
begin
  assign(data1,'status.txt');
  rewrite(data1);
  assign(data2,'jay.txt');
  rewrite(data2);
  for i:=1 to 16 do
  begin
    writeln(data1,theta_num[1,i]:6:2,',theta_num[2,i]:6:2',',
           theta_num[3,i]:6:2);
    writeln(data2,theta_num[1,i]:6:2,',theta_num[2,i]:6:2',',
           theta_num[3,i]:6:2);
  end;
  close(data1); close(data2);
  GA_out:=true; train_out:=true;
  esc_key:=#52; not_do_1:=false;
end;

{******** if any error <=0.001 then "over" ! *******}
for i:=1 to 16 do
  val(fit_str[i],fit_num[i],c);
for i:=1 to 16 do
begin
  if (error_num[i] <= 0.001) then
  begin
    train_out:=true; GA_out:=true;
    not_do_1:=true; i:=16;
  end
  else
  begin
    GA_out:=false;
  end;
end;

while not GA_out do       {genetic process}
begin

  {*** choose 4 smaller depending on "Fitness" ***}
  {******** reproduction *******}
  msg[1]:='Reproduction !';
tell_box(msg);
  for i:=1 to 3 do
  begin
    for j:=1 to 4 do
      media[i,j]:=theta_num[i,p_fit[j]];
  end;
end;
for i:=1 to 3 do
begin
for j:=1 to 4 do begin
  theta_num[i,j]:=media[i,j];
  theta_num[i,j+4]:=media[i,j];
  theta_num[i,j+8]:=media[i,j];
  theta_num[i,j+12]:=media[i,j];
end;
end;

{******* mutation *******}
msg[1]:=‘Mutation !’;
tell_box(msg);
for i:=1 to 3 do begin
  for j:=1 to 16 do begin
    s:=random(4); {probability=0.75}
    if (s=0) or (s=1) or (s=2) then begin
      val(error_str[j],error_num[j],c);
      if error_num[j] > 20 then {mutation rules}
        begin
          m:=random(101);
          n:=random(2);
          if n=0 then n:=n-1;
          f:=2.5+5.5*m*n*0.01;
        end;
      if (error_num[j]>10) and (error_num[j]<20) then begin
        m:=random(101);
        n:=random(2);
        if n=0 then n:=n-1;
        f:=1+4*m*n*0.01;
      end;
      if (error_num[j]>5) and (error_num[j]<10) then begin
        m:=random(101); n:=random(2);
        if n=0 then n:=n-1;
        f:=0.25+2.75*m*n*0.01;
      end;
      if (error_num[j]>1.5) and (error_num[j]<5) then begin
        m:=random(101); n:=random(2);
        if n=0 then n:=n-1;
        f:=0.01+1.99*m*n*0.01;
      end;
      if (error_num[j]>0.2) and (error_num[j]<1.5) then begin
        m:=random(101);
        n:=random(2);
if n=0 then n:=n-1;
f:=0.01+0.49*m*n*0.01;
end;
if error_num[j] < 0.2 then begin
  m:=random(101);
n:=random(2);
  if n=0 then n:=n-1;
f:=0.01+0.19*m*n*0.01;
end;
theta_num[i,j]:=theta_num[i,j]+f;
end;
end;

{****** limitation for theta 2,3,4 ******}
for i:=1 to 16 do begin
  if theta_num[1,i] < 60 then theta_num[1,i]:=60;
  if theta_num[1,i] > 190 then theta_num[1,i]:=190;
end;
for i:=1 to 16 do begin
  if theta_num[2,i] < 90 then theta_num[2,i]:=90;
  if theta_num[2,i] > 180 then theta_num[2,i]:=180;
end;
for i:=1 to 16 do begin
  if theta_num[3,i] < 90 then theta_num[3,i]:=90;
  if theta_num[3,i] > 270 then theta_num[3,i]:=270;
end;

{****** crossover ******}
msg[1]:='Crossover !';
tell box(msg);
for i:=1 to 3 do begin
  for j:=1 to 16 do begin
    temp[i,j]:=theta_num[i,j];
  end;
end;
for i:=1 to 8 do begin
  for j:=1 to 3 do begin
    w:=random(4); {probability=0.25}
    if (w=0) then
begin
  theta_num[j,2*i-1]:=theta_num[j,2*i];
  theta_num[j,2*i]:=temp[j,2*i-1];
end;
end;
end;

for i:=1 to 3 do
begin
  for j:=1 to 16 do
    str(theta_num[i,j]:6:2,theta_str[i,j]);
  setviewport(3,42,636,477,true);
  settextstyle(0,0,1);
  for i:=1 to 3 do
  begin
    for j:=1 to 16 do
    begin
      setviewport(10+(i-1)*95,120+(j-1)*23,90+(i-1)*95,130+(j-1)*23,true);
      clearviewport;
      outtextxy(150,theta_str[i,j]);
    end;
    GA_out:=true;
  end;
end;
end;

procedure gra_init(var
  out_graph,out_flag,out_retrain,exit_flag:boolean;
  var Theta2,Theta3,Theta4,Xf,Yf:input_type);

{gra_init to express how is the training for theta 2,3,4 going}
var ch:char;
  msg:msg_type;
  i,j,k:integer;
begin{gra_init}
  while not out_graph do
  begin{out_graph}
    msg[1]:='Go to training menu !';
    tell_box(msg);
    setviewport(3,42,636,477,true);
    setcolor(green);
    for i:=0 to 500 do
      line(0,i,640,i); setcolor(lightblue);
    for i:=0 to 500 do
      line(0,i,640,i); setcolor(white);
    setbkcolor(lightblue);
setfillstyle(0,0); setlinestyle(0,0,3);  
msg[1]:='Create random values for Theta 2,3,4 ! Please wait !';  
tell_box(msg);  
setviewport(3,42,636,477,true); settextstyle(1,0,1);  
outtextxy(120,15,'Training Table (Press ESC to DOS in processing !)');  
settextstyle(0,0,1); setcolor(white);  
setlinestyle(0,0,2); for i:= 1 to 17 do line(0,45+(i-1)*23,640,45+(i-1)*23);  
for i:=1 to 6 do line(95+(i-1)*95,45,95+(i-1)*95,480);  
outtextxy(17,54,'Theta 2');  
outtextxy(116,54,'Theta 3');  
outtextxy(211,54,'Theta 4');  
outtextxy(315,54,'Error');  
outtextxy(382,54,'Displacement');  
outtextxy(495,54,'Fitness');  
outtextxy(590,54,'Rank');  
delay(2500);  
set_random(theta_str);  
for i:=1 to 3 do begin  
for j:=1 to 16 do begin  
setviewport(10+(i-1)*95,120+(j-1)*23,90+(i-1)*95,130+(j-1)*23,true);  
clearviewport;  
outtextxy(15,0,theta_str[i,j]);  
end;  
end;  
msg[1]:='Completed ! Press ENTER to start training !';  
tell_box(msg);  
setviewport(3,42,636,477,true);  
sound(550); delay(50); nosound;  
key_press:=false; while not key_press do begin  
if keypressed then ch:=readkey;  
if ch=#13 then key_press:=true;  
end;  

training(Theta2,Theta3,Theta4,Xf,Yf,theta_str,Theta1_str,not_do_1);  

if not_do_1 then begin  
msg[1]:='Success ! Press ESC to go back main menu !';
tell_box(msg);
sound(550); delay(50); nosound;
key_press:=false;
while not key_press do
begin
  if keypressed then ch:=readkey;
  if ch=#27 then key_press:=true;
end;
msg[1]:='Go back to main menu !';
tell_box(msg); setviewport(3,42,636,477,true);
setcolor(green);
for i:=O to 500 do
  line(0,i,640,i);
  setcolor(lightblue);
for i:=O to 500 do
  line(0,i,640,i);
out_graph:=true; out_retrain:=true;
end;
else
begin
  out_graph:=true; out_flag:=true;
  out_retrain:=true; exit_flag:=true;
end;
end;{out_graph}
end;{gra_init}

{***** main program *****}
Begin{main}
  exit_flag:=false;
  while not exit_flag do
begin
  main_manu(exit_flag,out_flag,out_graph,out_retrain);
  if not out_flag then
  begin
    input_manu(out_flag,out_graph,out_retrain,
               Theta2,Theta3,Theta4,Xf,Yf);
    randomize;
    gra_init(out_graph,out_flag,out_retrain,exit_flag,
             Theta2,Theta3,Theta4,Xf,Yf);
  end;
end;
closegraph;
textmode(lastmode);
End.{main}
program JSSDEMO; {This program solves the job shop scheduling}

Uses Crt;

type input_data_type=array[1..15,1..4] of integer;
  setup_time_type=array[1..7,1..7] of integer;
  schedule_type=array[1..15,1..50] of integer;
  output_type=array[1..50] of real;
  x_type=array[1..8] of real;
  q_type=array[1..15] of integer;
  eg_type=array[1..4,1..15] of integer;
  value_type=array[1..15,1..50] of integer;
  p_fit_type=array[1..50] of integer;
  fitness_type=array[1..50,1..50] of real;

var i,j,k,l:integer;
  key_press,exit_flag,out_input,out_output,
    jump_out:boolean;
  x:x_type;
  x_over,yours,job,cur,pre,count,right_count,muta:integer;
  ip:input_data_type;
  sp:setup_time_type;
  sch:schedule_type;
  max_flow,mean_flow,max_tard,mean_tard,
  wip,util,th_put:output_type;
  no_tardy:output_type;
  fitness:fitness_type;

procedure input_manu(var out_input,out_output:boolean);

{****** input_manu to provide the data input area *******
var ch:char;
  right:boolean;
  JJJ:text;

begin{input_manu}
  while not-out_input do begin
    assign(JJJ,'jssdata.txt'); {read data from jssdata.txt}
    reset(JJJ); readln(JJJ,job,cur); close(JJJ);
    window(1,1,80,25); clrscr;
    textbackground(black); window(1,1,80,25); clrscr;
    textbackground(lightblue); textcolor(white);
    write('\\\\ Please Input the Pertinent Data for Training
    \\
    A. The Importance for Performance measures:
    \
    1:very important, 0.5:important,'
0:not important "\n\nwrite(' The importance for the Maximum Flow Time: "\nwrite(' The importance for the Mean Flow Time: "\nwrite(' The importance for the Maximum Tardiness: "\nwrite(' The importance for the Mean Tardiness: "\nwrite(' The importance for the Work-In Process Inventory: "\nwrite(' The importance for the Resource Utilization: "\nwrite(' The importance for the Throughput: "\nwrite(' The importance for the Number of Tardy Jobs: "\nwrite(' B. The Crossover Method: C. Mutation? (1:Yes, 2:No) "\nwrite(' 1:Partially-Mapped Crossover. 2:Order Crossover #1. "\nwrite(' 3:Order Crossover #2. 4:Enhanced Edge Recombination. "\nwrite(' D. How many schedules you want to input?(0-5) "\nwrite('"

right:=false;
while not right do begin
window(50,6,57,6); textbackground(brown);
crscr; gotoxy(1,1); textcolor(yellow);
readln(x[1]); write(x[1]:7:5);
window(47,7,54,7); clrscr; gotoxy(1,1);
readln(x[2]); write(x[2]:7:5); window(50,8,57,8);
crscr; gotoxy(1,1); readln(x[3]);
write(x[3]:7:5); window(47,9,54,9);
crscr; gotoxy(1,1); readln(x[4]);
write(x[4]:7:5); window(58,10,65,10);
crscr; gotoxy(1,1); readln(x[5]);
write(x[5]:7:5); window(53,11,60,11);
clrscr; gotoxy(1,1); readln(x[6]);
write(x[6]:7:5); window(43,12,50,12);
clrscr; gotoxy(1,1); readln(x[7]);
write(x[7]:7:5); window(53,13,60,13);
clrscr; gotoxy(1,1); readln(x[8]);
write(x[8]:7:5); window(30,14,32,14);
clrscr; gotoxy(1,1); readln(x_over);
write(x_over:2); window(69,14,71,14);
clrscr; gotoxy(1,l);
readln (x_over);
write(x_over:2);

if yours=0 then begin
  window(9,18,61,22); textbackground(lightblue);
crscr;
end else begin
  window(9,18,61,22); textbackground(lightcyan);
crscr;
  for i:=1 to yours do
    begin
gotoxy(1,i);
for j:=1 to job do
  begin
    window(9+(j-1)*3,17+i,12+(j-1)*3,17+i);
    textbackground(lightcyan); crscr;
    readln(sch[j,i]); write(sch[j,i]:2);
  end;
end;
end;
window(20,23,75,23); textbackground(lightblue);
crscr; gotoxy(3,1); textcolor(yellow);
write('Are you happy with your inputs? (Y/N)');
gotoxy(40,1); sound(550); delay(50);
nosound; readln(ch);
if upcase(ch)='Y' then right:=true;
if upcase(ch)='N' then begin
  window(20,23,75,23); textbackground(lightblue);
crscr; gotoxy(13,1); textcolor(yellow);
write('Please input again !');
gotoxy(33,1); sound(550); delay(50); nosound;
right:=false;
end;
end;
window(20,23,75,23); textbackground(lightblue);
crscr; gotoxy(6,1); textcolor(yellow);
write('Press any key to start training !');
gotoxy(39,1); sound(550); delay(50); nosound;
repeat
until keypressed;
if keypressed then
begin
exit_flag:=true; out_input:=true; out_output:=false;
end;
end;{input_manu}

procedure random_sch(var sch: schedule_type);
{random to generate 50 different schedules randomly}
var a,b:integer;
flag:boolean;
begin{random}
for i:=yours+1 to 50 do
begin
a:=random(job); a:=a+1;
sch[1,i]:=a;
for j:=2 to job do
begin
repeat
begin
flag:=true;
b:=random(job); b:=b+1;
for k:=j-1 downto 1 do
if b=sch[k,i] then flag:=false;
end;
until flag;
sch[j,i]:=b;
end;
end;
end;{random}

procedure pmx(var sch: schedule_type);
{pmx to do the Partially-Mapped Crossover operation}
var v,w,num,rn1,rn2:integer;
compare:boolean;
temp: schedule_type;
begin{pmx}
for i:=1 to job do
begin
for \( j := 1 \) to 50 do
\[
\text{temp}[i, j] := \text{sch}[i, j];
\]
end;

{****** 1st and 2nd schedules do not need "PMX" ******)

for \( \text{num} := 2 \) to 25 do
begin
\[
v := \text{random}(\text{job-2}); \ w := \text{random}(\text{job-2});
\]
compare := true;
while compare do
begin
if \( v > w \) then
begin
repeat
\[
v := \text{random}(\text{job-2}); \ w := \text{random}(\text{job-2});
\]
until \( v \leq w \);
end;
\[
rn1 := v + 2; \ \ rn2 := w + 2;
\]
compare := false;
end;
for \( i := rn1 \) to \( rn2 \) do
begin
\[
\text{sch}[i, (2*\text{num}) - 1] := \text{temp}[i, 2*\text{num}];
\]
\[
\text{sch}[i, 2*\text{num}] := \text{temp}[i, (2*\text{num}) - 1];
\]
end;
for \( i := 1 \) to \( rn1 - 1 \) do
begin
for \( j := 1 \) to \( rn2 - rn1 + 1 \) do
begin
for \( k := rn1 \) to \( rn2 \) do
begin
if \( \text{sch}[i, (2*\text{num}) - 1] = \text{sch}[k, (2*\text{num}) - 1] \) then
\[
\text{sch}[i, (2*\text{num}) - 1] := \text{temp}[k, (2*\text{num}) - 1];
\]
if \( \text{sch}[i, 2*\text{num}] = \text{sch}[k, 2*\text{num}] \) then
\[
\text{sch}[i, 2*\text{num}] := \text{temp}[k, 2*\text{num}];
\]
end;
end;
end;
for \( i := rn2 + 1 \) to \( \text{job} \) do
begin
for \( j := 1 \) to \( rn2 - rn1 + 1 \) do
begin
for \( k := rn1 \) to \( rn2 \) do
begin
if \( \text{sch}[i, (2*\text{num}) - 1] = \text{sch}[k, (2*\text{num}) - 1] \) then
\[
\text{sch}[i, (2*\text{num}) - 1] := \text{temp}[k, (2*\text{num}) - 1];
\]
if \( \text{sch}[i, 2*\text{num}] = \text{sch}[k, 2*\text{num}] \) then
\[
\text{sch}[i, 2*\text{num}] := \text{temp}[k, 2*\text{num}];
\]
end;
procedure ox1(var sch:schedule_type);

{****** ox1 to do the Order Crossover #1 operation ******}

var v,w,num,rn1,rn2,qq:integer;
  compare,gate:boolean;
  temp:schedule_type;

begin{ox1}
  for i:=1 to job do
  begin
    for j:=1 to 50 do
      temp[i,j]:=sch[i,j];
  end;

  {****** 1st and 2nd schedules do not need "OX1" ******}

  for num:=2 to 25 do
    begin
      v:=random(job-2);
      w:=random(job-2);
      compare:=true;
      while compare do
        begin
          if v > w then
            begin
              repeat
                v:=random(job-2);
                w:=random(job-2);
              until v <= w;
            end;
            rn1:=v+2;
            rn2:=w+2;
            compare:=false;
          end;
      for i:=rn1 to rn2 do
        begin
          sch[i,2*num]:=temp[i,2*num];
          sch[i,(2*num)-1]:=temp[i,(2*num)-1];
        end;
      for i:=rn2+1 to job do
        begin
          gate:=true;
          for j:=rn2+1 to job do
            begin
              \text{...}
            end;
        end;
    end;
end;{pmx}
qq:=0;
for k:=rn1 to rn2 do
  if temp[j,(2*num)-1] <> sch[k,(2*num)-1] then
    qq:=qq+1;
  if qq=rn2-rn1+1 then
    begin
      sch[i,(2*num)-1]:=temp[j,(2*num)-1];
      temp[j,(2*num)-1]:=sch[rn1,(2*num)-1];
      j:=job;
      gate:=false;
    end;
end;
while gate do
begin
  for j:=1 to rn2 do
    begin
      qq:=0;
      for k:=rn1 to rn2 do
        if temp[j,(2*num)-1] <> sch[k,(2*num)-1] then
          qq:=qq+1;
        if qq=rn2-rn1+1 then
          begin
            sch[i,(2*num)-1]:=temp[j,(2*num)-1];
            temp[j,(2*num)-1]:=sch[rn1,(2*num)-1];
            j:=rn2;
            gate:=false;
          end;
    end;
  end;
end;
for i:=1 to rn1-1 do
begin
  gate:=true;
  for j:=rn2+1 to job do
    begin
      qq:=0;
      for k:=rn1 to rn2 do
        if temp[j,(2*num)-1] <> sch[k,(2*num)-1] then
          qq:=qq+1;
        if qq=rn2-rn1+1 then
          begin
            sch[i,(2*num)-1]:=temp[j,(2*num)-1];
            temp[j,(2*num)-1]:=sch[rn1,(2*num)-1];
            j:=job;
            gate:=false;
          end;
    end;
  while gate do
    begin
      for j:=1 to rn2 do
        begin
          qq:=0;
        end;
  end;
end;
for k:=rn1 to rn2 do
  if temp[j,(2*num)-1] <> sch[k,(2*num)-1] then
    qq:=qq+1;
  if qq=rn2-rn1+1 then
    begin
      sch[i,(2*num)-1]:=temp[j,(2*num)-1];
      temp[j,(2*num)-1]:=sch[rn1,(2*num)-1];
      j:=rn2;
      gate:=false;
    end;
  end;
end;
end;
for i:=rn2+1 to job do
begin
  gate:=true;
  for j:=rn2+1 to job do
begin
    qq:=0;
    for k:=rn1 to rn2 do
      if temp[j,2*num] <> sch[k,2*num] then qq:=qq+1;
    if qq=rn2-rn1+1 then
      begin
        sch[i,2*num]:=temp[j,2*num];
        temp[j,2*num]:=sch[rn1,2*num];
        j:=job;
        gate:=false;
      end;
    end;
  while gate do
  begin
    for j:=1 to rn2 do
    begin
      qq:=0;
      for k:=rn1 to rn2 do
        if temp[j,2*num] <> sch[k,2*num] then qq:=qq+1;
      if qq=rn2-rn1+1 then
        begin
          sch[i,2*num]:=temp[j,2*num];
          temp[j,2*num]:=sch[rn1,2*num];
          j:=rn2;
          gate:=false;
        end;
    end;
  end;
end;
for i:=1 to rn1-1 do
begin
  gate:=true;
  for j:=rn2+1 to job do
  begin
    qq:=0;
  end;
for k:=rnl to rn2 do
if temp[j,2*num] <> sch[k,2*num] then qq:=qq+1;
if qq=rn2-rnl+1 then
begin
    sch[i,2*num]:=temp[j,2*num];
    temp[j,2*num]:=sch[rn1,2*num];
    j:=job;
    gate:=false;
end;
end;
while gate do
begin
    for j:=1 to rn2 do
    begin
        qq:=0;
        for k:=rnl to rn2 do
        if temp[j,2*num] <> sch[k,2*num] then qq:=qq+1;
        if qq=rn2-rnl+1 then
        begin
            sch[i,2*num]:=temp[j,2*num];
            temp[j,2*num]:=sch[rn1,2*num];
            j:=rn2;
            gate:=false;
        end;
        end;
    end;
end;
end;
end;

procedure ox2(var sch:schedule_type);
{****** ox2 to do the Order Crossover #2 operatation ******}
var num,v,w,t_rn,tem1,tem3:integer;
    rn:array[1..20] of integer;
    temp:array[1..20,1..4] of integer;
    flag:boolean;
begin{ox2}
{****** 1st and 2nd schedules do not need "OX2" ******}
for num:=2 to 25 do
begin{num}
    v:=random(job-2); v:=v+2;
    rn[1]:=random(job); rn[1]:=rn[1]+1;
    for i:=2 to v do
    begin

repeat
begin
  flag:=true;
  w:=random(job); w:=w+1;
  for j:=i-1 downto 1 do
    if w=rn[j] then flag:=false;
  end;
until flag;
  rn[i]:=w;
end;
for i:=1 to v-1 do  {Sorting for random number}
begin
  for j:=1 to v-1 do
    begin
      if rn[j] > rn[j+1] then
        begin
          t_rn:=rn[j];
          rn[j]:=rn[j+1];
          rn[j+1]:=t_rn;
        end;
    end;
  end;
for i:=1 to v do
begin
  for j:=1 to job do
    begin
      if sch[rn[i],2*num] = sch[j,(2*num)-1] then
        begin
          temp[i,1]:=j;
          temp[i,2]:=sch[rn[i],2*num];
        end;
    end;
end;
for i:=1 to v do
begin
  for j:=1 to job do
    begin
      if sch[rn[i],(2*num)-1] = sch[j,2*num] then
        begin
          temp[i,3]:=j;
          temp[i,4]:=sch[rn[i],(2*num)-1];
        end;
    end;
end;
for i:=1 to v-1 do  {***** Sorting for position *****}
begin
  for j:=1 to v-1 do
    begin
      if temp[j,1] > temp[j+1,1] then
        begin
          temp1:=temp[j,1];
          temp[j,1]:=temp[j+1,1];
          temp[j+1,1]:=temp1;
        end;
    end;
end;
end;
procedure eer(var sch:schedule_type);
{
{eer to do the Enhanced Edge Recombination Crossover operation}

var nm,v,w:integer;
flag:boolean;
eg1,eg2:eg_type;
q1,q2:q_type;
tem:array[1..4] of integer;
begin{eer}

{***** 1st and 2nd schedules do not need "EER" *****}

for nm:=2 to 25 do
begin{nm}
eg1[1,sch[1,2*nm-1]]:=sch[job,2*nm-1];
eg1[2,sch[1,2*nm-1]]:=sch[2,2*nm-1];
eg1[1,sch[job,2*nm-1]]:=sch[job-1,2*nm-1];
eg1[2,sch[job,2*nm-1]]:=sch[1,2*nm-1];
for i:=2 to job-1 do
begin
  egl[1,sch[i,2*nm-1]]:=sch[i-1,2*nm-1];
egl[2,sch[i,2*nm-1]]:=sch[i+1,2*nm-1];
do
end;
end;
end;
end;

for i:=1 to v-1 do   {***** Sorting for position *****}
begin
  for j:=1 to v-1 do
  begin
    if temp[j,3] > temp[j+1,3] then
    begin
      tem3:=temp[j,3];
temp[j,3]:=temp[j+1,3];
temp[j+1,3]:=tem3;
    end;
  end;
end;
for i:=1 to v do
begin
  sch[temp[i,1],(2*num)-1]:=temp[i,2];
sch[temp[i,3],2*num]:=temp[i,4];
end;
end;
end;
end;
end

{******** Sorting for position ********}
egl[3, sch[job, 2*nm]] := sch[job, 2*nm];
egl[4, sch[job, 2*nm]] := sch[2, 2*nm];
egl[3, sch[job-1, 2*nm]] := sch[job-1, 2*nm];
egl[4, sch[job, 2*nm]] := sch[1, 2*nm];
for i := 2 to job-1 do
begin
  egl[3, sch[i, 2*nm]] := sch[i-1, 2*nm];
egl[4, sch[i, 2*nm]] := sch[i+1, 2*nm];
end;
for i := 1 to job do
begin
  if (egl[3, i] = egl[1, i]) then egl[3, i] := 21;
  if (egl[4, i] = egl[1, i]) then egl[4, i] := 21;
  if (egl[3, i] = egl[2, i]) then egl[3, i] := 21;
  if (egl[4, i] = egl[2, i]) then egl[4, i] := 21;
end;
for i := 1 to job do
begin
  q1[i] := 0; q2[i] := 0;
end;
for i := 1 to job do
begin
  for j := 1 to 4 do
  begin
    if egl[j, i] < 21 then q1[i] := q1[i] + 1;
  end;
end;
for i := 1 to job do
begin
  q2[i] := q1[i];
  for j := 1 to 4 do
  begin
    eg2[j, i] := egl[j, i];
  end;
sch[1, 2*nm-1] := sch[1, 2*nm-1]; {Offspring #1}
for i := 2 to job-1 do
begin
  offspring #1
  for j := 1 to 4 do
  begin
    tem[j] := egl[j, sch[i-1, 2*nm-1]];
  end;
  for j := 1 to 4 do
  begin
    if tem[j] < 21 then
    begin
      for k := 1 to 4 do
      begin
        if egl[k, tem[j]] = sch[i-1, 2*nm-1] then
        begin
          egl[k, tem[j]] := 21;
          q1[tem[j]] := q1[tem[j]] - 1;
          if q1[tem[j]] = 0 then q1[tem[j]] := 0;
        end;
        end;
      end;
    end;
  end;
end;
for j:=1 to 4 do
begin
if tem[j] < 21 then
begin
    sch[i,2*nm-1]:=tem[j];
    j:=4;
end;
end;
for j:=1 to 4 do
begin
if tem[j] < 21 then
begin
    if q1[tem[j]] <= q1[sch[i,2*nm-1]] then
        sch[i,2*nm-1]:=tem[j];
end;
end;
end;  

repeat
begin
    flag:=true;
v:=random(job); v:=v+1;
    for i:=job-1 downto 1 do
        if v=sch[i,2*nm-1] then flag:=false;
end;
until flag;
sch[job,2*nm-1]:=v;
sch[1,2*nm]:=sch[1,2*nm];  

for i:=2 to job-1 do
begin
    for j:=1 to 4 do
        tem[j]:=eg2[j,sch[i-1,2*nm]];
for j:=1 to 4 do
begin
    if tem[j] < 21 then
begin
        for k:=1 to 4 do
begin
            if eg2[k,tem[j]]==sch[i-1,2*nm] then
            begin
                eg2[k,tem[j]]:=21;
                q2[tem[j]]:=q2[tem[j]]-1;
                if q2[tem[j]]==0 then q2[tem[j]]:=0;
            end;
        end;
end;
for j:=1 to 4 do
begin
    if tem[j] < 21 then
begin
    sch[i,2*nm]:=tem[j];
    j:=4;
end;
end;
for j:=1 to 4 do
begin
  if temp[j] < 21 then
  begin
    if q2[temp[j]] <= q2[sch[i,2*nm]] then
      sch[i,2*nm]:=temp[j];
  end;
end; {offspring #2}
repeat
begin
  flag:=true;
  w:=random(job); w:=w+1;
  for i:=job-1 downto 1 do
    if w=sch[i,2*nm] then flag:=false;
end;
until flag;
sch[job,2*nm]:=w;
end; {nm}
end; {ear}

procedure mutation(var sch:schedule_type);
{****** mutation to do the mutation operation ******}

var v,w,num,rn1,rn2:integer;
  compare:boolean;
  temp:schedule_type;
  rate:integer;
begin{mutation}
  for i:=1 to job do
    begin
      for j:=1 to 50 do
        temp[i,j]:=sch[i,j];
    end;

{1st and 2nd schedules do not need "MUTATION"}

  for num:=3 to 50 do
    begin{num}
      rate:=random(20);
      if rate=0 then
        begin
          v:=random(job); v:=v+1;
          w:=random(job); w:=w+1;
          compare:=true;
          while compare do
          {end;}
        end;
    end;
end{mutation}
begin
  if v = w then
  begin
    repeat
      v:=random(job); v:=v+1;
      w:=random(job); w:=w+1;
      until v <> w;
    end;
    rnl:=v;
    rn2:=w;
    compare:=false;
  end;
  sch[rnl,num]:=temp[rn2,num];
  sch[rn2,num]:=temp[rn1,num];
end;
end;{num}
end;{mutation}

procedure crossover(var sch:schedule_type);

{****** crossover to choose the crossover method ******}
var pmicro,oxcro1,oxcro2,eecro:boolean;

begin{crossover}
  if x_over=1 then pmx(sch);
  if x_over=2 then ox1(sch);
  if x_over=3 then ox2(sch);
  if x_over=4 then eer(sch);
end;{crossover}
procedure training(var sch:schedule_type);

{training to start GENETIC ALGORITHMS processes}

var ch:char;
  tard,flow:value_type;
  out_train,out_GA:boolean;
  temp_fit:output_type;
  t_fit,q:real;
  p_fit:p_fit_type;
  data1,data2,data3:text;
  p_temp:integer;
  junk:schedule_type;

begin{training}
  count:=0;
  assign(data1,'jssdata.txt');  {read input data from 'jssdata.txt'}
  reset(data1);
  readln(data1,job,cur);
  readln(data1,pre);
  for i:=1 to job do
  begin
    readln(data1,ip[i,1],ip[i,2]);
    readln(data1,ip[i,3],ip[i,4]);
  end;
  close(data1);
  assign(data2,'jsssetup.txt');  {read setup time data from 'jsssetup.txt'}
  reset(data2);
  for i:=1 to 7 do
  readln(data2,sp[i,1],sp[i,2],sp[i,3],sp[i,4],sp[i,5],sp[i,6],sp[i,7]);
  close(data2);
  out_train:=false;
  while not out_train do
  begin{out_train}
    count:=count+1;

    {Calculation for Maximum Flow Time & Mean Flow Time}
    for i:=1 to 50 do
    begin
      flow[1,i]:=cur+sp[pre,ip[sch[1,i],1]]+ip[sch[1,i],3]
                 -ip[sch[1,i],2];
      for j:=2 to job do
        flow[j,i]:=flow[j-1,i]+ip[sch[j-1,i],2]
                   +sp[ip[sch[j-1,i],1],ip[sch[j-1,i],1]]
                   +ip[sch[j,i],3]-ip[sch[j,i],2];
    end;
  for i:=1 to 50 do
    begin
\( \text{max}_i \) = \text{flow}[1,i];
for \( j = 1 \) to \( \text{job} - 1 \) do
    if \( \text{flow}[j+1,i] > \text{max}_i \) then
        \( \text{max}_i = \text{flow}[j+1,i]; \)
end;
for \( i = 1 \) to 50 do
begin
    \( \text{qq} = 0; \)
for \( j = 1 \) to \( \text{job} \) do
    \( \text{qq} = \text{qq} + \text{flow}[j,i]; \)
mean_flow[i] = \( \text{qq} / \text{job}; \)
end;

{Calculation for Maximum Tardiness & Mean Tardiness}

{******* Calculation for Number of Tardy Jobs *******}

for \( i = 1 \) to 50 do
begin
    \( \text{tard}[1,i] = \text{cur} + \text{sp} [\text{pre}, \text{ip} [\text{sch}[1,i], 1]] + \text{ip}[\text{sch}[1,i], 3] \)
    \( - \text{ip}[\text{sch}[1,i], 4]; \)
for \( j = 2 \) to \( \text{job} \) do
    \( \text{tard}[j,i] = \text{tard}[j-1,i] + \text{ip}[\text{sch}[j-1,i], 4] \)
    \( + \text{sp}[\text{ip}[\text{sch}[j-1,i], 1], \text{ip}[\text{sch}[j,i], 1]] \)
    \( + \text{ip}[\text{sch}[j,i], 3] - \text{ip}[\text{sch}[j,i], 4]; \)
end;
for \( i = 1 \) to 50 do
begin
    \( \text{no}_{\text{tardy}}[i] = 0; \)
for \( i = 1 \) to 50 do
begin
    \( \text{if tard}[j,i] > 0 \) then \( \text{no}_{\text{tardy}}[i] = \text{no}_{\text{tardy}}[i] + 1 \)
end;
for \( i = 1 \) to 50 do
begin
    \( \text{max}_{\text{tard}}[i] = \text{tard}[1,i]; \)
for \( j = 1 \) to \( \text{job} - 1 \) do
    if \( \text{tard}[j+1,i] > \text{max}_{\text{tard}}[i] \) then
        \( \text{max}_{\text{tard}}[i] = \text{tard}[j+1,i]; \)
end;
for \( i = 1 \) to 50 do
begin
    \( \text{qq} = 0; \)
for \( j = 1 \) to \( \text{job} \) do
    \( \text{qq} = \text{qq} + \text{tard}[j,i]; \)
mean_tard[i] = \( \text{qq} / \text{job}; \)
end;
{Calculation for Work-In Process Inventory}

for i:=1 to 50 do begin
    qq:=0;
    qq:=qq+job*(ip[sch[1,i],3]+sp[pre,ip[sch[1,i],1]]);
    for j:=2 to job do
        qq:=qq+(job-j+1)*(ip[sch[j,i],3]
            +sp[ip[sch[j-1,i],1],ip[sch[j,i],1]]);
    wip[i]:=qq/(flow[job,i]+ip[sch[job,i],2]-cur);
end;

{****** Calculation for Resource Utilization ******}

for i:=1 to 50 do begin
    qq:=0;
    for j:=1 to job do
        qq:=qq+ip[sch[j,i],3];
    util[i]:=qq/(flow[job,i]+ip[sch[job,i],2]-cur);
end;

{****** Calculation for Throughput ******}

for i:=1 to 50 do
    th_put[i]:=job/(flow[job,i]+ip[sch[job,i],2]-cur);

{****** Calculation for Fitness Function ******}

for i:=1 to 50 do
    fitness[i,count]:=x[1]*max_flow[i]+x[2]*mean_flow[i]
        +x[3]*max_tard[i]
        +x[4]*mean_tard[i]+x[5]*wip[i]
        +x[6]*(1/util[i])+x[7]*(1/th_put[i])
        +x[8]*(no_tardy[i]/10);

{****** rank depending on "Fitness" ******}

for i:=1 to 50 do begin
    temp_fit[i]:=fitness[i,count];
    p_fit[i]:=i;
end;
for i:=1 to 49 do begin
    for j:=1 to 49 do begin
        if temp_fit[j] > temp_fit[j+1] then begin
            t_fit:=temp_fit[j];
            p_temp:=p_fit[j];
            temp_fit[j]:=temp_fit[j+1];
            p_fit[j]:=j;
            t_fit:=p_temp;
        end;
    end;
end;
p_fit[j]:=p_fit[j+1];
temp_fit[j+1]:=t_fit;
p_fit[j+1]:=p_temp;
end;
end;
right_count:=count-1;

{****** if " then "over" ******}  

if (fitness[p_fit[1],count]=fitness[p_fit[1],count-1])  
and  
(fitness[p_fit[1],count]=fitness[p_fit[1],count-2])  
then  
begin  
out_train:=true;
out_GA:=true;
jump_out:=false;
end  
else  
out_GA:=false;

{****** GENETIC ALGORITHMS’ training processes ******}  

while not out_GA do  
begin{out_GA}  
{ reproduction (choose 25 smaller depend on "fitness")}  
for i:=1 to job do  
begin  
for j:=1 to 50 do  
junk[i,j]:=sch[i,j];
end;
for i:=1 to job do  
begin  
for j:=1 to 25 do  
begin  
sch[i,j]:=junk[i,p_fit[j]];
sch[i,j+25]:=junk[i,p_fit[j]];
end;
end;
if muta=1 then mutation(sch);    {Mutation operation}
crossover(sch);    {****** Crossover Methods ******}  
out_GA:=true;
end;{out_GA}  
end;{out_train}  
end;{training}
procedure output_manu(var out_output, out_input, wxit_flag: boolean);

{output_manu to show the results of GENETIC processes}
var ch: char;

begin{output_manu}
while not out_output do
begin
  window(1,1,80,25);
  clrscr;
  textbackground(black);
  window(1,1,80,25);
  clrscr;
  textbackground(lightblue);
  textcolor(white);
  write('');
  write(' | Iterations: ');\
  write(' | The Near-Optimal Schedule #1= ');
  write(' | The Maximum Flow Time= ');
  write(' | K1= ');
  write(' | The Mean Flow Time= ');
  write(' | K2= ');
  write(' | The Maximum Tardiness= ');
  write(' | K3= ');
  write(' | The Mean Tardiness= ');
  write(' | K4= ');
  write(' | The Work-In Process Inventory= ');
  write(' | K5= ');
  write(' | The Resource Utilization= ');
  write(' | K6= ');
  write(' | The Throughput= ');
  write(' | K7= ');
  write(' | The Number of Tardy Jobs= ');
  write(' | K8= ');
  write(' | The Fitness Function= ');
  write(' | ');\
  write(' | The Near-Optimal Schedule #2= ');
  write(' | The Maximum Flow Time= ');
  write(' | K1= ');
  write(' | The Mean Flow Time= ');
  write(' | K2= ');
  write(' | The Maximum Tardiness= ');
  write(' | K3= ');
  write(' | The Mean Tardiness= ');
  write(' | K4= ');
  write(' | The Work-In Process Inventory= ');

K5= || The Resource Utilization= \\
write('')
K6= || The Throughput= \\
write('')
K7= || The Number of tardy Jobs= \\
write('')
K8= || The Fitness Function= \\
write('')
write('')

window(20,23,75,23);
textbackground(lightblue);
clrscr;
gotoxy(7,1);
textcolor(yellow);
write('Press ESC to quit in processing !');
gotoxy(40,1);
random_sch(sch); {generate 50 different schedules randomly}

training(sch); {GENETIC ALGORITHM training processes}

while not jump_out do
begin
window(20,23,70,23); textbackground(lightblue);
clrscr; textcolor(yellow);
write('Success ! Press ESC to go back main menu !');
gotoxy(43,1); sound(550); delay(50); nosound;
key_press:=false;
while not key_press do
begin
if keypressed then ch:=readkey;
if ch=#27 then key_press:=true;
end;
out_input:=true; out_output:=true;
exit_flag:=false; jump_out:=true;
end;
end;{output_menu}
Begin{main} {***** Main Program *****}
exit_flag:=false;
while not exit_flag do
begin
    main_manu(exit_flag,out_input,out_output);
    if not out_input then
    begin
        input_manu(out_input,out_output);
        randomize;
        output_manu(out_output,out_input,exit_flag);
    end;
end;
textmode(lastmode);
clrscr;
End.{main}
Appendix II:

Data Description and Listing
JOB DATA

Queue size — 10 1355 — Current time
Previous job type — 4
Job type — 5 1058 — Arrival time
Mean processing time — 12 1122 — Due date
5 1073
12 1142
5 1177
12 1249
5 1212
12 1276
5 1225
12 1303
5 1326
12 1421
3 1349
5 1396
3 1353
5 1399
3 1353
5 1398
7 1355
5 1402

SET-UP TIMES

Current job-type

<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>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Previous job-type 4

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>2</th>
<th>0</th>
<th>4</th>
<th>4</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Group #1</td>
<td>Group #2</td>
<td>Group #3</td>
<td>Group #4</td>
<td>Group #5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 1355</td>
<td>10 1362</td>
<td>10 1367</td>
<td>10 1386</td>
<td>10 1393</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 1058</td>
<td>5 1058</td>
<td>5 1058</td>
<td>5 1058</td>
<td>5 1058</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 1122</td>
<td>12 1122</td>
<td>12 1122</td>
<td>12 1122</td>
<td>12 1122</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 1073</td>
<td>5 1073</td>
<td>5 1073</td>
<td>5 1073</td>
<td>5 1073</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 1142</td>
<td>12 1142</td>
<td>12 1142</td>
<td>12 1142</td>
<td>12 1142</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 1177</td>
<td>5 1177</td>
<td>5 1177</td>
<td>5 1177</td>
<td>5 1177</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 1249</td>
<td>12 1249</td>
<td>12 1249</td>
<td>12 1249</td>
<td>12 1249</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 1212</td>
<td>5 1212</td>
<td>5 1212</td>
<td>5 1212</td>
<td>5 1212</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 1276</td>
<td>12 1276</td>
<td>12 1276</td>
<td>12 1276</td>
<td>12 1276</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 1225</td>
<td>5 1225</td>
<td>5 1225</td>
<td>5 1225</td>
<td>5 1225</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 1303</td>
<td>12 1303</td>
<td>12 1303</td>
<td>12 1303</td>
<td>12 1303</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 1326</td>
<td>5 1326</td>
<td>5 1326</td>
<td>5 1326</td>
<td>5 1326</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 1421</td>
<td>12 1421</td>
<td>12 1421</td>
<td>12 1421</td>
<td>12 1421</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 1349</td>
<td>3 1353</td>
<td>3 1353</td>
<td>7 1363</td>
<td>6 1374</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 1396</td>
<td>5 1399</td>
<td>5 1398</td>
<td>5 1412</td>
<td>6 1435</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 1353</td>
<td>3 1353</td>
<td>7 1355</td>
<td>6 1374</td>
<td>5 1381</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 1399</td>
<td>5 1398</td>
<td>5 1402</td>
<td>6 1435</td>
<td>12 1462</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 1353</td>
<td>7 1355</td>
<td>3 1358</td>
<td>5 1381</td>
<td>3 1383</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 1398</td>
<td>5 1402</td>
<td>5 1402</td>
<td>12 1462</td>
<td>5 1433</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 1355</td>
<td>3 1358</td>
<td>7 1363</td>
<td>3 1383</td>
<td>3 1389</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 1402</td>
<td>5 1402</td>
<td>5 1412</td>
<td>5 1433</td>
<td>5 1435</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group #6</td>
<td>Group #7</td>
<td>Group #8</td>
<td>Group #9</td>
<td>Group #10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>----------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 774</td>
<td>10 791</td>
<td>10 802</td>
<td>10 808</td>
<td>10 815</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 754</td>
<td>6 756</td>
<td>6 756</td>
<td>6 756</td>
<td>6 756</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 800</td>
<td>6 819</td>
<td>6 819</td>
<td>6 819</td>
<td>6 819</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 756</td>
<td>5 758</td>
<td>5 758</td>
<td>5 758</td>
<td>5 758</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 819</td>
<td>12 829</td>
<td>12 829</td>
<td>12 829</td>
<td>12 829</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 758</td>
<td>6 760</td>
<td>6 760</td>
<td>6 760</td>
<td>6 760</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 829</td>
<td>6 823</td>
<td>6 823</td>
<td>6 823</td>
<td>6 823</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 760</td>
<td>7 761</td>
<td>7 761</td>
<td>7 761</td>
<td>5 770</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 823</td>
<td>5 810</td>
<td>5 810</td>
<td>5 810</td>
<td>12 853</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 761</td>
<td>2 769</td>
<td>5 770</td>
<td>5 770</td>
<td>7 780</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 810</td>
<td>4 805</td>
<td>12 853</td>
<td>12 853</td>
<td>5 831</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 763</td>
<td>3 769</td>
<td>7 780</td>
<td>7 780</td>
<td>2 786</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 794</td>
<td>5 809</td>
<td>5 831</td>
<td>5 831</td>
<td>4 820</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 763</td>
<td>5 770</td>
<td>2 786</td>
<td>2 786</td>
<td>1 790</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 801</td>
<td>12 853</td>
<td>4 820</td>
<td>4 820</td>
<td>4 837</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 769</td>
<td>7 780</td>
<td>1 790</td>
<td>1 790</td>
<td>1 797</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 805</td>
<td>5 831</td>
<td>4 837</td>
<td>4 837</td>
<td>4 831</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 769</td>
<td>2 786</td>
<td>4 796</td>
<td>1 797</td>
<td>4 804</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 809</td>
<td>4 820</td>
<td>3 805</td>
<td>4 831</td>
<td>3 829</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 770</td>
<td>1 790</td>
<td>1 797</td>
<td>4 804</td>
<td>1 813</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 853</td>
<td>4 837</td>
<td>4 831</td>
<td>3 829</td>
<td>4 852</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group#11</td>
<td>Group#12</td>
<td>Group#13</td>
<td>Group#14</td>
<td>Group#15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 196</td>
<td>10 203</td>
<td>10 209</td>
<td>10 214</td>
<td>10 221</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 148</td>
<td>7 154</td>
<td>7 154</td>
<td>7 154</td>
<td>7 154</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 190</td>
<td>5 203</td>
<td>5 203</td>
<td>5 203</td>
<td>5 203</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 154</td>
<td>3 154</td>
<td>3 154</td>
<td>3 154</td>
<td>3 160</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 203</td>
<td>5 193</td>
<td>5 193</td>
<td>5 193</td>
<td>5 208</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 154</td>
<td>4 159</td>
<td>3 160</td>
<td>3 160</td>
<td>1 170</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 193</td>
<td>3 194</td>
<td>5 208</td>
<td>5 208</td>
<td>4 202</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 159</td>
<td>3 160</td>
<td>2 166</td>
<td>1 170</td>
<td>3 185</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 194</td>
<td>5 208</td>
<td>4 194</td>
<td>4 202</td>
<td>5 231</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 160</td>
<td>2 166</td>
<td>1 170</td>
<td>3 185</td>
<td>2 186</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 208</td>
<td>4 194</td>
<td>4 202</td>
<td>5 231</td>
<td>4 221</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 166</td>
<td>1 170</td>
<td>3 185</td>
<td>2 186</td>
<td>7 192</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 194</td>
<td>4 202</td>
<td>5 231</td>
<td>4 221</td>
<td>5 243</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 170</td>
<td>3 185</td>
<td>2 186</td>
<td>7 192</td>
<td>7 200</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 202</td>
<td>5 231</td>
<td>4 221</td>
<td>5 243</td>
<td>5 250</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 185</td>
<td>2 186</td>
<td>7 192</td>
<td>7 200</td>
<td>4 208</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 231</td>
<td>4 221</td>
<td>5 243</td>
<td>5 250</td>
<td>3 212</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 186</td>
<td>7 192</td>
<td>7 200</td>
<td>4 208</td>
<td>2 214</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 221</td>
<td>5 243</td>
<td>5 250</td>
<td>3 212</td>
<td>4 256</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 192</td>
<td>7 200</td>
<td>4 208</td>
<td>2 214</td>
<td>1 218</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 243</td>
<td>5 250</td>
<td>3 212</td>
<td>4 256</td>
<td>4 254</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group#16</td>
<td>Group#17</td>
<td>Group#18</td>
<td>Group#19</td>
<td>Group#20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 258</td>
<td>10 263</td>
<td>10 273</td>
<td>10 280</td>
<td>10 285</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 236</td>
<td>6 236</td>
<td>3 238</td>
<td>3 243</td>
<td>1 246</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 247</td>
<td>6 297</td>
<td>5 283</td>
<td>5 282</td>
<td>4 283</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 236</td>
<td>3 238</td>
<td>3 243</td>
<td>1 246</td>
<td>6 249</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 297</td>
<td>5 283</td>
<td>5 282</td>
<td>4 283</td>
<td>6 310</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 238</td>
<td>3 243</td>
<td>1 246</td>
<td>6 249</td>
<td>6 251</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 283</td>
<td>5 282</td>
<td>4 283</td>
<td>6 310</td>
<td>6 310</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 243</td>
<td>1 246</td>
<td>6 249</td>
<td>6 251</td>
<td>2 252</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 282</td>
<td>4 283</td>
<td>6 310</td>
<td>6 310</td>
<td>4 286</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 246</td>
<td>6 249</td>
<td>6 251</td>
<td>2 252</td>
<td>3 257</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 283</td>
<td>6 310</td>
<td>6 310</td>
<td>4 286</td>
<td>5 309</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 249</td>
<td>6 251</td>
<td>2 252</td>
<td>3 257</td>
<td>1 257</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 310</td>
<td>6 310</td>
<td>4 286</td>
<td>5 309</td>
<td>4 299</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 251</td>
<td>2 252</td>
<td>3 257</td>
<td>1 257</td>
<td>3 263</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 310</td>
<td>4 286</td>
<td>5 309</td>
<td>4 299</td>
<td>5 313</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 252</td>
<td>3 257</td>
<td>1 257</td>
<td>3 263</td>
<td>7 265</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 286</td>
<td>5 309</td>
<td>4 299</td>
<td>5 313</td>
<td>5 313</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 257</td>
<td>1 257</td>
<td>3 263</td>
<td>7 265</td>
<td>3 277</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 309</td>
<td>4 299</td>
<td>5 313</td>
<td>5 313</td>
<td>5 325</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 257</td>
<td>3 263</td>
<td>7 265</td>
<td>3 277</td>
<td>6 281</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 299</td>
<td>5 313</td>
<td>5 313</td>
<td>5 325</td>
<td>6 340</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group#21</td>
<td>Group#22</td>
<td>Group#23</td>
<td>Group#24</td>
<td>Group#25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 432</td>
<td>10 444</td>
<td>10 453</td>
<td>10 467</td>
<td>10 476</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 380</td>
<td>6 389</td>
<td>5 393</td>
<td>6 398</td>
<td>5 399</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 427</td>
<td>6 451</td>
<td>12 464</td>
<td>6 461</td>
<td>12 469</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 389</td>
<td>5 393</td>
<td>6 398</td>
<td>5 399</td>
<td>6 415</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 451</td>
<td>12 464</td>
<td>6 461</td>
<td>12 469</td>
<td>6 475</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 393</td>
<td>6 398</td>
<td>5 399</td>
<td>6 415</td>
<td>4 427</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 464</td>
<td>6 461</td>
<td>12 469</td>
<td>6 475</td>
<td>3 479</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 398</td>
<td>5 399</td>
<td>6 415</td>
<td>4 427</td>
<td>7 429</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 461</td>
<td>12 469</td>
<td>6 475</td>
<td>3 479</td>
<td>5 480</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 399</td>
<td>6 415</td>
<td>4 427</td>
<td>7 429</td>
<td>3 429</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 469</td>
<td>6 475</td>
<td>3 479</td>
<td>5 480</td>
<td>5 477</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 407</td>
<td>4 427</td>
<td>7 429</td>
<td>3 429</td>
<td>3 440</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 441</td>
<td>3 479</td>
<td>5 480</td>
<td>5 477</td>
<td>5 485</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 415</td>
<td>7 429</td>
<td>3 429</td>
<td>3 440</td>
<td>6 444</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 475</td>
<td>5 480</td>
<td>5 477</td>
<td>5 485</td>
<td>5 477</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 427</td>
<td>3 429</td>
<td>3 440</td>
<td>6 444</td>
<td>5 452</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 479</td>
<td>5 477</td>
<td>5 485</td>
<td>6 503</td>
<td>12 534</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 429</td>
<td>3 440</td>
<td>6 444</td>
<td>5 452</td>
<td>2 466</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 480</td>
<td>5 485</td>
<td>6 503</td>
<td>12 534</td>
<td>4 494</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 429</td>
<td>6 444</td>
<td>5 452</td>
<td>2 466</td>
<td>1 469</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 477</td>
<td>6 503</td>
<td>12 534</td>
<td>4 494</td>
<td>4 500</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix III:
Distributions for All Performance Measures

(25 problems)
<table>
<thead>
<tr>
<th>Data Group #1</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Flow Time</td>
<td>* Mean Flow Time</td>
<td>* Maximum Tardiness</td>
<td></td>
</tr>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
<td>Numbers</td>
</tr>
<tr>
<td>313.000-323.200</td>
<td>109800</td>
<td>153.700-155.700</td>
<td>8640</td>
</tr>
<tr>
<td>323.200-333.400</td>
<td>172800</td>
<td>155.700-157.700</td>
<td>21800</td>
</tr>
<tr>
<td>333.400-343.800</td>
<td>233588</td>
<td>157.700-158.700</td>
<td>56180</td>
</tr>
<tr>
<td>343.800-353.800</td>
<td>326304</td>
<td>159.700-161.700</td>
<td>103680</td>
</tr>
<tr>
<td>353.800-364.000</td>
<td>354240</td>
<td>161.700-163.700</td>
<td>354240</td>
</tr>
<tr>
<td>364.000-374.200</td>
<td>527760</td>
<td>163.700-165.700</td>
<td>358560</td>
</tr>
<tr>
<td>374.200-384.400</td>
<td>540144</td>
<td>165.700-167.700</td>
<td>804800</td>
</tr>
<tr>
<td>384.400-394.600</td>
<td>640856</td>
<td>167.700-169.700</td>
<td>881280</td>
</tr>
<tr>
<td>394.600-404.800</td>
<td>399744</td>
<td>169.700-171.700</td>
<td>630720</td>
</tr>
<tr>
<td>404.800-415.000</td>
<td>302400</td>
<td>171.700-173.700</td>
<td>809120</td>
</tr>
</tbody>
</table>

**Mean = 372.800**

**Variance = 578.885**

**Std.Dev. = 24.056**

<table>
<thead>
<tr>
<th>Mean Tardiness</th>
<th>WIP Inventory</th>
<th>Resource Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>100.300-102.080</td>
<td>109800</td>
<td>4.776- 4.942</td>
</tr>
<tr>
<td>102.080-103.820</td>
<td>58080</td>
<td>4.942- 5.109</td>
</tr>
<tr>
<td>103.820-105.580</td>
<td>184800</td>
<td>5.109- 5.275</td>
</tr>
<tr>
<td>105.580-107.340</td>
<td>476840</td>
<td>5.276- 5.442</td>
</tr>
<tr>
<td>107.340-109.100</td>
<td>788840</td>
<td>5.442- 5.608</td>
</tr>
<tr>
<td>109.100-110.860</td>
<td>803400</td>
<td>5.608- 5.774</td>
</tr>
<tr>
<td>110.860-112.620</td>
<td>691080</td>
<td>5.774- 5.941</td>
</tr>
<tr>
<td>112.620-114.380</td>
<td>306000</td>
<td>5.941- 6.107</td>
</tr>
<tr>
<td>114.380-116.140</td>
<td>183680</td>
<td>6.107- 6.274</td>
</tr>
<tr>
<td>116.140-117.900</td>
<td>50880</td>
<td>6.274- 6.440</td>
</tr>
</tbody>
</table>

**Mean = 109.773**

**Variance = 8.184**

**Std.Dev. = 2.857**

<table>
<thead>
<tr>
<th>Throughput</th>
<th>No. of Tardy jobs</th>
<th>Summation of *</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>0.085- 0.087</td>
<td>259200</td>
<td>5.000- 5.500</td>
</tr>
<tr>
<td>0.087- 0.088</td>
<td>432000</td>
<td>5.500- 6.000</td>
</tr>
<tr>
<td>0.088- 0.090</td>
<td>691200</td>
<td>6.000- 6.500</td>
</tr>
<tr>
<td>0.090- 0.092</td>
<td>604800</td>
<td>6.500- 7.000</td>
</tr>
<tr>
<td>0.092- 0.095</td>
<td>777600</td>
<td>7.000- 7.500</td>
</tr>
<tr>
<td>0.094- 0.095</td>
<td>432000</td>
<td>7.500- 8.000</td>
</tr>
<tr>
<td>0.095- 0.097</td>
<td>267840</td>
<td>8.000- 8.500</td>
</tr>
<tr>
<td>0.097- 0.099</td>
<td>129960</td>
<td>8.500- 9.000</td>
</tr>
<tr>
<td>0.099- 0.100</td>
<td>25920</td>
<td>9.000- 9.500</td>
</tr>
<tr>
<td>0.100- 0.102</td>
<td>8640</td>
<td>9.500- 10.000</td>
</tr>
</tbody>
</table>

**Mean = 0.091**

**Variance = 0.000**

**Std.Dev. = 0.003**
<table>
<thead>
<tr>
<th>Data Group #2</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Maximum Flow Time</td>
<td>* Mean Flow Time</td>
<td>* Maximum Tardiness</td>
<td></td>
</tr>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>320.000-330.200</td>
<td>107280</td>
<td>157.800-160.000</td>
<td>4320</td>
<td>256.000-266.200</td>
</tr>
<tr>
<td>330.200-340.400</td>
<td>17224</td>
<td>180.000-182.200</td>
<td>38880</td>
<td>288.200-276.400</td>
</tr>
<tr>
<td>340.400-350.800</td>
<td>258320</td>
<td>162.200-164.400</td>
<td>47520</td>
<td>278.400-286.800</td>
</tr>
<tr>
<td>350.800-360.800</td>
<td>322272</td>
<td>164.400-166.800</td>
<td>164160</td>
<td>286.800-289.800</td>
</tr>
<tr>
<td>360.800-371.000</td>
<td>390672</td>
<td>168.600-168.800</td>
<td>285120</td>
<td>298.800-307.000</td>
</tr>
<tr>
<td>371.000-381.200</td>
<td>528464</td>
<td>168.800-171.000</td>
<td>328320</td>
<td>307.000-317.200</td>
</tr>
<tr>
<td>381.200-391.400</td>
<td>546336</td>
<td>171.000-173.200</td>
<td>777800</td>
<td>317.200-327.400</td>
</tr>
<tr>
<td>391.400-401.800</td>
<td>653040</td>
<td>173.200-176.400</td>
<td>658840</td>
<td>327.400-337.800</td>
</tr>
<tr>
<td>401.800-411.800</td>
<td>397792</td>
<td>175.400-177.800</td>
<td>658840</td>
<td>337.800-347.800</td>
</tr>
<tr>
<td>411.800-422.000</td>
<td>286400</td>
<td>177.800-179.800</td>
<td>669800</td>
<td>347.800-358.000</td>
</tr>
<tr>
<td>Mean = 379.000</td>
<td>Mean = 173.400</td>
<td>Mean = 313.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variance = 577.911</td>
<td>Variance = 17.120</td>
<td>Variance = 800.467</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Std.Dev. = 24.040</td>
<td>Std.Dev. = 4.138</td>
<td>Std.Dev. = 24.504</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean Tardiness</td>
<td>WIP Inventory</td>
<td>Resource Utilization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>105.500-107.270</td>
<td>28400</td>
<td>4.687-4.844</td>
<td>43200</td>
<td>0.780-0.798</td>
</tr>
<tr>
<td>107.270-109.040</td>
<td>111360</td>
<td>4.844-5.022</td>
<td>125280</td>
<td>0.798-0.816</td>
</tr>
<tr>
<td>109.040-110.810</td>
<td>284400</td>
<td>5.022-5.199</td>
<td>276480</td>
<td>0.816-0.833</td>
</tr>
<tr>
<td>110.810-112.580</td>
<td>495280</td>
<td>5.199-5.376</td>
<td>583200</td>
<td>0.833-0.851</td>
</tr>
<tr>
<td>112.580-114.350</td>
<td>790800</td>
<td>5.376-5.554</td>
<td>864000</td>
<td>0.851-0.869</td>
</tr>
<tr>
<td>114.350-116.120</td>
<td>773760</td>
<td>5.554-5.731</td>
<td>758000</td>
<td>0.869-0.887</td>
</tr>
<tr>
<td>116.120-117.890</td>
<td>622800</td>
<td>5.731-5.908</td>
<td>548640</td>
<td>0.887-0.905</td>
</tr>
<tr>
<td>117.890-119.660</td>
<td>348480</td>
<td>5.808-6.085</td>
<td>311040</td>
<td>0.905-0.922</td>
</tr>
<tr>
<td>119.660-121.430</td>
<td>140160</td>
<td>6.085-6.263</td>
<td>90720</td>
<td>0.922-0.940</td>
</tr>
<tr>
<td>121.430-123.200</td>
<td>45120</td>
<td>6.263-6.440</td>
<td>30240</td>
<td>0.940-0.958</td>
</tr>
<tr>
<td>Mean = 114.807</td>
<td>Mean = 5.540</td>
<td>Mean = 0.845</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variance = 9.481</td>
<td>Variance = 0.089</td>
<td>Variance = 0.001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Std.Dev. = 3.079</td>
<td>Std.Dev. = 0.289</td>
<td>Std.Dev. = 0.032</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Throughput</td>
<td>No. of Tardy jobs</td>
<td>Summation of *</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>0.085-0.087</td>
<td>259200</td>
<td>5.000-5.500</td>
<td>14400</td>
<td>424.200-435.500</td>
</tr>
<tr>
<td>0.087-0.089</td>
<td>302400</td>
<td>5.500-6.000</td>
<td>0</td>
<td>435.500-446.800</td>
</tr>
<tr>
<td>0.089-0.091</td>
<td>691200</td>
<td>6.000-6.500</td>
<td>167040</td>
<td>448.800-458.100</td>
</tr>
<tr>
<td>0.091-0.093</td>
<td>1231200</td>
<td>6.500-7.000</td>
<td>0</td>
<td>458.100-469.400</td>
</tr>
<tr>
<td>0.093-0.095</td>
<td>583200</td>
<td>7.000-7.500</td>
<td>664320</td>
<td>469.400-480.700</td>
</tr>
<tr>
<td>0.095-0.098</td>
<td>259200</td>
<td>7.500-8.000</td>
<td>0</td>
<td>480.700-492.000</td>
</tr>
<tr>
<td>0.096-0.098</td>
<td>220320</td>
<td>8.000-8.500</td>
<td>1323840</td>
<td>492.000-503.300</td>
</tr>
<tr>
<td>0.098-0.100</td>
<td>604800</td>
<td>8.500-9.000</td>
<td>0</td>
<td>503.300-514.800</td>
</tr>
<tr>
<td>0.100-0.102</td>
<td>172800</td>
<td>9.000-9.500</td>
<td>1200000</td>
<td>514.800-525.900</td>
</tr>
<tr>
<td>0.102-0.104</td>
<td>4320</td>
<td>9.500-10.000</td>
<td>259200</td>
<td>526.900-537.200</td>
</tr>
<tr>
<td>Mean = 0.082</td>
<td>Mean = 8.187</td>
<td>Mean = 488.400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variance = 0.000</td>
<td>Variance = 0.985</td>
<td>Variance = 804.873</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Std.Dev. = 0.003</td>
<td>Std.Dev. = 0.982</td>
<td>Std.Dev. = 24.584</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Group #3</td>
<td>Maximum Flow Time</td>
<td>* Mean Flow Time</td>
<td>* Maximum Tardiness</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-------------------</td>
<td>------------------</td>
<td>-------------------</td>
<td></td>
</tr>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>325.000-335.000</td>
<td>100800</td>
<td>182.000-184.150</td>
<td>2880</td>
<td>281.000-271.000</td>
</tr>
<tr>
<td>335.000-345.000</td>
<td>158400</td>
<td>184.150-186.300</td>
<td>25820</td>
<td>271.000-281.000</td>
</tr>
<tr>
<td>345.000-355.000</td>
<td>254304</td>
<td>186.300-188.450</td>
<td>48080</td>
<td>281.000-291.000</td>
</tr>
<tr>
<td>355.000-365.000</td>
<td>309024</td>
<td>188.450-190.600</td>
<td>175880</td>
<td>291.000-301.000</td>
</tr>
<tr>
<td>365.000-375.000</td>
<td>405312</td>
<td>190.600-192.750</td>
<td>253440</td>
<td>301.000-311.000</td>
</tr>
<tr>
<td>375.000-385.000</td>
<td>478388</td>
<td>192.750-194.900</td>
<td>403200</td>
<td>311.000-321.000</td>
</tr>
<tr>
<td>385.000-395.000</td>
<td>526368</td>
<td>194.900-197.050</td>
<td>792000</td>
<td>321.000-331.000</td>
</tr>
<tr>
<td>395.000-405.000</td>
<td>641668</td>
<td>197.050-200.000</td>
<td>578880</td>
<td>331.000-341.000</td>
</tr>
<tr>
<td>405.000-415.000</td>
<td>423456</td>
<td>170.200-181.350</td>
<td>774720</td>
<td>341.000-351.000</td>
</tr>
<tr>
<td>415.000-425.000</td>
<td>331200</td>
<td>181.350-183.500</td>
<td>576000</td>
<td>351.000-361.000</td>
</tr>
</tbody>
</table>

Mean = 383.533  Mean = 177.200  Mean = 317.633
Variance = 567.508  Variance = 15.208  Variance = 590.308
Std.Dev. = 23.822  Std.Dev. = 3.899  Std.Dev. = 24.298

<table>
<thead>
<tr>
<th>Mean Tardiness</th>
<th>WIP Inventory</th>
<th>Resource Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>108.500-110.200</td>
<td>15720</td>
<td>4.888- 4.881</td>
</tr>
<tr>
<td>110.200-111.900</td>
<td>68400</td>
<td>4.881- 5.034</td>
</tr>
<tr>
<td>111.900-113.600</td>
<td>213000</td>
<td>5.034- 5.208</td>
</tr>
<tr>
<td>113.600-115.300</td>
<td>443400</td>
<td>5.208- 5.381</td>
</tr>
<tr>
<td>115.300-117.000</td>
<td>720840</td>
<td>5.381- 5.554</td>
</tr>
<tr>
<td>117.000-118.700</td>
<td>844080</td>
<td>5.554- 5.727</td>
</tr>
<tr>
<td>118.700-120.400</td>
<td>887840</td>
<td>5.727- 5.900</td>
</tr>
<tr>
<td>120.400-122.100</td>
<td>409680</td>
<td>5.900- 6.074</td>
</tr>
<tr>
<td>122.100-123.800</td>
<td>170160</td>
<td>6.074- 6.247</td>
</tr>
<tr>
<td>123.800-125.500</td>
<td>55680</td>
<td>6.247- 6.420</td>
</tr>
</tbody>
</table>

Mean = 117.658  Mean = 5.552  Mean = 0.850
Variance = 8.229  Variance = 0.082  Variance = 0.001
Std.Dev. = 2.889  Std.Dev. = 0.287  Std.Dev. = 0.029

<table>
<thead>
<tr>
<th>Throughput</th>
<th>No. of Tardy jobs</th>
<th>Summation of *</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>0.088- 0.088</td>
<td>345600</td>
<td>5.000- 5.500</td>
</tr>
<tr>
<td>0.088- 0.090</td>
<td>518400</td>
<td>5.500- 6.000</td>
</tr>
<tr>
<td>0.090- 0.091</td>
<td>806400</td>
<td>6.000- 6.500</td>
</tr>
<tr>
<td>0.091- 0.093</td>
<td>678600</td>
<td>6.500- 7.000</td>
</tr>
<tr>
<td>0.093- 0.095</td>
<td>678600</td>
<td>7.000- 7.500</td>
</tr>
<tr>
<td>0.095- 0.097</td>
<td>345600</td>
<td>7.500- 8.000</td>
</tr>
<tr>
<td>0.097- 0.099</td>
<td>181440</td>
<td>8.000- 8.500</td>
</tr>
<tr>
<td>0.099- 0.100</td>
<td>83380</td>
<td>8.500- 9.000</td>
</tr>
<tr>
<td>0.100- 0.102</td>
<td>11520</td>
<td>9.000- 9.500</td>
</tr>
<tr>
<td>0.102- 0.104</td>
<td>2880</td>
<td>9.500-10.000</td>
</tr>
</tbody>
</table>

Mean = 0.092  Mean = 8.293  Mean = 494.733
Variance = 0.000  Variance = 0.947  Variance = 589.986
Std.Dev. = 0.003  Std.Dev. = 0.873  Std.Dev. = 24.289
<table>
<thead>
<tr>
<th>Data Group #4</th>
<th>Maximum Flow Time</th>
<th>Mean Flow Time</th>
<th>Maximum Tardiness</th>
<th>Intervals</th>
<th>Numbers</th>
<th>Intervals</th>
<th>Numbers</th>
<th>Intervals</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>344.000-354.400</td>
<td>96480</td>
<td>182.200-183.710</td>
<td>10080</td>
<td>280.000-290.400</td>
<td>109440</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>354.400-364.800</td>
<td>137280</td>
<td>183.710-185.220</td>
<td>55440</td>
<td>290.400-300.800</td>
<td>174120</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>364.800-375.200</td>
<td>237380</td>
<td>185.220-186.730</td>
<td>30240</td>
<td>300.800-311.200</td>
<td>258360</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>375.200-385.600</td>
<td>306000</td>
<td>186.730-188.240</td>
<td>131040</td>
<td>311.200-321.600</td>
<td>330800</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>385.600-396.000</td>
<td>343920</td>
<td>188.240-189.750</td>
<td>267120</td>
<td>321.600-332.000</td>
<td>355320</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>396.000-406.400</td>
<td>478080</td>
<td>189.750-191.260</td>
<td>418320</td>
<td>332.000-342.400</td>
<td>481320</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>406.400-416.800</td>
<td>468880</td>
<td>191.260-192.770</td>
<td>322580</td>
<td>342.400-352.800</td>
<td>530880</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>416.800-427.200</td>
<td>615000</td>
<td>192.770-194.280</td>
<td>524160</td>
<td>352.800-363.200</td>
<td>590280</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>427.200-437.600</td>
<td>573480</td>
<td>194.280-195.790</td>
<td>972720</td>
<td>363.200-373.600</td>
<td>416160</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>437.600-448.000</td>
<td>382320</td>
<td>195.790-197.300</td>
<td>897120</td>
<td>373.600-384.000</td>
<td>382320</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean= 407.533</td>
<td>Mean = 193.400</td>
<td>Mean = 341.533</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variance= 657.871</td>
<td>Variance = 8.819</td>
<td>Variance = 679.782</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Std.Dev. = 25.649</td>
<td>Std.Dev. = 2.970</td>
<td>Std.Dev. = 26.073</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mean Tardiness</th>
<th>WIP Inventory</th>
<th>Resource Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>121.900-123.450</td>
<td>51000</td>
<td>4.991- 5.103</td>
</tr>
<tr>
<td>123.450-125.000</td>
<td>179820</td>
<td>5.103- 5.215</td>
</tr>
<tr>
<td>125.000-128.550</td>
<td>442080</td>
<td>5.215- 5.326</td>
</tr>
<tr>
<td>128.550-128.100</td>
<td>630000</td>
<td>5.326- 5.438</td>
</tr>
<tr>
<td>128.100-128.850</td>
<td>734400</td>
<td>5.438- 5.550</td>
</tr>
<tr>
<td>128.850-131.200</td>
<td>611880</td>
<td>5.550- 5.662</td>
</tr>
<tr>
<td>131.200-132.750</td>
<td>473280</td>
<td>5.662- 5.774</td>
</tr>
<tr>
<td>132.750-134.300</td>
<td>320280</td>
<td>5.774- 5.885</td>
</tr>
<tr>
<td>134.300-135.850</td>
<td>138120</td>
<td>5.885- 5.997</td>
</tr>
<tr>
<td>135.850-137.400</td>
<td>48240</td>
<td>5.997- 6.109</td>
</tr>
<tr>
<td>Mean= 129.289</td>
<td>Mean = 5.573</td>
<td>Mean = 0.863</td>
</tr>
<tr>
<td>Variance = 8.747</td>
<td>Variance = 0.038</td>
<td>Variance = 0.000</td>
</tr>
<tr>
<td>Std.Dev. = 2.858</td>
<td>Std.Dev. = 0.186</td>
<td>Std.Dev. = 0.021</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Throughput</th>
<th>No. of Tardy jobs</th>
<th>Summation of *</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>0.083-0.084</td>
<td>604800</td>
<td>5.000- 5.500</td>
</tr>
<tr>
<td>0.084-0.085</td>
<td>453600</td>
<td>5.500- 6.000</td>
</tr>
<tr>
<td>0.085-0.086</td>
<td>453600</td>
<td>6.000- 6.500</td>
</tr>
<tr>
<td>0.086-0.087</td>
<td>878980</td>
<td>6.500- 7.000</td>
</tr>
<tr>
<td>0.087-0.088</td>
<td>514080</td>
<td>7.000- 7.500</td>
</tr>
<tr>
<td>0.088-0.089</td>
<td>302400</td>
<td>7.500- 8.000</td>
</tr>
<tr>
<td>0.089-0.090</td>
<td>198580</td>
<td>8.000- 8.500</td>
</tr>
<tr>
<td>0.090-0.091</td>
<td>181440</td>
<td>8.500- 9.000</td>
</tr>
<tr>
<td>0.091-0.092</td>
<td>20160</td>
<td>9.000- 9.500</td>
</tr>
<tr>
<td>0.092-0.093</td>
<td>25200</td>
<td>9.500-10.000</td>
</tr>
<tr>
<td>Mean= 0.086</td>
<td>Mean = 8.163</td>
<td>Mean = 534.933</td>
</tr>
<tr>
<td>Variance = 0.000</td>
<td>Variance = 0.799</td>
<td>Variance = 679.921</td>
</tr>
<tr>
<td>Std.Dev. = 0.002</td>
<td>Std.Dev. = 0.894</td>
<td>Std.Dev. = 26.075</td>
</tr>
</tbody>
</table>
### Data Group #5

<table>
<thead>
<tr>
<th>Maximum Flow Time</th>
<th>* Mean Flow Time</th>
<th>* Maximum Tardiness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>349.000-359.800</td>
<td>95780</td>
<td>187.800-188.810</td>
</tr>
<tr>
<td>359.800-370.200</td>
<td>178640</td>
<td>188.810-190.020</td>
</tr>
<tr>
<td>370.200-380.800</td>
<td>140040</td>
<td>190.020-191.230</td>
</tr>
<tr>
<td>380.800-391.400</td>
<td>316900</td>
<td>191.230-192.440</td>
</tr>
<tr>
<td>391.400-402.000</td>
<td>354480</td>
<td>192.440-193.650</td>
</tr>
<tr>
<td>402.000-412.600</td>
<td>445680</td>
<td>193.650-194.860</td>
</tr>
<tr>
<td>412.600-423.200</td>
<td>543120</td>
<td>194.860-196.070</td>
</tr>
<tr>
<td>423.200-433.800</td>
<td>549840</td>
<td>196.070-197.280</td>
</tr>
<tr>
<td>433.800-444.400</td>
<td>637440</td>
<td>197.280-198.490</td>
</tr>
<tr>
<td>444.400-455.000</td>
<td>360000</td>
<td>198.490-199.700</td>
</tr>
<tr>
<td>Mean = 414.033</td>
<td>Mean = 197.000</td>
<td>Mean = 348.033</td>
</tr>
<tr>
<td>Variance = 874.236</td>
<td>Variance = 4.964</td>
<td>Variance = 995.981</td>
</tr>
<tr>
<td>Std.Dev. = 25.866</td>
<td>Std.Dev. = 2.228</td>
<td>Std.Dev. = 26.381</td>
</tr>
</tbody>
</table>

### Mean Tardiness

<table>
<thead>
<tr>
<th>Intervals</th>
<th>Numbers</th>
<th>Intervals</th>
<th>Numbers</th>
<th>Intervals</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>125.100-126.500</td>
<td>24240</td>
<td>5.037-5.144</td>
<td>90720</td>
<td>0.833-0.843</td>
<td>804800</td>
</tr>
<tr>
<td>126.500-127.900</td>
<td>80640</td>
<td>5.144-5.252</td>
<td>80840</td>
<td>0.843-0.853</td>
<td>302400</td>
</tr>
<tr>
<td>127.900-129.300</td>
<td>202800</td>
<td>5.252-5.359</td>
<td>332640</td>
<td>0.853-0.864</td>
<td>1209600</td>
</tr>
<tr>
<td>129.300-130.700</td>
<td>342240</td>
<td>5.359-5.487</td>
<td>927380</td>
<td>0.864-0.874</td>
<td>1209600</td>
</tr>
<tr>
<td>130.700-132.100</td>
<td>822840</td>
<td>5.467-5.575</td>
<td>907200</td>
<td>0.874-0.884</td>
<td>544320</td>
</tr>
<tr>
<td>132.100-133.500</td>
<td>865320</td>
<td>5.576-5.682</td>
<td>604900</td>
<td>0.884-0.894</td>
<td>423360</td>
</tr>
<tr>
<td>133.500-134.900</td>
<td>537000</td>
<td>5.682-5.789</td>
<td>433440</td>
<td>0.894-0.904</td>
<td>211680</td>
</tr>
<tr>
<td>134.900-136.300</td>
<td>383400</td>
<td>5.789-5.897</td>
<td>161280</td>
<td>0.904-0.915</td>
<td>141120</td>
</tr>
<tr>
<td>136.300-137.700</td>
<td>285720</td>
<td>5.897-6.005</td>
<td>70580</td>
<td>0.915-0.925</td>
<td>20180</td>
</tr>
<tr>
<td>137.700-139.100</td>
<td>84800</td>
<td>6.005-6.112</td>
<td>20180</td>
<td>0.925-0.935</td>
<td>50400</td>
</tr>
<tr>
<td>Mean = 132.784</td>
<td>Mean = 6.520</td>
<td>Mean = 0.888</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variance = 6.312</td>
<td>Variance = 0.030</td>
<td>Variance = 0.001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Std.Dev. = 2.512</td>
<td>Std.Dev. = 0.173</td>
<td>Std.Dev. = 0.023</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Throughput

<table>
<thead>
<tr>
<th>Intervals</th>
<th>Numbers</th>
<th>Intervals</th>
<th>Numbers</th>
<th>Summation of *</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.084-0.084</td>
<td>804800</td>
<td>5.000-5.500</td>
<td>3380</td>
<td>477.000-488.350</td>
</tr>
<tr>
<td>0.084-0.085</td>
<td>302400</td>
<td>5.500-6.000</td>
<td>0</td>
<td>488.350-499.700</td>
</tr>
<tr>
<td>0.085-0.086</td>
<td>804800</td>
<td>6.000-6.500</td>
<td>98880</td>
<td>499.700-511.050</td>
</tr>
<tr>
<td>0.086-0.087</td>
<td>725760</td>
<td>6.500-7.000</td>
<td>0</td>
<td>511.050-522.400</td>
</tr>
<tr>
<td>0.087-0.088</td>
<td>544320</td>
<td>7.000-7.500</td>
<td>581280</td>
<td>522.400-533.750</td>
</tr>
<tr>
<td>0.088-0.089</td>
<td>302400</td>
<td>7.500-8.000</td>
<td>0</td>
<td>533.750-545.100</td>
</tr>
<tr>
<td>0.089-0.090</td>
<td>120960</td>
<td>8.000-8.500</td>
<td>1389600</td>
<td>545.100-556.450</td>
</tr>
<tr>
<td>0.090-0.091</td>
<td>352800</td>
<td>8.500-9.000</td>
<td>0</td>
<td>556.450-567.800</td>
</tr>
<tr>
<td>0.091-0.092</td>
<td>20160</td>
<td>9.000-9.500</td>
<td>1275360</td>
<td>567.800-579.150</td>
</tr>
<tr>
<td>0.092-0.093</td>
<td>50400</td>
<td>9.500-10.000</td>
<td>280320</td>
<td>579.150-590.500</td>
</tr>
<tr>
<td>Mean = 0.087</td>
<td>Mean = 8.288</td>
<td>Mean = 0.856</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variance = 0.000</td>
<td>Variance = 0.856</td>
<td>Variance = 692.114</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Std.Dev. = 0.002</td>
<td>Std.Dev. = 0.925</td>
<td>Std.Dev. = 26.308</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Group #6</td>
<td>Maximum Flow Time</td>
<td>* Mean Flow Time</td>
<td>* Maximum Tardiness</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------</td>
<td>------------------</td>
<td>---------------------</td>
<td></td>
</tr>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
<td>Numbers</td>
<td></td>
</tr>
<tr>
<td>77.000-80.100</td>
<td>2988</td>
<td>43.900-46.270</td>
<td>1312</td>
<td>6.000-12.300</td>
</tr>
<tr>
<td>80.100-83.200</td>
<td>38376</td>
<td>46.270-48.840</td>
<td>28736</td>
<td>12.300-18.800</td>
</tr>
<tr>
<td>83.200-86.300</td>
<td>190686</td>
<td>48.840-51.010</td>
<td>164160</td>
<td>18.800-24.900</td>
</tr>
<tr>
<td>86.300-89.400</td>
<td>421580</td>
<td>51.010-53.380</td>
<td>399920</td>
<td>24.900-31.200</td>
</tr>
<tr>
<td>89.400-92.500</td>
<td>423628</td>
<td>53.380-56.750</td>
<td>677808</td>
<td>31.200-37.500</td>
</tr>
<tr>
<td>92.500-95.600</td>
<td>614264</td>
<td>55.750-58.120</td>
<td>870480</td>
<td>37.500-43.800</td>
</tr>
<tr>
<td>95.600-98.700</td>
<td>830200</td>
<td>58.120-60.490</td>
<td>784872</td>
<td>43.800-50.100</td>
</tr>
<tr>
<td>98.700-101.800</td>
<td>643216</td>
<td>60.490-62.860</td>
<td>522464</td>
<td>50.100-56.400</td>
</tr>
<tr>
<td>101.800-104.900</td>
<td>385528</td>
<td>62.860-65.230</td>
<td>180304</td>
<td>56.400-62.700</td>
</tr>
<tr>
<td>104.900-108.000</td>
<td>78200</td>
<td>65.230-67.800</td>
<td>189444</td>
<td>62.700-69.000</td>
</tr>
<tr>
<td>Mean = 95.308</td>
<td>Mean = 57.150</td>
<td>Mean = 48.471</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variance = 28.920</td>
<td>Variance = 13.416</td>
<td>Variance = 99.197</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Std.Dev. = 5.378</td>
<td>Std.Dev. = 3.883</td>
<td>Std.Dev. = 9.880</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mean Tardiness</th>
<th>WIP Inventory</th>
<th>Resource Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>0.600-3.150</td>
<td>3681</td>
<td>4.300-4.537</td>
</tr>
<tr>
<td>3.150-5.700</td>
<td>46004</td>
<td>4.637-4.769</td>
</tr>
<tr>
<td>5.700-8.250</td>
<td>191147</td>
<td>4.769-5.002</td>
</tr>
<tr>
<td>8.250-10.800</td>
<td>396253</td>
<td>5.002-5.234</td>
</tr>
<tr>
<td>10.800-13.350</td>
<td>683227</td>
<td>5.234-5.487</td>
</tr>
<tr>
<td>13.350-15.900</td>
<td>885612</td>
<td>5.467-5.700</td>
</tr>
<tr>
<td>15.900-18.450</td>
<td>780687</td>
<td>5.700-5.932</td>
</tr>
<tr>
<td>18.450-21.000</td>
<td>499596</td>
<td>5.932-6.165</td>
</tr>
<tr>
<td>21.000-23.550</td>
<td>159716</td>
<td>6.165-6.397</td>
</tr>
<tr>
<td>23.550-28.100</td>
<td>12876</td>
<td>6.397-6.830</td>
</tr>
<tr>
<td>Mean = 14.644</td>
<td>Mean = 5.482</td>
<td>Mean = 0.781</td>
</tr>
<tr>
<td>Variance = 16.082</td>
<td>Variance = 0.168</td>
<td>Variance = 0.000</td>
</tr>
<tr>
<td>Std.Dev. = 4.008</td>
<td>Std.Dev. = 0.410</td>
<td>Std.Dev. = 0.022</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Throughput</th>
<th>No. of Tardy jobs</th>
<th>Summation of *</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>0.112-0.115</td>
<td>8880</td>
<td>1.000-1.700</td>
</tr>
<tr>
<td>0.115-0.117</td>
<td>146400</td>
<td>1.700-2.400</td>
</tr>
<tr>
<td>0.117-0.119</td>
<td>1238944</td>
<td>2.400-3.100</td>
</tr>
<tr>
<td>0.119-0.122</td>
<td>1013104</td>
<td>3.100-3.800</td>
</tr>
<tr>
<td>0.122-0.125</td>
<td>384928</td>
<td>3.800-4.500</td>
</tr>
<tr>
<td>0.125-0.127</td>
<td>391472</td>
<td>4.500-5.200</td>
</tr>
<tr>
<td>0.127-0.130</td>
<td>87936</td>
<td>5.200-5.900</td>
</tr>
<tr>
<td>0.130-0.132</td>
<td>64480</td>
<td>5.900-6.600</td>
</tr>
<tr>
<td>0.132-0.134</td>
<td>8424</td>
<td>6.600-7.300</td>
</tr>
<tr>
<td>0.134-0.137</td>
<td>3232</td>
<td>7.300-8.000</td>
</tr>
<tr>
<td>Mean = 0.121</td>
<td>Mean = 5.470</td>
<td>Mean = 105.621</td>
</tr>
<tr>
<td>Variance = 0.000</td>
<td>Variance = 1.243</td>
<td>Variance = 153.765</td>
</tr>
<tr>
<td>Std.Dev. = 0.003</td>
<td>Std.Dev. = 1.115</td>
<td>Std.Dev. = 12.400</td>
</tr>
</tbody>
</table>
### Data Group #7

<table>
<thead>
<tr>
<th>Intervals</th>
<th>Numbers</th>
<th>* Mean Flow Time</th>
<th>* Maximum Tardiness</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>81,000-85,400</td>
<td>722</td>
<td>53,800-55,940</td>
<td>1440</td>
<td>19,000-24,700</td>
</tr>
<tr>
<td>85,400-89,800</td>
<td>6988</td>
<td>55,940-58,280</td>
<td>24448</td>
<td>24,700-30,400</td>
</tr>
<tr>
<td>89,800-94,200</td>
<td>56188</td>
<td>58,280-60,620</td>
<td>151744</td>
<td>30,400-36,100</td>
</tr>
<tr>
<td>94,200-98,600</td>
<td>156324</td>
<td>60,620-62,960</td>
<td>380736</td>
<td>36,100-41,800</td>
</tr>
<tr>
<td>98,600-103,000</td>
<td>360786</td>
<td>62,960-65,300</td>
<td>633504</td>
<td>41,800-47,500</td>
</tr>
<tr>
<td>103,000-107,400</td>
<td>874916</td>
<td>65,300-67,640</td>
<td>876338</td>
<td>47,500-53,200</td>
</tr>
<tr>
<td>107,400-111,800</td>
<td>624446</td>
<td>67,640-69,980</td>
<td>795392</td>
<td>53,200-58,900</td>
</tr>
<tr>
<td>111,800-116,200</td>
<td>954348</td>
<td>69,980-72,320</td>
<td>553440</td>
<td>58,900-64,600</td>
</tr>
<tr>
<td>116,200-120,600</td>
<td>519440</td>
<td>72,320-74,660</td>
<td>188096</td>
<td>64,600-70,300</td>
</tr>
<tr>
<td>120,600-125,000</td>
<td>74784</td>
<td>74,660-77,000</td>
<td>23664</td>
<td>70,300-76,000</td>
</tr>
</tbody>
</table>

**Mean** = 109.428  
**Variance** = 45.499  
**Std.Dev.** = 6.745

<table>
<thead>
<tr>
<th>Intervals</th>
<th>Numbers</th>
<th>Mean WIP Inventory</th>
<th>Resource Utilization</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,900-8,190</td>
<td>1667</td>
<td>4,313-4,548</td>
<td>6444</td>
<td>0.700-0.716</td>
</tr>
<tr>
<td>8,190-10,480</td>
<td>29563</td>
<td>4,548-4,784</td>
<td>119712</td>
<td>0.716-0.733</td>
</tr>
<tr>
<td>10,480-12,770</td>
<td>151526</td>
<td>4,784-5,019</td>
<td>373712</td>
<td>0.733-0.749</td>
</tr>
<tr>
<td>12,770-15,080</td>
<td>356828</td>
<td>5,019-5,255</td>
<td>581472</td>
<td>0.749-0.768</td>
</tr>
<tr>
<td>15,080-17,350</td>
<td>600876</td>
<td>5,255-5,490</td>
<td>757058</td>
<td>0.768-0.782</td>
</tr>
<tr>
<td>17,350-19,640</td>
<td>794622</td>
<td>5,490-5,725</td>
<td>746592</td>
<td>0.782-0.800</td>
</tr>
<tr>
<td>19,640-21,930</td>
<td>783556</td>
<td>5,725-5,961</td>
<td>551024</td>
<td>0.798-0.814</td>
</tr>
<tr>
<td>21,930-24,220</td>
<td>619762</td>
<td>5,961-6,196</td>
<td>363458</td>
<td>0.814-0.830</td>
</tr>
<tr>
<td>24,220-26,510</td>
<td>285791</td>
<td>6,196-6,432</td>
<td>121936</td>
<td>0.830-0.847</td>
</tr>
<tr>
<td>26,510-28,800</td>
<td>25610</td>
<td>6,432-6,687</td>
<td>7378</td>
<td>0.847-0.863</td>
</tr>
</tbody>
</table>

**Mean** = 19.153  
**Variance** = 13.678  
**Std.Dev.** = 3.898

<table>
<thead>
<tr>
<th>Intervals</th>
<th>Numbers</th>
<th>Throughput</th>
<th>No. of Tardy jobs</th>
<th>Summation of *</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.111-0.114</td>
<td>10912</td>
<td>2.000-2.700</td>
<td>12</td>
<td>75.600-83.180</td>
</tr>
<tr>
<td>0.114-0.116</td>
<td>298796</td>
<td>2.700-3.400</td>
<td>2564</td>
<td>83.180-90.720</td>
</tr>
<tr>
<td>0.116-0.119</td>
<td>1008416</td>
<td>3.400-4.100</td>
<td>69130</td>
<td>90.720-96.280</td>
</tr>
<tr>
<td>0.119-0.121</td>
<td>110016</td>
<td>4.100-4.800</td>
<td>0</td>
<td>96.280-105.840</td>
</tr>
<tr>
<td>0.121-0.124</td>
<td>738040</td>
<td>4.800-5.500</td>
<td>507550</td>
<td>105.840-113.400</td>
</tr>
<tr>
<td>0.124-0.127</td>
<td>333886</td>
<td>5.500-6.200</td>
<td>1277360</td>
<td>113.400-120.800</td>
</tr>
<tr>
<td>0.127-0.129</td>
<td>69904</td>
<td>6.200-6.900</td>
<td>0</td>
<td>120.800-128.520</td>
</tr>
<tr>
<td>0.129-0.132</td>
<td>50784</td>
<td>6.900-7.600</td>
<td>1288698</td>
<td>128.520-136.080</td>
</tr>
<tr>
<td>0.132-0.134</td>
<td>5936</td>
<td>7.600-8.300</td>
<td>443714</td>
<td>136.080-143.840</td>
</tr>
<tr>
<td>0.134-0.137</td>
<td>1920</td>
<td>8.300-9.000</td>
<td>39774</td>
<td>143.840-151.200</td>
</tr>
</tbody>
</table>

**Mean** = 0.120  
**Variance** = 0.000  
**Std.Dev.** = 0.004
<table>
<thead>
<tr>
<th>Data Group #8</th>
<th>* Mean Flow Time</th>
<th>* Maximum Tardiness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>81.000- 86.300</td>
<td>360 58.700- 82.010</td>
<td>4088 29.000- 34.500</td>
</tr>
<tr>
<td>86.300- 91.800</td>
<td>4132 62.010- 64.320</td>
<td>85280 34.500- 40.000</td>
</tr>
<tr>
<td>91.800- 96.900</td>
<td>25058 64.320- 66.630</td>
<td>251848 40.000- 45.500</td>
</tr>
<tr>
<td>96.900- 102.200</td>
<td>113038 66.630- 68.940</td>
<td>494400 45.500- 51.000</td>
</tr>
<tr>
<td>102.200- 107.500</td>
<td>243184 68.940- 71.250</td>
<td>731588 51.000- 56.500</td>
</tr>
<tr>
<td>107.500- 112.800</td>
<td>441388 71.250- 73.560</td>
<td>837088 56.500- 62.000</td>
</tr>
<tr>
<td>112.800- 118.100</td>
<td>831944 73.560- 75.870</td>
<td>689152 62.000- 67.500</td>
</tr>
<tr>
<td>118.100- 123.400</td>
<td>843476 75.870- 78.180</td>
<td>406976 67.500- 73.000</td>
</tr>
<tr>
<td>123.400- 128.700</td>
<td>949952 78.180- 80.490</td>
<td>133896 73.000- 78.500</td>
</tr>
<tr>
<td>128.700- 134.000</td>
<td>176288 80.490- 82.800</td>
<td>14898 76.500- 84.000</td>
</tr>
<tr>
<td>Mean = 118.182</td>
<td>Mean = 71.900</td>
<td>Mean = 63.126</td>
</tr>
<tr>
<td>Variance = 61.011</td>
<td>Variance = 13.707</td>
<td>Variance = 90.408</td>
</tr>
<tr>
<td>Std.Dev. = 7.811</td>
<td>Std.Dev. = 3.702</td>
<td>Std.Dev. = 9.508</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mean Tardiness</th>
<th>WIP Inventory</th>
<th>Resource Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>12.200- 14.570</td>
<td>4906 4.418- 4.649</td>
<td>22752 0.685- 0.700</td>
</tr>
<tr>
<td>14.570- 16.940</td>
<td>69410 4.649- 4.880</td>
<td>201712 0.700- 0.715</td>
</tr>
<tr>
<td>16.940- 19.310</td>
<td>262363 4.880- 5.111</td>
<td>412832 0.715- 0.730</td>
</tr>
<tr>
<td>19.310- 21.680</td>
<td>484818 5.111- 5.342</td>
<td>567888 0.730- 0.745</td>
</tr>
<tr>
<td>21.680- 24.050</td>
<td>736752 5.342- 5.573</td>
<td>713952 0.745- 0.761</td>
</tr>
<tr>
<td>24.050- 26.420</td>
<td>825199 5.573- 5.804</td>
<td>680832 0.761- 0.778</td>
</tr>
<tr>
<td>26.420- 28.790</td>
<td>661683 5.804- 6.035</td>
<td>507584 0.776- 0.791</td>
</tr>
<tr>
<td>28.790- 31.160</td>
<td>431659 6.035- 6.286</td>
<td>343552 0.791- 0.806</td>
</tr>
<tr>
<td>31.160- 33.530</td>
<td>141214 6.286- 6.497</td>
<td>141984 0.806- 0.821</td>
</tr>
<tr>
<td>33.530- 35.900</td>
<td>10796 6.497- 6.728</td>
<td>15712 0.821- 0.836</td>
</tr>
<tr>
<td>Mean = 24.709</td>
<td>Mean = 5.538</td>
<td>Mean = 0.746</td>
</tr>
<tr>
<td>Variance = 14.844</td>
<td>Variance = 0.178</td>
<td>Variance = 0.000</td>
</tr>
<tr>
<td>Std.Dev. = 3.853</td>
<td>Std.Dev. = 0.422</td>
<td>Std.Dev. = 0.022</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Throughput</th>
<th>No. of Tardy jobs</th>
<th>Summation of *</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>0.112- 0.115</td>
<td>1904 4.000- 4.600</td>
<td>1158 82.400- 99.630</td>
</tr>
<tr>
<td>0.115- 0.117</td>
<td>11760 4.600- 5.200</td>
<td>49398 99.630- 106.860</td>
</tr>
<tr>
<td>0.117- 0.119</td>
<td>776592 5.200- 5.800</td>
<td>0 106.860- 114.090</td>
</tr>
<tr>
<td>0.119- 0.122</td>
<td>1261424 5.800- 6.400</td>
<td>436602 114.090- 121.320</td>
</tr>
<tr>
<td>0.122- 0.125</td>
<td>494224 6.400- 7.000</td>
<td>1326378 121.320- 128.550</td>
</tr>
<tr>
<td>0.125- 0.127</td>
<td>652416 7.000- 7.600</td>
<td>0 128.550- 135.780</td>
</tr>
<tr>
<td>0.127- 0.130</td>
<td>167008 7.600- 8.200</td>
<td>1357980 135.780- 143.010</td>
</tr>
<tr>
<td>0.130- 0.132</td>
<td>132144 8.200- 8.800</td>
<td>0 143.010- 150.240</td>
</tr>
<tr>
<td>0.132- 0.134</td>
<td>22498 8.800- 9.400</td>
<td>454464 150.240- 157.470</td>
</tr>
<tr>
<td>0.134- 0.137</td>
<td>8832 9.400- 10.000</td>
<td>2820 157.470- 164.700</td>
</tr>
<tr>
<td>Mean = 0.122</td>
<td>Mean = 7.479</td>
<td>Mean = 136.026</td>
</tr>
<tr>
<td>Variance = 0.000</td>
<td>Variance = 0.831</td>
<td>Variance = 131.950</td>
</tr>
<tr>
<td>Std.Dev. = 0.004</td>
<td>Std.Dev. = 0.911</td>
<td>Std.Dev. = 11.487</td>
</tr>
<tr>
<td>Data Group #9</td>
<td>* Mean Flow Time</td>
<td>* Maximum Tardiness</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>86.400- 91.400</td>
<td>184</td>
<td>61.900- 64.510</td>
</tr>
<tr>
<td>91.400- 96.800</td>
<td>2800</td>
<td>64.510- 67.120</td>
</tr>
<tr>
<td>96.800- 102.200</td>
<td>28840</td>
<td>67.120- 69.730</td>
</tr>
<tr>
<td>107.800- 113.000</td>
<td>222632</td>
<td>72.340- 74.850</td>
</tr>
<tr>
<td>113.000- 118.400</td>
<td>529828</td>
<td>74.850- 77.560</td>
</tr>
<tr>
<td>118.400- 123.800</td>
<td>709224</td>
<td>77.560- 80.170</td>
</tr>
<tr>
<td>123.800- 129.200</td>
<td>1011856</td>
<td>80.170- 82.780</td>
</tr>
<tr>
<td>129.200- 134.600</td>
<td>891348</td>
<td>82.780- 85.390</td>
</tr>
<tr>
<td>134.600- 140.000</td>
<td>165816</td>
<td>85.390- 88.000</td>
</tr>
<tr>
<td><strong>Mean</strong> = 123.862</td>
<td>Mean = 76.860</td>
<td><strong>Mean</strong> = 84.985</td>
</tr>
<tr>
<td><strong>Variance</strong> = 80.718</td>
<td><strong>Variance</strong> = 16.217</td>
<td><strong>Variance</strong> = 71.975</td>
</tr>
<tr>
<td><strong>Std.Dev.</strong> = 7.792</td>
<td><strong>Std.Dev.</strong> = 4.027</td>
<td><strong>Std.Dev.</strong> = 8.484</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mean Tardiness</th>
<th>WIP Inventory</th>
<th>Resource Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>13.800- 16.310</td>
<td>2408</td>
<td>4.197- 4.450</td>
</tr>
<tr>
<td>16.310- 18.820</td>
<td>50086</td>
<td>4.450- 4.703</td>
</tr>
<tr>
<td>18.820- 21.330</td>
<td>2400070</td>
<td>4.703- 4.956</td>
</tr>
<tr>
<td>21.330- 23.840</td>
<td>508750</td>
<td>4.956- 5.209</td>
</tr>
<tr>
<td>23.840- 26.350</td>
<td>752251</td>
<td>5.209- 5.463</td>
</tr>
<tr>
<td>26.350- 28.860</td>
<td>847151</td>
<td>5.463- 5.716</td>
</tr>
<tr>
<td>28.860- 31.370</td>
<td>889645</td>
<td>5.716- 5.989</td>
</tr>
<tr>
<td>31.370- 33.880</td>
<td>404995</td>
<td>5.989- 6.222</td>
</tr>
<tr>
<td>33.880- 36.390</td>
<td>122654</td>
<td>6.222- 6.475</td>
</tr>
<tr>
<td>36.390- 38.900</td>
<td>10780</td>
<td>6.475- 6.728</td>
</tr>
<tr>
<td><strong>Mean</strong> = 27.058</td>
<td><strong>Mean</strong> = 5.519</td>
<td><strong>Mean</strong> = 0.749</td>
</tr>
<tr>
<td><strong>Variance</strong> = 15.362</td>
<td><strong>Variance</strong> = 0.192</td>
<td><strong>Variance</strong> = 0.001</td>
</tr>
<tr>
<td><strong>Std.Dev.</strong> = 3.919</td>
<td><strong>Std.Dev.</strong> = 0.439</td>
<td><strong>Std.Dev.</strong> = 0.024</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Throughput</th>
<th>No. of Tardy jobs</th>
<th>Summation of *</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>0.112- 0.115</td>
<td>24352</td>
<td>4.000- 4.600</td>
</tr>
<tr>
<td>0.115- 0.118</td>
<td>350816</td>
<td>4.600- 5.200</td>
</tr>
<tr>
<td>0.118- 0.121</td>
<td>1038544</td>
<td>5.200- 5.800</td>
</tr>
<tr>
<td>0.121- 0.124</td>
<td>1047328</td>
<td>5.800- 6.400</td>
</tr>
<tr>
<td>0.124- 0.128</td>
<td>689512</td>
<td>6.400- 7.000</td>
</tr>
<tr>
<td>0.128- 0.131</td>
<td>328352</td>
<td>7.000- 7.600</td>
</tr>
<tr>
<td>0.131- 0.134</td>
<td>112498</td>
<td>7.600- 8.200</td>
</tr>
<tr>
<td>0.134- 0.137</td>
<td>18844</td>
<td>8.200- 8.800</td>
</tr>
<tr>
<td>0.137- 0.140</td>
<td>10192</td>
<td>8.800- 9.400</td>
</tr>
<tr>
<td>0.140- 0.143</td>
<td>1360</td>
<td>9.400- 10.000</td>
</tr>
<tr>
<td><strong>Mean</strong> = 0.123</td>
<td><strong>Mean</strong> = 7.886</td>
<td><strong>Mean</strong> = 141.785</td>
</tr>
<tr>
<td><strong>Variance</strong> = 0.000</td>
<td><strong>Variance</strong> = 0.753</td>
<td><strong>Variance</strong> = 113.035</td>
</tr>
<tr>
<td><strong>Std.Dev.</strong> = 0.204</td>
<td><strong>Std.Dev.</strong> = 0.888</td>
<td><strong>Std.Dev.</strong> = 10.832</td>
</tr>
</tbody>
</table>
# Data Group#10

<table>
<thead>
<tr>
<th>Intervals</th>
<th>Numbers</th>
<th>Intervals</th>
<th>Numbers</th>
<th>Intervals</th>
<th>Numbers</th>
<th>Mean Flow Time</th>
<th>*</th>
<th>Mean Flow Time</th>
<th>*</th>
<th>Maximum Tardiness</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>86.000- 91.600</td>
<td>254</td>
<td>64.400- 68.800</td>
<td>4488</td>
<td>38.000- 40.300</td>
<td>2550</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>91.600- 97.200</td>
<td>4080</td>
<td>66.660- 72.300</td>
<td>62712</td>
<td>40.300- 44.600</td>
<td>28066</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>97.200- 102.800</td>
<td>24352</td>
<td>68.920- 71.180</td>
<td>218856</td>
<td>44.800- 48.900</td>
<td>82554</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>108.400-114.000</td>
<td>214856</td>
<td>73.440- 75.700</td>
<td>692928</td>
<td>53.200- 57.500</td>
<td>190738</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>114.000-119.600</td>
<td>480708</td>
<td>75.700- 77.980</td>
<td>817848</td>
<td>57.500- 61.800</td>
<td>473748</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>119.600-125.200</td>
<td>768920</td>
<td>77.980- 80.220</td>
<td>751176</td>
<td>61.800- 66.100</td>
<td>1135188</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>125.200-130.800</td>
<td>629684</td>
<td>80.220- 82.480</td>
<td>485288</td>
<td>66.100- 70.400</td>
<td>610020</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>130.800-136.400</td>
<td>971664</td>
<td>82.480- 84.740</td>
<td>158816</td>
<td>70.400- 74.700</td>
<td>642492</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>136.400-142.000</td>
<td>422448</td>
<td>84.740- 87.000</td>
<td>13224</td>
<td>74.700- 79.000</td>
<td>314868</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mean = 126.203  Mean = 76.650  Mean = 86.229
Variance = 80.585  Variance = 13.312  Variance = 51.348
Std.Dev. = 8.977  Std.Dev. = 3.849  Std.Dev. = 7.168

<table>
<thead>
<tr>
<th>Intervals</th>
<th>Numbers</th>
<th>Intervals</th>
<th>Numbers</th>
<th>Intervals</th>
<th>Numbers</th>
<th>WIP Inventory</th>
<th>*</th>
<th>Resource Utilization</th>
<th>*</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.200- 18.460</td>
<td>6020</td>
<td>4.289- 4.511</td>
<td>24744</td>
<td>0.714- 0.728</td>
<td>48672</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18.460- 20.720</td>
<td>85938</td>
<td>4.511- 4.754</td>
<td>180432</td>
<td>0.728- 0.743</td>
<td>882632</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20.720-22.980</td>
<td>273688</td>
<td>4.754- 4.998</td>
<td>373488</td>
<td>0.743- 0.757</td>
<td>688808</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22.980- 25.240</td>
<td>549988</td>
<td>4.998- 5.238</td>
<td>548112</td>
<td>0.757- 0.771</td>
<td>1156224</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25.240- 27.500</td>
<td>770484</td>
<td>5.238- 5.481</td>
<td>696120</td>
<td>0.771- 0.786</td>
<td>409680</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27.500- 29.760</td>
<td>853880</td>
<td>5.481- 5.723</td>
<td>895544</td>
<td>0.786- 0.800</td>
<td>295608</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29.760- 32.020</td>
<td>675518</td>
<td>5.723- 5.965</td>
<td>538296</td>
<td>0.800- 0.814</td>
<td>269952</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32.020- 34.280</td>
<td>325318</td>
<td>5.965- 6.207</td>
<td>371160</td>
<td>0.814- 0.828</td>
<td>47856</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34.280- 36.540</td>
<td>825850</td>
<td>6.207- 6.450</td>
<td>174000</td>
<td>0.828- 0.843</td>
<td>22920</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36.540- 38.800</td>
<td>5668</td>
<td>6.450- 6.892</td>
<td>26904</td>
<td>0.843- 0.857</td>
<td>8448</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mean = 27.704  Mean = 5.478  Mean = 0.764
Variance = 12.500  Variance = 0.203  Variance = 0.000
Std.Dev. = 3.538  Std.Dev. = 0.451  Std.Dev. = 0.022

<table>
<thead>
<tr>
<th>Intervals</th>
<th>Numbers</th>
<th>Intervals</th>
<th>Numbers</th>
<th>Intervals</th>
<th>Numbers</th>
<th>Throughput</th>
<th>No. of Tardy jobs</th>
<th>Summation of *</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.119- 0.121</td>
<td>48672</td>
<td>4.000- 4.600</td>
<td>24</td>
<td>104.800-110.560</td>
<td>1294</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.121- 0.124</td>
<td>882832</td>
<td>4.600- 5.200</td>
<td>5784</td>
<td>110.560-116.320</td>
<td>26297</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.124- 0.126</td>
<td>688808</td>
<td>5.200- 5.800</td>
<td>0</td>
<td>116.320-122.080</td>
<td>97983</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.126- 0.129</td>
<td>1156224</td>
<td>5.800- 6.400</td>
<td>125328</td>
<td>122.080-127.840</td>
<td>157745</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.129- 0.131</td>
<td>408680</td>
<td>6.400- 7.000</td>
<td>698112</td>
<td>127.840-133.600</td>
<td>284171</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.131- 0.133</td>
<td>471312</td>
<td>7.000- 7.600</td>
<td>0</td>
<td>133.600-139.360</td>
<td>648848</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.133- 0.136</td>
<td>94248</td>
<td>7.600- 8.200</td>
<td>1533000</td>
<td>139.360-145.120</td>
<td>996430</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.136- 0.138</td>
<td>47568</td>
<td>8.200- 8.800</td>
<td>0</td>
<td>145.120-150.880</td>
<td>880276</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.138- 0.141</td>
<td>22920</td>
<td>8.800- 9.400</td>
<td>1151328</td>
<td>150.880-156.640</td>
<td>475840</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.141- 0.143</td>
<td>8448</td>
<td>9.400- 10.000</td>
<td>115224</td>
<td>156.640-162.400</td>
<td>53918</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mean = 0.127  Mean = 8.115  Mean = 14.179
Variance = 0.000  Variance = 0.778  Variance = 7.7175
Std.Dev. = 0.004  Std.Dev. = 0.881  Std.Dev. = 8.785
<table>
<thead>
<tr>
<th>Data Group#11</th>
<th>Maximum Flow Time</th>
<th>* Mean Flow Time</th>
<th>* Maximum Tardiness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
<td>Numbers</td>
</tr>
<tr>
<td>66.000-70.800</td>
<td>174</td>
<td>57.300-58.170</td>
<td>1920</td>
</tr>
<tr>
<td>70.800-75.600</td>
<td>5468</td>
<td>58.170-59.040</td>
<td>1448</td>
</tr>
<tr>
<td>75.600-80.400</td>
<td>3448</td>
<td>59.040-59.910</td>
<td>61414</td>
</tr>
<tr>
<td>80.400-85.200</td>
<td>125720</td>
<td>59.910-60.780</td>
<td>195854</td>
</tr>
<tr>
<td>85.200-90.000</td>
<td>249302</td>
<td>60.780-61.650</td>
<td>505544</td>
</tr>
<tr>
<td>90.000-94.800</td>
<td>634232</td>
<td>61.650-62.520</td>
<td>786896</td>
</tr>
<tr>
<td>94.800-99.600</td>
<td>919484</td>
<td>62.520-63.390</td>
<td>847564</td>
</tr>
<tr>
<td>99.600-104.400</td>
<td>885356</td>
<td>63.390-64.260</td>
<td>771186</td>
</tr>
<tr>
<td>104.400-109.200</td>
<td>563176</td>
<td>64.260-65.130</td>
<td>370176</td>
</tr>
<tr>
<td>109.200-114.000</td>
<td>211392</td>
<td>65.130-66.000</td>
<td>60288</td>
</tr>
</tbody>
</table>

Mean = 96.411  Mean = 62.700  Mean = 58.836
Variance = 51.785  Variance = 1.713  Variance = 43.997
Std.Dev. = 7.186  Std.Dev. = 1.309  Std.Dev. = 8.633

<table>
<thead>
<tr>
<th>Mean Tardiness</th>
<th>WIP Inventory</th>
<th>Resource Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>17.900-19.280</td>
<td>3598</td>
<td>5.100-5.187</td>
</tr>
<tr>
<td>19.280-20.660</td>
<td>53748</td>
<td>5.187-5.273</td>
</tr>
<tr>
<td>20.680-22.040</td>
<td>253232</td>
<td>5.273-5.360</td>
</tr>
<tr>
<td>22.040-23.420</td>
<td>804622</td>
<td>5.380-5.447</td>
</tr>
<tr>
<td>23.420-24.800</td>
<td>859787</td>
<td>5.447-5.533</td>
</tr>
<tr>
<td>24.800-26.180</td>
<td>782028</td>
<td>5.533-5.620</td>
</tr>
<tr>
<td>26.180-27.560</td>
<td>624172</td>
<td>5.620-5.707</td>
</tr>
<tr>
<td>27.560-28.840</td>
<td>324830</td>
<td>5.707-5.794</td>
</tr>
<tr>
<td>28.840-30.320</td>
<td>105198</td>
<td>5.794-5.880</td>
</tr>
<tr>
<td>30.320-31.700</td>
<td>20308</td>
<td>5.880-5.987</td>
</tr>
</tbody>
</table>

Mean = 24.972  Mean = 5.527  Mean = 0.730
Variance = 4.633  Variance = 0.012  Variance = 0.001
Std.Dev. = 2.152  Std.Dev. = 0.109  Std.Dev. = 0.027

<table>
<thead>
<tr>
<th>Throughput</th>
<th>No. of Tardy jobs</th>
<th>Summatation of</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>0.152-0.158</td>
<td>374976</td>
<td>5.000-5.500</td>
</tr>
<tr>
<td>0.156-0.159</td>
<td>1081678</td>
<td>5.500-6.000</td>
</tr>
<tr>
<td>0.159-0.163</td>
<td>528182</td>
<td>6.000-6.500</td>
</tr>
<tr>
<td>0.163-0.167</td>
<td>1008033</td>
<td>6.500-7.000</td>
</tr>
<tr>
<td>0.167-0.171</td>
<td>297792</td>
<td>7.000-7.500</td>
</tr>
<tr>
<td>0.171-0.174</td>
<td>174720</td>
<td>7.500-8.000</td>
</tr>
<tr>
<td>0.174-0.178</td>
<td>74888</td>
<td>8.000-8.500</td>
</tr>
<tr>
<td>0.178-0.182</td>
<td>38028</td>
<td>8.500-9.000</td>
</tr>
<tr>
<td>0.182-0.185</td>
<td>13632</td>
<td>9.000-9.500</td>
</tr>
<tr>
<td>0.185-0.189</td>
<td>384</td>
<td>9.500-10.000</td>
</tr>
</tbody>
</table>

Mean = 0.182  Mean = 8.054  Mean = 12.538
Variance = 0.000  Variance = 0.748  Variance = 51.042
Std.Dev. = 0.006  Std.Dev. = 0.884  Std.Dev. = 7.144
<table>
<thead>
<tr>
<th>Intervals</th>
<th>Numbers</th>
<th>Intervals</th>
<th>Numbers</th>
<th>Intervals</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.600-7.000</td>
<td>74</td>
<td>58.800-59.520</td>
<td>3384</td>
<td>30.000-34.600</td>
<td>422</td>
</tr>
<tr>
<td>7.000-7.500</td>
<td>2822</td>
<td>59.520-60.440</td>
<td>29808</td>
<td>34.600-39.200</td>
<td>7652</td>
</tr>
<tr>
<td>7.500-8.000</td>
<td>24358</td>
<td>60.440-61.360</td>
<td>101864</td>
<td>39.200-43.800</td>
<td>26156</td>
</tr>
<tr>
<td>8.000-8.500</td>
<td>75806</td>
<td>61.360-62.280</td>
<td>282096</td>
<td>43.800-48.400</td>
<td>105974</td>
</tr>
<tr>
<td>8.500-9.000</td>
<td>201576</td>
<td>62.280-63.200</td>
<td>514296</td>
<td>48.400-53.000</td>
<td>203890</td>
</tr>
<tr>
<td>9.000-9.500</td>
<td>470902</td>
<td>63.200-64.120</td>
<td>859484</td>
<td>53.000-57.600</td>
<td>500526</td>
</tr>
<tr>
<td>9.500-10.000</td>
<td>787092</td>
<td>64.120-65.040</td>
<td>832104</td>
<td>57.600-62.200</td>
<td>822032</td>
</tr>
<tr>
<td>10.000-10.500</td>
<td>897336</td>
<td>65.040-65.960</td>
<td>628888</td>
<td>62.200-68.800</td>
<td>765336</td>
</tr>
<tr>
<td>105.200-110.100</td>
<td>789232</td>
<td>65.880-68.880</td>
<td>320040</td>
<td>66.800-71.400</td>
<td>782772</td>
</tr>
<tr>
<td>110.100-115.000</td>
<td>379800</td>
<td>66.880-67.800</td>
<td>59258</td>
<td>71.400-76.000</td>
<td>413640</td>
</tr>
</tbody>
</table>

**Mean = 101.332**  
**Variance = 55.245**  
**Std.Dev. = 7.433**

<table>
<thead>
<tr>
<th>Intervals</th>
<th>Numbers</th>
<th>Intervals</th>
<th>Numbers</th>
<th>Intervals</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>17.700-19.310</td>
<td>1942</td>
<td>5.100-5.187</td>
<td>5780</td>
<td>0.682-0.702</td>
<td>352224</td>
</tr>
<tr>
<td>19.310-20.820</td>
<td>34126</td>
<td>5.187-5.273</td>
<td>81648</td>
<td>0.702-0.722</td>
<td>920736</td>
</tr>
<tr>
<td>20.820-22.530</td>
<td>185318</td>
<td>5.273-5.360</td>
<td>375120</td>
<td>0.722-0.742</td>
<td>1142784</td>
</tr>
<tr>
<td>22.530-24.140</td>
<td>512192</td>
<td>5.360-5.446</td>
<td>851832</td>
<td>0.742-0.782</td>
<td>449208</td>
</tr>
<tr>
<td>24.140-25.750</td>
<td>841118</td>
<td>5.446-5.533</td>
<td>1065024</td>
<td>0.782-0.822</td>
<td>558992</td>
</tr>
<tr>
<td>25.750-27.360</td>
<td>924780</td>
<td>5.533-5.620</td>
<td>808344</td>
<td>0.822-0.862</td>
<td>110376</td>
</tr>
<tr>
<td>27.360-29.870</td>
<td>706888</td>
<td>5.620-5.706</td>
<td>345834</td>
<td>0.862-0.882</td>
<td>84384</td>
</tr>
<tr>
<td>28.870-30.580</td>
<td>332576</td>
<td>5.706-5.793</td>
<td>81286</td>
<td>0.882-0.942</td>
<td>9576</td>
</tr>
<tr>
<td>30.580-32.190</td>
<td>82248</td>
<td>5.793-5.879</td>
<td>13032</td>
<td>0.942-0.982</td>
<td>1666</td>
</tr>
<tr>
<td>32.190-33.800</td>
<td>7632</td>
<td>5.879-5.966</td>
<td>1368</td>
<td>0.882-0.882</td>
<td>884</td>
</tr>
</tbody>
</table>

**Mean = 28.145**  
**Variance = 5.310**  
**Std.Dev. = 2.304**

<table>
<thead>
<tr>
<th>Intervals</th>
<th>Numbers</th>
<th>Intervals</th>
<th>Numbers</th>
<th>Intervals</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.152-0.158</td>
<td>683208</td>
<td>5.000-5.500</td>
<td>5423</td>
<td>91.700-96.870</td>
<td>470</td>
</tr>
<tr>
<td>0.158-0.161</td>
<td>589752</td>
<td>5.500-6.000</td>
<td>0</td>
<td>96.870-102.040</td>
<td>6248</td>
</tr>
<tr>
<td>0.161-0.165</td>
<td>1142784</td>
<td>6.000-6.500</td>
<td>181885</td>
<td>102.040-107.210</td>
<td>36638</td>
</tr>
<tr>
<td>0.165-0.170</td>
<td>789408</td>
<td>6.500-7.000</td>
<td>0</td>
<td>107.210-112.380</td>
<td>119864</td>
</tr>
<tr>
<td>0.170-0.174</td>
<td>216792</td>
<td>7.000-7.500</td>
<td>989866</td>
<td>112.380-117.550</td>
<td>301448</td>
</tr>
<tr>
<td>0.174-0.178</td>
<td>110376</td>
<td>7.500-8.000</td>
<td>0</td>
<td>117.550-122.720</td>
<td>610284</td>
</tr>
<tr>
<td>0.178-0.183</td>
<td>84384</td>
<td>8.000-8.500</td>
<td>1613876</td>
<td>122.720-127.890</td>
<td>835236</td>
</tr>
<tr>
<td>0.183-0.187</td>
<td>9576</td>
<td>8.500-9.000</td>
<td>0</td>
<td>127.890-133.060</td>
<td>864988</td>
</tr>
<tr>
<td>0.187-0.192</td>
<td>1656</td>
<td>9.000-9.500</td>
<td>791051</td>
<td>133.080-138.230</td>
<td>660904</td>
</tr>
<tr>
<td>0.192-0.196</td>
<td>864</td>
<td>9.500-10.000</td>
<td>87797</td>
<td>138.230-143.400</td>
<td>202620</td>
</tr>
</tbody>
</table>

**Mean = 0.183**  
**Variance = 0.000**  
**Std.Dev. = 0.006**
<table>
<thead>
<tr>
<th>Data Group #13</th>
<th>Maximum Flow Time</th>
<th>Mean Flow Time</th>
<th>Maximum Tardiness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
<td>Numbers</td>
</tr>
<tr>
<td>69.000-74.200</td>
<td>381</td>
<td>58.100-59.180</td>
<td>866</td>
</tr>
<tr>
<td>74.200-79.400</td>
<td>4913</td>
<td>58.180-60.260</td>
<td>8352</td>
</tr>
<tr>
<td>79.400-84.600</td>
<td>33078</td>
<td>60.260-61.340</td>
<td>47520</td>
</tr>
<tr>
<td>84.600-89.800</td>
<td>121016</td>
<td>61.340-62.420</td>
<td>123048</td>
</tr>
<tr>
<td>89.800-95.000</td>
<td>237684</td>
<td>62.420-63.500</td>
<td>252468</td>
</tr>
<tr>
<td>95.000-100.200</td>
<td>559360</td>
<td>63.500-64.580</td>
<td>546768</td>
</tr>
<tr>
<td>100.200-105.400</td>
<td>707080</td>
<td>64.580-65.660</td>
<td>891792</td>
</tr>
<tr>
<td>105.400-110.600</td>
<td>728080</td>
<td>65.660-67.740</td>
<td>1020024</td>
</tr>
<tr>
<td>110.800-115.800</td>
<td>727280</td>
<td>66.740-67.820</td>
<td>613872</td>
</tr>
<tr>
<td>115.800-121.000</td>
<td>509928</td>
<td>67.820-68.900</td>
<td>123912</td>
</tr>
</tbody>
</table>

**Mean = 105.855**

**Variance = 73.380**

**Std. Dev. = 8.568**

<table>
<thead>
<tr>
<th>Data Group #13</th>
<th>WIP Inventory</th>
<th>Resource Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>19.900-21.530</td>
<td>1156</td>
<td>4.915-5.020</td>
</tr>
<tr>
<td>21.530-23.160</td>
<td>12884</td>
<td>5.020-5.125</td>
</tr>
<tr>
<td>23.180-24.790</td>
<td>69540</td>
<td>5.125-5.231</td>
</tr>
<tr>
<td>24.790-26.420</td>
<td>258016</td>
<td>5.231-5.338</td>
</tr>
<tr>
<td>26.420-28.050</td>
<td>601494</td>
<td>5.336-5.441</td>
</tr>
<tr>
<td>28.050-29.680</td>
<td>946724</td>
<td>5.446-5.564</td>
</tr>
<tr>
<td>29.680-31.310</td>
<td>1001520</td>
<td>5.546-5.651</td>
</tr>
<tr>
<td>31.310-32.940</td>
<td>500888</td>
<td>5.651-5.757</td>
</tr>
<tr>
<td>32.940-34.570</td>
<td>163224</td>
<td>5.757-5.862</td>
</tr>
<tr>
<td>34.570-36.200</td>
<td>13156</td>
<td>5.862-5.987</td>
</tr>
</tbody>
</table>

**Mean = 29.444**

**Variance = 4.913**

**Std. Dev. = 2.217**

<table>
<thead>
<tr>
<th>Data Group #13</th>
<th>Throughput</th>
<th>No. of Tardy jobs</th>
<th>Summation of *</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
<td>Numbers</td>
</tr>
<tr>
<td>0.152-0.158</td>
<td>781488</td>
<td>5.000-5.500</td>
<td>1158</td>
</tr>
<tr>
<td>0.158-0.161</td>
<td>627120</td>
<td>5.500-6.000</td>
<td>0</td>
</tr>
<tr>
<td>0.161-0.165</td>
<td>1107648</td>
<td>6.000-6.500</td>
<td>50412</td>
</tr>
<tr>
<td>0.165-0.170</td>
<td>721728</td>
<td>6.500-7.000</td>
<td>0</td>
</tr>
<tr>
<td>0.170-0.174</td>
<td>216568</td>
<td>7.000-7.500</td>
<td>504822</td>
</tr>
<tr>
<td>0.174-0.178</td>
<td>79820</td>
<td>7.500-8.000</td>
<td>0</td>
</tr>
<tr>
<td>0.178-0.183</td>
<td>80064</td>
<td>8.000-8.500</td>
<td>1517179</td>
</tr>
<tr>
<td>0.183-0.187</td>
<td>11232</td>
<td>8.500-9.000</td>
<td>0</td>
</tr>
<tr>
<td>0.187-0.192</td>
<td>2592</td>
<td>9.000-9.500</td>
<td>1299876</td>
</tr>
<tr>
<td>0.192-0.198</td>
<td>1440</td>
<td>9.500-10.000</td>
<td>255354</td>
</tr>
</tbody>
</table>

**Mean = 0.183**

**Variance = 0.000**

**Std. Dev. = 0.006**
<table>
<thead>
<tr>
<th>Intervals</th>
<th>Numbers</th>
<th>Numbers</th>
<th>* Mean Flow Time</th>
<th>Numbers</th>
<th>Numbers</th>
<th>* Maximum Tardiness</th>
</tr>
</thead>
<tbody>
<tr>
<td>74.000- 79.200</td>
<td>3307</td>
<td>59.100- 60.100</td>
<td>2088</td>
<td>34.000- 39.300</td>
<td>1304</td>
<td></td>
</tr>
<tr>
<td>79.200- 84.400</td>
<td>2001</td>
<td>60.100- 61.100</td>
<td>1641</td>
<td>39.300- 44.600</td>
<td>13902</td>
<td></td>
</tr>
<tr>
<td>84.400- 89.600</td>
<td>7517</td>
<td>61.100- 62.100</td>
<td>65088</td>
<td>44.600- 49.900</td>
<td>61798</td>
<td></td>
</tr>
<tr>
<td>89.600- 94.800</td>
<td>190340</td>
<td>62.100- 63.100</td>
<td>172800</td>
<td>49.900- 55.200</td>
<td>225032</td>
<td></td>
</tr>
<tr>
<td>94.800- 100.000</td>
<td>288888</td>
<td>63.100- 64.100</td>
<td>401888</td>
<td>55.200- 60.500</td>
<td>397074</td>
<td></td>
</tr>
<tr>
<td>100.000- 105.200</td>
<td>558884</td>
<td>64.100- 65.100</td>
<td>666288</td>
<td>60.500- 65.800</td>
<td>657988</td>
<td></td>
</tr>
<tr>
<td>105.200- 110.400</td>
<td>808860</td>
<td>65.100- 66.100</td>
<td>871488</td>
<td>65.800- 71.100</td>
<td>994620</td>
<td></td>
</tr>
<tr>
<td>110.400- 115.600</td>
<td>689932</td>
<td>66.100- 67.100</td>
<td>818136</td>
<td>71.100- 76.400</td>
<td>695616</td>
<td></td>
</tr>
<tr>
<td>115.600- 120.800</td>
<td>715128</td>
<td>67.100- 68.100</td>
<td>494064</td>
<td>76.400- 81.700</td>
<td>337064</td>
<td></td>
</tr>
<tr>
<td>120.800- 126.000</td>
<td>500472</td>
<td>68.100- 69.100</td>
<td>120744</td>
<td>81.700- 87.000</td>
<td>244512</td>
<td></td>
</tr>
<tr>
<td>Mean = 109.885</td>
<td>Mean = 65.500</td>
<td>Mean = 67.990</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variance = 91.851</td>
<td>Variance = 2.467</td>
<td>Variance = 74.760</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Std.Dev = 9.573</td>
<td>Std.Dev = 1.571</td>
<td>Std.Dev = 8.848</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Intervals</th>
<th>Numbers</th>
<th>Numbers</th>
<th>Intervals</th>
<th>Numbers</th>
<th>Numbers</th>
<th>Intervals</th>
<th>Numbers</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>21.200- 22.850</td>
<td>2114</td>
<td>4.987- 5.067</td>
<td>3188</td>
<td>0.682- 0.700</td>
<td>370224</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22.650- 24.100</td>
<td>27874</td>
<td>5.067- 5.167</td>
<td>33552</td>
<td>0.700- 0.719</td>
<td>972864</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24.100- 25.550</td>
<td>178234</td>
<td>5.167- 5.267</td>
<td>150480</td>
<td>0.719- 0.737</td>
<td>541728</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25.550- 27.000</td>
<td>547116</td>
<td>5.267- 5.367</td>
<td>422208</td>
<td>0.737- 0.755</td>
<td>1028016</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27.000- 28.450</td>
<td>869810</td>
<td>5.367- 5.467</td>
<td>832536</td>
<td>0.755- 0.774</td>
<td>319998</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28.450- 29.900</td>
<td>993282</td>
<td>5.467- 5.567</td>
<td>1089936</td>
<td>0.774- 0.792</td>
<td>309744</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29.900- 31.350</td>
<td>648372</td>
<td>5.567- 5.667</td>
<td>788760</td>
<td>0.792- 0.810</td>
<td>57456</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31.350- 32.800</td>
<td>288208</td>
<td>5.667- 5.767</td>
<td>259344</td>
<td>0.810- 0.826</td>
<td>17424</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32.800- 34.250</td>
<td>68942</td>
<td>5.767- 5.867</td>
<td>44856</td>
<td>0.826- 0.847</td>
<td>9216</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34.250- 35.700</td>
<td>7050</td>
<td>5.867- 5.967</td>
<td>3960</td>
<td>0.847- 0.865</td>
<td>2180</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean = 28.734</td>
<td>Mean = 5.498</td>
<td>Mean = 0.733</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variance = 4.084</td>
<td>Variance = 0.017</td>
<td>Variance = 0.001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Std.Dev = 2.021</td>
<td>Std.Dev = 0.131</td>
<td>Std.Dev = 0.028</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Intervals</th>
<th>Numbers</th>
<th>Numbers</th>
<th>Intervals</th>
<th>Numbers</th>
<th>Numbers</th>
<th>Intervals</th>
<th>Numbers</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.152- 0.158</td>
<td>370224</td>
<td>5.000- 5.500</td>
<td>3488</td>
<td>98.500- 104.220</td>
<td>1292</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.158- 0.160</td>
<td>972864</td>
<td>5.500- 6.000</td>
<td>0</td>
<td>104.220- 109.940</td>
<td>18442</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.160- 0.164</td>
<td>1127088</td>
<td>6.000- 6.500</td>
<td>108738</td>
<td>109.940- 115.680</td>
<td>86244</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.164- 0.168</td>
<td>442665</td>
<td>6.500- 7.000</td>
<td>0</td>
<td>115.680- 121.380</td>
<td>247928</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.168- 0.172</td>
<td>319894</td>
<td>7.000- 7.500</td>
<td>782628</td>
<td>121.380- 127.100</td>
<td>503716</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.172- 0.176</td>
<td>309744</td>
<td>7.500- 8.000</td>
<td>0</td>
<td>127.100- 132.820</td>
<td>809138</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.176- 0.180</td>
<td>57456</td>
<td>8.000- 8.500</td>
<td>1580104</td>
<td>132.820- 138.540</td>
<td>880032</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.180- 0.184</td>
<td>17424</td>
<td>8.500- 9.000</td>
<td>0</td>
<td>138.540- 144.260</td>
<td>645520</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.184- 0.188</td>
<td>9216</td>
<td>9.000- 9.500</td>
<td>88310</td>
<td>144.260- 149.980</td>
<td>327240</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.188- 0.192</td>
<td>2180</td>
<td>9.500- 10.000</td>
<td>159534</td>
<td>149.980- 155.700</td>
<td>109248</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean = 0.183</td>
<td>Mean = 8.081</td>
<td>Mean = 133.490</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variance = 0.000</td>
<td>Variance = 0.785</td>
<td>Variance = 81.631</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Std.Dev = 0.006</td>
<td>Std.Dev = 0.886</td>
<td>Std.Dev = 9.035</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>*Mean Flow Time</td>
<td>Numbers</td>
<td>*Maximum Tardiness</td>
<td>Numbers</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>----------------</td>
<td>---------</td>
<td>-----------------</td>
<td>---------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>75.000- 80.700</td>
<td>4863</td>
<td>80.000- 80.910</td>
<td>2304</td>
<td>34.000- 39.000</td>
<td>1211</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80.700- 86.400</td>
<td>45244</td>
<td>80.810- 81.820</td>
<td>24000</td>
<td>39.000- 44.000</td>
<td>11035</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>86.400- 92.100</td>
<td>171545</td>
<td>61.820- 62.730</td>
<td>107712</td>
<td>44.000- 49.000</td>
<td>47040</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>92.100- 97.800</td>
<td>286018</td>
<td>62.730- 63.640</td>
<td>283152</td>
<td>49.000- 54.000</td>
<td>142540</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>97.800- 103.500</td>
<td>451018</td>
<td>63.640- 64.550</td>
<td>563808</td>
<td>54.000- 59.000</td>
<td>289940</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>103.500- 109.200</td>
<td>600272</td>
<td>64.550- 65.460</td>
<td>810000</td>
<td>59.000- 64.000</td>
<td>522274</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>109.200- 114.900</td>
<td>607392</td>
<td>65.460- 66.370</td>
<td>868944</td>
<td>64.000- 69.000</td>
<td>863592</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>114.900- 120.600</td>
<td>653608</td>
<td>66.370- 67.280</td>
<td>639360</td>
<td>69.000- 74.000</td>
<td>839368</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>120.600- 126.300</td>
<td>585496</td>
<td>67.280- 68.190</td>
<td>295488</td>
<td>74.000- 79.000</td>
<td>861312</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>126.300- 132.000</td>
<td>243344</td>
<td>68.190- 69.100</td>
<td>34032</td>
<td>79.000- 84.000</td>
<td>410488</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mean = 110.715
Variance = 119.965
Std. Dev. = 10.853

<table>
<thead>
<tr>
<th>Intervals</th>
<th>Numbers</th>
<th>*Mean Flow Time</th>
<th>Numbers</th>
<th>*Maximum Tardiness</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>21.700- 23.020</td>
<td>2982</td>
<td>5.068- 5.166</td>
<td>3648</td>
<td>0.677- 0.697</td>
<td>139728</td>
</tr>
<tr>
<td>23.020- 24.340</td>
<td>35364</td>
<td>5.168- 5.265</td>
<td>75312</td>
<td>0.697- 0.718</td>
<td>879800</td>
</tr>
<tr>
<td>24.340- 25.660</td>
<td>172769</td>
<td>5.265- 5.363</td>
<td>393744</td>
<td>0.718- 0.738</td>
<td>1277424</td>
</tr>
<tr>
<td>25.660- 26.980</td>
<td>493985</td>
<td>5.363- 5.462</td>
<td>883538</td>
<td>0.738- 0.758</td>
<td>499104</td>
</tr>
<tr>
<td>26.980- 28.300</td>
<td>928937</td>
<td>5.462- 5.560</td>
<td>1075728</td>
<td>0.758- 0.779</td>
<td>613536</td>
</tr>
<tr>
<td>28.300- 29.620</td>
<td>913220</td>
<td>5.560- 5.658</td>
<td>755952</td>
<td>0.779- 0.799</td>
<td>127392</td>
</tr>
<tr>
<td>29.620- 30.940</td>
<td>664242</td>
<td>5.658- 5.757</td>
<td>330912</td>
<td>0.799- 0.819</td>
<td>810720</td>
</tr>
<tr>
<td>30.940- 32.260</td>
<td>318316</td>
<td>5.757- 5.855</td>
<td>92736</td>
<td>0.819- 0.839</td>
<td>8208</td>
</tr>
<tr>
<td>32.260- 33.580</td>
<td>86546</td>
<td>5.855- 5.954</td>
<td>18176</td>
<td>0.839- 0.860</td>
<td>2016</td>
</tr>
<tr>
<td>33.580- 34.900</td>
<td>7028</td>
<td>5.954- 6.052</td>
<td>1058</td>
<td>0.880- 0.880</td>
<td>720</td>
</tr>
</tbody>
</table>

Mean = 28.606
Variance = 3.565
Std. Dev. = 1.888

<table>
<thead>
<tr>
<th>Intervals</th>
<th>Numbers</th>
<th>*Mean Flow Time</th>
<th>Numbers</th>
<th>*Maximum Tardiness</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.154- 0.159</td>
<td>139728</td>
<td>5.000- 5.500</td>
<td>3048</td>
<td>98.000- 103.460</td>
<td>1082</td>
</tr>
<tr>
<td>0.159- 0.163</td>
<td>879600</td>
<td>5.500- 6.000</td>
<td>0</td>
<td>103.460- 108.920</td>
<td>14405</td>
</tr>
<tr>
<td>0.163- 0.168</td>
<td>1277424</td>
<td>6.000- 6.500</td>
<td>109464</td>
<td>108.920- 114.380</td>
<td>65997</td>
</tr>
<tr>
<td>0.168- 0.172</td>
<td>499104</td>
<td>6.500- 7.000</td>
<td>0</td>
<td>114.380- 119.840</td>
<td>191938</td>
</tr>
<tr>
<td>0.172- 0.177</td>
<td>613536</td>
<td>7.000- 7.500</td>
<td>767772</td>
<td>119.840- 125.300</td>
<td>377234</td>
</tr>
<tr>
<td>0.177- 0.182</td>
<td>127392</td>
<td>7.500- 8.000</td>
<td>0</td>
<td>125.300- 130.760</td>
<td>617608</td>
</tr>
<tr>
<td>0.182- 0.186</td>
<td>81072</td>
<td>8.000- 8.500</td>
<td>1566720</td>
<td>130.760- 136.220</td>
<td>824532</td>
</tr>
<tr>
<td>0.186- 0.191</td>
<td>8208</td>
<td>8.500- 9.000</td>
<td>0</td>
<td>136.220- 141.680</td>
<td>823368</td>
</tr>
<tr>
<td>0.191- 0.195</td>
<td>2016</td>
<td>9.000- 9.500</td>
<td>1010316</td>
<td>141.680- 147.140</td>
<td>552860</td>
</tr>
<tr>
<td>0.195- 0.200</td>
<td>720</td>
<td>9.500- 10.000</td>
<td>171480</td>
<td>147.140- 152.600</td>
<td>159776</td>
</tr>
</tbody>
</table>

Mean = 0.187
Variance = 0.000
Std. Dev. = 0.006
<table>
<thead>
<tr>
<th>Data Group#16</th>
<th>Maximum Flow Time</th>
<th>* Mean Flow Time</th>
<th>* Maximum Tardiness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals Numbers</td>
<td>Intervals Numbers</td>
<td>Intervals Numbers</td>
<td>Intervals Numbers</td>
</tr>
<tr>
<td>59.000- 62.800</td>
<td>1844</td>
<td>40.400- 41.680</td>
<td>2736</td>
</tr>
<tr>
<td>62.600- 66.200</td>
<td>4260</td>
<td>41.680- 42.980</td>
<td>30312</td>
</tr>
<tr>
<td>66.200- 69.800</td>
<td>132746</td>
<td>42.980- 44.240</td>
<td>120312</td>
</tr>
<tr>
<td>69.800- 73.400</td>
<td>432718</td>
<td>44.240- 45.520</td>
<td>361872</td>
</tr>
<tr>
<td>73.400- 77.000</td>
<td>511632</td>
<td>45.520- 46.800</td>
<td>693000</td>
</tr>
<tr>
<td>77.000- 80.600</td>
<td>775128</td>
<td>46.800- 48.080</td>
<td>863208</td>
</tr>
<tr>
<td>80.600- 84.200</td>
<td>718820</td>
<td>48.080- 49.380</td>
<td>853920</td>
</tr>
<tr>
<td>84.200- 87.800</td>
<td>478320</td>
<td>49.380- 50.640</td>
<td>510984</td>
</tr>
<tr>
<td>87.800- 91.400</td>
<td>478030</td>
<td>50.640- 51.920</td>
<td>169272</td>
</tr>
<tr>
<td>91.400- 95.000</td>
<td>53323</td>
<td>51.920- 53.200</td>
<td>23184</td>
</tr>
<tr>
<td>Mean = 80.020</td>
<td>Mean = 47.650</td>
<td>Mean = 50.051</td>
<td></td>
</tr>
<tr>
<td>Variance = 40.040</td>
<td>Variance = 3.644</td>
<td>Variance = 237.523</td>
<td></td>
</tr>
<tr>
<td>Std.Dev. = 6.326</td>
<td>Std.Dev. = 1.909</td>
<td>Std.Dev. = 15.412</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mean Tardiness</th>
<th>WIP Inventory</th>
<th>Resource Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals Numbers</td>
<td>Intervals Numbers</td>
<td>Intervals Numbers</td>
</tr>
<tr>
<td>2.800- 4.480</td>
<td>1894</td>
<td>4.803- 4.929</td>
</tr>
<tr>
<td>4.480- 6.160</td>
<td>21354</td>
<td>4.929- 5.055</td>
</tr>
<tr>
<td>6.160- 7.840</td>
<td>103582</td>
<td>5.055- 5.182</td>
</tr>
<tr>
<td>7.840- 9.520</td>
<td>315108</td>
<td>5.182- 5.308</td>
</tr>
<tr>
<td>9.520- 11.200</td>
<td>650290</td>
<td>5.308- 5.434</td>
</tr>
<tr>
<td>11.200- 12.880</td>
<td>904110</td>
<td>5.434- 5.560</td>
</tr>
<tr>
<td>12.880- 14.560</td>
<td>918248</td>
<td>5.560- 5.686</td>
</tr>
<tr>
<td>14.560- 16.240</td>
<td>523472</td>
<td>5.686- 5.813</td>
</tr>
<tr>
<td>16.240- 17.920</td>
<td>161620</td>
<td>5.813- 5.939</td>
</tr>
<tr>
<td>17.920- 19.600</td>
<td>191222</td>
<td>5.939- 6.065</td>
</tr>
<tr>
<td>Mean = 12.431</td>
<td>Mean = 2.477</td>
<td>Mean = 0.728</td>
</tr>
<tr>
<td>Variance = 5.790</td>
<td>Variance = 0.037</td>
<td>Variance = 237.523</td>
</tr>
<tr>
<td>Std.Dev. = 2.406</td>
<td>Std.Dev. = 0.192</td>
<td>Std.Dev. = 0.030</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Throughput</th>
<th>No. of Tardy jobs</th>
<th>Summation of</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals Numbers</td>
<td>Intervals Numbers</td>
<td>Intervals Numbers</td>
</tr>
<tr>
<td>0.137- 0.141</td>
<td>18272</td>
<td>1.000- 1.700</td>
</tr>
<tr>
<td>0.141- 0.145</td>
<td>30180</td>
<td>1.700- 2.400</td>
</tr>
<tr>
<td>0.145- 0.148</td>
<td>786688</td>
<td>2.400- 3.100</td>
</tr>
<tr>
<td>0.148- 0.152</td>
<td>1069344</td>
<td>3.100- 3.800</td>
</tr>
<tr>
<td>0.152- 0.156</td>
<td>431712</td>
<td>3.800- 4.500</td>
</tr>
<tr>
<td>0.156- 0.160</td>
<td>700650</td>
<td>4.500- 5.200</td>
</tr>
<tr>
<td>0.160- 0.164</td>
<td>141984</td>
<td>5.200- 5.900</td>
</tr>
<tr>
<td>0.164- 0.167</td>
<td>187472</td>
<td>5.900- 6.600</td>
</tr>
<tr>
<td>0.167- 0.171</td>
<td>13536</td>
<td>6.600- 7.300</td>
</tr>
<tr>
<td>0.171- 0.175</td>
<td>15652</td>
<td>7.300- 8.000</td>
</tr>
<tr>
<td>Mean = 0.152</td>
<td>Mean = 5.264</td>
<td>Mean = 97.701</td>
</tr>
<tr>
<td>Variance = 0.000</td>
<td>Variance = 0.920</td>
<td>Variance = 252.782</td>
</tr>
<tr>
<td>Std.Dev. = 0.000</td>
<td>Std.Dev. = 0.959</td>
<td>Std.Dev. = 15.899</td>
</tr>
<tr>
<td>Data Group#17</td>
<td>* Mean Flow Time</td>
<td>* Maximum Flow Time</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>60.000 - 64.100</td>
<td>2728</td>
<td>43.300 - 44.700</td>
</tr>
<tr>
<td>64.100 - 68.200</td>
<td>2592</td>
<td>44.700 - 46.100</td>
</tr>
<tr>
<td>68.200 - 72.300</td>
<td>140062</td>
<td>46.100 - 47.500</td>
</tr>
<tr>
<td>72.300 - 76.400</td>
<td>380790</td>
<td>47.500 - 48.900</td>
</tr>
<tr>
<td>76.400 - 80.500</td>
<td>884084</td>
<td>48.900 - 50.300</td>
</tr>
<tr>
<td>80.500 - 84.600</td>
<td>761888</td>
<td>50.300 - 51.700</td>
</tr>
<tr>
<td>84.600 - 88.700</td>
<td>703520</td>
<td>51.700 - 53.100</td>
</tr>
<tr>
<td>88.700 - 92.800</td>
<td>488268</td>
<td>53.100 - 54.500</td>
</tr>
<tr>
<td>92.800 - 96.900</td>
<td>359018</td>
<td>54.500 - 55.900</td>
</tr>
<tr>
<td>96.900 - 101.000</td>
<td>82684</td>
<td>55.900 - 57.300</td>
</tr>
</tbody>
</table>

Mean = 83.722
Variance = 46.501
Std.Dev. = 6.819

<table>
<thead>
<tr>
<th>Mean Tardiness</th>
<th>WIP Inventory</th>
<th>Resource Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>2.100 - 3.860</td>
<td>2704</td>
<td>4.898 - 5.025</td>
</tr>
<tr>
<td>3.860 - 5.620</td>
<td>45008</td>
<td>5.025 - 5.154</td>
</tr>
<tr>
<td>5.620 - 7.380</td>
<td>210788</td>
<td>5.164 - 5.284</td>
</tr>
<tr>
<td>7.380 - 9.140</td>
<td>592224</td>
<td>5.284 - 5.413</td>
</tr>
<tr>
<td>9.140 - 10.900</td>
<td>978628</td>
<td>5.413 - 5.542</td>
</tr>
<tr>
<td>10.900 - 12.660</td>
<td>942894</td>
<td>5.542 - 5.671</td>
</tr>
<tr>
<td>12.660 - 14.420</td>
<td>811102</td>
<td>5.671 - 5.800</td>
</tr>
<tr>
<td>14.420 - 16.180</td>
<td>197484</td>
<td>5.800 - 5.930</td>
</tr>
<tr>
<td>16.180 - 17.940</td>
<td>41062</td>
<td>5.930 - 6.059</td>
</tr>
<tr>
<td>17.940 - 19.700</td>
<td>2848</td>
<td>6.059 - 6.188</td>
</tr>
</tbody>
</table>

Mean = 10.894
Variance = 5.857
Std.Dev. = 2.378

<table>
<thead>
<tr>
<th>Throughput</th>
<th>No. of Tardy jobs</th>
<th>Summation of *</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>0.135 - 0.139</td>
<td>189832</td>
<td>2.000 - 2.600</td>
</tr>
<tr>
<td>0.139 - 0.143</td>
<td>512460</td>
<td>2.600 - 3.200</td>
</tr>
<tr>
<td>0.143 - 0.147</td>
<td>431424</td>
<td>3.200 - 3.800</td>
</tr>
<tr>
<td>0.147 - 0.151</td>
<td>902880</td>
<td>3.800 - 4.400</td>
</tr>
<tr>
<td>0.151 - 0.155</td>
<td>865152</td>
<td>4.400 - 5.000</td>
</tr>
<tr>
<td>0.155 - 0.159</td>
<td>516808</td>
<td>5.000 - 5.600</td>
</tr>
<tr>
<td>0.159 - 0.163</td>
<td>98486</td>
<td>5.600 - 6.200</td>
</tr>
<tr>
<td>0.163 - 0.167</td>
<td>114048</td>
<td>6.200 - 6.800</td>
</tr>
<tr>
<td>0.167 - 0.171</td>
<td>8840</td>
<td>6.800 - 7.400</td>
</tr>
<tr>
<td>0.171 - 0.175</td>
<td>10080</td>
<td>7.400 - 8.000</td>
</tr>
</tbody>
</table>

Mean = 0.150
Variance = 0.000
Std.Dev. = 0.007

Mean = 5.353
Variance = 89.808
Std.Dev. = 9.364
<table>
<thead>
<tr>
<th>Data Group#18</th>
<th>Maximum Flow Time</th>
<th>* Mean Flow Time</th>
<th>* Maximum Tardiness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
<td>Numbers</td>
</tr>
<tr>
<td>68.000- 70.100</td>
<td>4942</td>
<td>50.200- 51.420</td>
<td>2976</td>
</tr>
<tr>
<td>70.100- 74.200</td>
<td>44890</td>
<td>51.420- 52.840</td>
<td>28932</td>
</tr>
<tr>
<td>74.200- 78.300</td>
<td>150408</td>
<td>52.640- 53.880</td>
<td>164388</td>
</tr>
<tr>
<td>78.300- 82.400</td>
<td>384438</td>
<td>53.880- 55.080</td>
<td>421536</td>
</tr>
<tr>
<td>82.400- 86.500</td>
<td>674132</td>
<td>55.080- 56.500</td>
<td>818112</td>
</tr>
<tr>
<td>86.500- 90.600</td>
<td>841180</td>
<td>56.500- 57.520</td>
<td>900460</td>
</tr>
<tr>
<td>90.600- 94.700</td>
<td>701112</td>
<td>57.520- 58.740</td>
<td>737376</td>
</tr>
<tr>
<td>94.700- 98.800</td>
<td>471224</td>
<td>58.740- 59.960</td>
<td>404640</td>
</tr>
<tr>
<td>98.800- 102.900</td>
<td>259844</td>
<td>59.960- 61.180</td>
<td>137184</td>
</tr>
<tr>
<td>102.900- 107.000</td>
<td>88832</td>
<td>61.180- 62.400</td>
<td>23136</td>
</tr>
<tr>
<td>Mean = 89.225</td>
<td>Mean = 56.850</td>
<td>Mean = 47.162</td>
<td></td>
</tr>
<tr>
<td>Variance = 45.781</td>
<td>Variance = 3.296</td>
<td>Variance = 75.024</td>
<td></td>
</tr>
<tr>
<td>Std.Dev = 6.786</td>
<td>Std.Dev. = 1.815</td>
<td>Std.Dev. = 8.862</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mean Tardiness</th>
<th>WIP Inventory</th>
<th>Resource Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>6.600- 8.260</td>
<td>1782</td>
<td>4.891- 5.011</td>
</tr>
<tr>
<td>8.260- 9.920</td>
<td>30312</td>
<td>5.011- 5.132</td>
</tr>
<tr>
<td>9.920- 11.580</td>
<td>164420</td>
<td>5.132- 5.252</td>
</tr>
<tr>
<td>11.580- 13.240</td>
<td>517722</td>
<td>5.252- 5.372</td>
</tr>
<tr>
<td>13.240- 14.900</td>
<td>901708</td>
<td>5.372- 5.493</td>
</tr>
<tr>
<td>14.900- 16.560</td>
<td>937394</td>
<td>5.493- 5.613</td>
</tr>
<tr>
<td>16.560- 18.220</td>
<td>692168</td>
<td>5.613- 5.733</td>
</tr>
<tr>
<td>18.220- 19.880</td>
<td>297348</td>
<td>5.733- 5.853</td>
</tr>
<tr>
<td>19.880- 21.540</td>
<td>77758</td>
<td>5.853- 5.974</td>
</tr>
<tr>
<td>21.540- 23.200</td>
<td>81912</td>
<td>5.974- 6.094</td>
</tr>
<tr>
<td>Mean = 15.282</td>
<td>Mean = 5.489</td>
<td>Mean = 0.749</td>
</tr>
<tr>
<td>Variance = 5.399</td>
<td>Variance = 0.029</td>
<td>Variance = 0.001</td>
</tr>
<tr>
<td>Std.Dev = 2.324</td>
<td>Std.Dev. = 0.170</td>
<td>Std.Dev. = 0.031</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Throughput</th>
<th>No. of Tardy jobs</th>
<th>Summation of *</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>0.139- 0.143</td>
<td>220320</td>
<td>4.000- 4.500</td>
</tr>
<tr>
<td>0.143- 0.148</td>
<td>629568</td>
<td>4.500- 5.000</td>
</tr>
<tr>
<td>0.148- 0.152</td>
<td>1020384</td>
<td>5.000- 5.500</td>
</tr>
<tr>
<td>0.152- 0.156</td>
<td>498240</td>
<td>5.500- 6.000</td>
</tr>
<tr>
<td>0.156- 0.160</td>
<td>757824</td>
<td>6.000- 6.500</td>
</tr>
<tr>
<td>0.160- 0.165</td>
<td>380064</td>
<td>6.500- 7.000</td>
</tr>
<tr>
<td>0.165- 0.169</td>
<td>68048</td>
<td>7.000- 7.500</td>
</tr>
<tr>
<td>0.169- 0.173</td>
<td>50868</td>
<td>7.500- 8.000</td>
</tr>
<tr>
<td>0.173- 0.178</td>
<td>3456</td>
<td>8.000- 8.500</td>
</tr>
<tr>
<td>0.178- 0.182</td>
<td>2208</td>
<td>8.500- 9.000</td>
</tr>
<tr>
<td>Mean = 0.153</td>
<td>Mean = 6.458</td>
<td>Mean = 104.012</td>
</tr>
<tr>
<td>Variance = 0.000</td>
<td>Variance = 0.708</td>
<td>Variance = 91.136</td>
</tr>
<tr>
<td>Std.Dev. = 0.008</td>
<td>Std.Dev. = 0.841</td>
<td>Std.Dev. = 8.547</td>
</tr>
<tr>
<td>Data Group#19</td>
<td>* Mean Flow Time</td>
<td>* Maximum Tardiness</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>Maximum Flow Time</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>68.000- 72.100</td>
<td>1512</td>
<td>52.900- 54.200</td>
</tr>
<tr>
<td>72.100- 76.200</td>
<td>15324</td>
<td>54.200- 55.500</td>
</tr>
<tr>
<td>76.200- 80.300</td>
<td>88744</td>
<td>55.500- 56.800</td>
</tr>
<tr>
<td>80.300- 84.400</td>
<td>265080</td>
<td>56.800- 58.100</td>
</tr>
<tr>
<td>84.400- 88.500</td>
<td>605104</td>
<td>58.100- 59.400</td>
</tr>
<tr>
<td>88.500- 92.600</td>
<td>795776</td>
<td>59.400- 60.700</td>
</tr>
<tr>
<td>92.600- 96.700</td>
<td>897468</td>
<td>60.700- 62.000</td>
</tr>
<tr>
<td>96.700- 100.800</td>
<td>825236</td>
<td>62.000- 63.300</td>
</tr>
<tr>
<td>100.800- 104.900</td>
<td>353588</td>
<td>63.300- 64.600</td>
</tr>
<tr>
<td>104.900- 109.000</td>
<td>91008</td>
<td>64.600- 65.900</td>
</tr>
</tbody>
</table>

Mean = 92.955  Mean = 59.860  Mean = 51.086
Variance = 39.744  Variance = 3.858  Variance = 88.529

<table>
<thead>
<tr>
<th>Mean Tardiness</th>
<th>WIP Inventory</th>
<th>Resource Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>8.200- 9.850</td>
<td>1288</td>
<td>4.905- 5.031</td>
</tr>
<tr>
<td>9.850- 11.500</td>
<td>15762</td>
<td>5.031- 5.156</td>
</tr>
<tr>
<td>11.500- 13.150</td>
<td>110536</td>
<td>5.156- 5.282</td>
</tr>
<tr>
<td>13.150- 14.800</td>
<td>400506</td>
<td>5.282- 5.407</td>
</tr>
<tr>
<td>14.800- 16.450</td>
<td>748324</td>
<td>5.407- 5.533</td>
</tr>
<tr>
<td>16.450- 18.100</td>
<td>1025590</td>
<td>5.533- 5.659</td>
</tr>
<tr>
<td>18.100- 19.750</td>
<td>814788</td>
<td>5.659- 5.784</td>
</tr>
<tr>
<td>19.750- 21.400</td>
<td>410474</td>
<td>5.784- 5.910</td>
</tr>
<tr>
<td>21.400- 23.050</td>
<td>93304</td>
<td>5.910- 6.035</td>
</tr>
<tr>
<td>23.050- 24.700</td>
<td>8248</td>
<td>6.035- 6.161</td>
</tr>
</tbody>
</table>

Mean = 17.297  Mean = 5.481  Mean = 0.750
Variance = 5.080  Variance = 0.038  Variance = 0.001
Std.Dev = 2.254  Std.Dev = 0.195  Std.Dev = 0.030

<table>
<thead>
<tr>
<th>Throughput</th>
<th>No. of Tardy jobs</th>
<th>Summation of *</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>0.139- 0.143</td>
<td>180864</td>
<td>4.000- 4.600</td>
</tr>
<tr>
<td>0.143- 0.148</td>
<td>593280</td>
<td>4.600- 5.200</td>
</tr>
<tr>
<td>0.148- 0.152</td>
<td>1006120</td>
<td>5.200- 5.800</td>
</tr>
<tr>
<td>0.152- 0.156</td>
<td>578496</td>
<td>5.800- 6.400</td>
</tr>
<tr>
<td>0.156- 0.160</td>
<td>722982</td>
<td>6.400- 7.000</td>
</tr>
<tr>
<td>0.160- 0.165</td>
<td>379800</td>
<td>7.000- 7.600</td>
</tr>
<tr>
<td>0.165- 0.169</td>
<td>63284</td>
<td>7.600- 8.200</td>
</tr>
<tr>
<td>0.169- 0.173</td>
<td>51186</td>
<td>8.200- 8.800</td>
</tr>
<tr>
<td>0.173- 0.178</td>
<td>2688</td>
<td>8.800- 9.400</td>
</tr>
<tr>
<td>0.178- 0.182</td>
<td>1248</td>
<td>9.400- 10.000</td>
</tr>
</tbody>
</table>

Mean = 0.153  Mean = 6.847  Mean = 110.818
Variance = 0.000  Variance = 0.713  Variance = 115.270
Std.Dev = 0.006  Std.Dev = 0.844  Std.Dev = 10.736
<table>
<thead>
<tr>
<th>Data Group#20</th>
<th>* Mean Flow Time</th>
<th>* Maximum Tardiness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>68.000-72.500</td>
<td>935</td>
<td>54.800-56.130</td>
</tr>
<tr>
<td>72.500-77.000</td>
<td>7441</td>
<td>56.130-57.480</td>
</tr>
<tr>
<td>77.000-81.500</td>
<td>54000</td>
<td>57.480-58.790</td>
</tr>
<tr>
<td>81.500-86.000</td>
<td>142168</td>
<td>58.790-60.120</td>
</tr>
<tr>
<td>86.000-90.500</td>
<td>482638</td>
<td>60.120-61.450</td>
</tr>
<tr>
<td>90.500-95.000</td>
<td>803186</td>
<td>61.450-62.780</td>
</tr>
<tr>
<td>95.000-99.500</td>
<td>988132</td>
<td>62.780-64.110</td>
</tr>
<tr>
<td>99.500-104.000</td>
<td>744084</td>
<td>64.110-65.440</td>
</tr>
<tr>
<td>104.000-108.500</td>
<td>530984</td>
<td>65.440-66.770</td>
</tr>
<tr>
<td>108.500-113.000</td>
<td>86284</td>
<td>66.770-68.100</td>
</tr>
<tr>
<td>Mean = 96.814</td>
<td>Mean = 82.150</td>
<td>Mean = 53.019</td>
</tr>
<tr>
<td>Variance = 45.448</td>
<td>Variance = 3.912</td>
<td>Variance = 114.296</td>
</tr>
<tr>
<td>Std.Dev. = 6.742</td>
<td>Std.Dev. = 1.978</td>
<td>Std.Dev. = 10.691</td>
</tr>
<tr>
<td>Mean Tardiness</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>8.400-10.200</td>
<td>1544</td>
<td>4.924-5.046</td>
</tr>
<tr>
<td>10.200-12.000</td>
<td>20988</td>
<td>5.046-5.186</td>
</tr>
<tr>
<td>12.000-13.800</td>
<td>133924</td>
<td>5.188-5.290</td>
</tr>
<tr>
<td>13.800-15.600</td>
<td>442770</td>
<td>5.290-5.412</td>
</tr>
<tr>
<td>15.600-17.400</td>
<td>827904</td>
<td>5.412-5.533</td>
</tr>
<tr>
<td>17.400-19.200</td>
<td>981344</td>
<td>5.533-5.655</td>
</tr>
<tr>
<td>19.200-21.000</td>
<td>764750</td>
<td>5.655-5.777</td>
</tr>
<tr>
<td>21.000-22.800</td>
<td>360210</td>
<td>5.777-5.889</td>
</tr>
<tr>
<td>22.800-24.600</td>
<td>88092</td>
<td>5.889-6.021</td>
</tr>
<tr>
<td>24.600-26.400</td>
<td>7276</td>
<td>6.021-6.143</td>
</tr>
<tr>
<td>Mean = 18.038</td>
<td>Mean = 5.498</td>
<td>Mean = 0.745</td>
</tr>
<tr>
<td>Variance = 8.191</td>
<td>Variance = 0.033</td>
<td>Variance = 0.001</td>
</tr>
<tr>
<td>Std.Dev. = 2.488</td>
<td>Std.Dev. = 0.181</td>
<td>Std.Dev. = 0.031</td>
</tr>
<tr>
<td>Throughput</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>0.135-0.139</td>
<td>181512</td>
<td>4.000-4.500</td>
</tr>
<tr>
<td>0.139-0.144</td>
<td>573336</td>
<td>4.500-5.000</td>
</tr>
<tr>
<td>0.144-0.148</td>
<td>978120</td>
<td>5.000-5.500</td>
</tr>
<tr>
<td>0.148-0.153</td>
<td>938520</td>
<td>5.500-6.000</td>
</tr>
<tr>
<td>0.153-0.157</td>
<td>607248</td>
<td>6.000-6.500</td>
</tr>
<tr>
<td>0.157-0.161</td>
<td>272664</td>
<td>6.500-7.000</td>
</tr>
<tr>
<td>0.161-0.166</td>
<td>41904</td>
<td>7.000-7.500</td>
</tr>
<tr>
<td>0.166-0.170</td>
<td>32616</td>
<td>7.500-8.000</td>
</tr>
<tr>
<td>0.170-0.175</td>
<td>1944</td>
<td>8.000-8.500</td>
</tr>
<tr>
<td>0.175-0.179</td>
<td>938</td>
<td>8.500-9.000</td>
</tr>
<tr>
<td>Mean = 0.149</td>
<td>Mean = 6.770</td>
<td>Mean = 115.189</td>
</tr>
<tr>
<td>Variance = 0.000</td>
<td>Variance = 0.773</td>
<td>Variance = 138.382</td>
</tr>
<tr>
<td>Std.Dev. = 0.008</td>
<td>Std.Dev. = 0.879</td>
<td>Std.Dev. = 11.678</td>
</tr>
<tr>
<td>Data Group#21</td>
<td>* Mean Flow Time</td>
<td>* Maximum Tardiness</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>85.000-91.100</td>
<td>1700</td>
<td>60.000-62.610</td>
</tr>
<tr>
<td>91.100-97.200</td>
<td>16044</td>
<td>62.610-64.720</td>
</tr>
<tr>
<td>97.200-103.300</td>
<td>77832</td>
<td>64.720-66.830</td>
</tr>
<tr>
<td>103.300-109.400</td>
<td>248120</td>
<td>66.830-68.940</td>
</tr>
<tr>
<td>109.400-115.500</td>
<td>506796</td>
<td>68.940-71.050</td>
</tr>
<tr>
<td>115.500-121.800</td>
<td>805648</td>
<td>71.050-73.160</td>
</tr>
<tr>
<td>121.800-127.700</td>
<td>910008</td>
<td>73.160-75.270</td>
</tr>
<tr>
<td>127.700-133.800</td>
<td>599324</td>
<td>75.270-77.380</td>
</tr>
<tr>
<td>133.800-139.900</td>
<td>304804</td>
<td>77.380-79.490</td>
</tr>
<tr>
<td>139.900-146.000</td>
<td>158364</td>
<td>79.490-81.600</td>
</tr>
</tbody>
</table>

Mean = 122.398
Mean = 73.100
Mean = 87.336

Variance = 92.057
Variance = 10.953
Variance = 199.505

Std. Dev. = 9.585
Std. Dev. = 3.310
Std. Dev. = 14.125

Mean Tardiness | WIP Inventory | Resource Utilization |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>10.400-12.540</td>
<td>470</td>
<td>4.560-6.491</td>
</tr>
<tr>
<td>12.540-14.680</td>
<td>9008</td>
<td>4.691-4.882</td>
</tr>
<tr>
<td>16.820-18.960</td>
<td>269163</td>
<td>5.073-5.264</td>
</tr>
<tr>
<td>18.960-21.100</td>
<td>601555</td>
<td>5.264-5.455</td>
</tr>
<tr>
<td>21.100-23.240</td>
<td>969833</td>
<td>5.464-5.646</td>
</tr>
<tr>
<td>23.240-25.380</td>
<td>968100</td>
<td>5.646-5.837</td>
</tr>
<tr>
<td>25.380-27.520</td>
<td>594906</td>
<td>5.837-6.028</td>
</tr>
<tr>
<td>27.520-29.660</td>
<td>167601</td>
<td>6.028-6.219</td>
</tr>
<tr>
<td>29.660-31.800</td>
<td>94022</td>
<td>6.219-6.410</td>
</tr>
</tbody>
</table>

Mean = 22.929
Mean = 5.471
Mean = 0.736

Variance = 8.408
Variance = 0.123
Variance = 0.001

Std. Dev. = 2.900
Std. Dev. = 0.351
Std. Dev. = 0.024

Throughput | No. of Tardy jobs | Summation of |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>0.106-0.109</td>
<td>38328</td>
<td>3.000-3.700</td>
</tr>
<tr>
<td>0.109-0.111</td>
<td>783960</td>
<td>3.700-4.400</td>
</tr>
<tr>
<td>0.111-0.114</td>
<td>932520</td>
<td>4.400-5.100</td>
</tr>
<tr>
<td>0.114-0.116</td>
<td>928944</td>
<td>5.100-5.800</td>
</tr>
<tr>
<td>0.116-0.119</td>
<td>319344</td>
<td>5.800-6.500</td>
</tr>
<tr>
<td>0.119-0.122</td>
<td>411432</td>
<td>6.500-7.200</td>
</tr>
<tr>
<td>0.122-0.124</td>
<td>156192</td>
<td>7.200-7.900</td>
</tr>
<tr>
<td>0.124-0.127</td>
<td>48264</td>
<td>7.900-8.600</td>
</tr>
<tr>
<td>0.127-0.129</td>
<td>6072</td>
<td>8.600-9.300</td>
</tr>
<tr>
<td>0.129-0.132</td>
<td>3744</td>
<td>9.300-10.000</td>
</tr>
</tbody>
</table>

Mean = 0.115
Mean = 0.631
Mean = 140.436

Variance = 0.000
Variance = 0.780
Variance = 243.106

Std. Dev. = 0.004
Std. Dev. = 0.883
Std. Dev. = 15.592
<table>
<thead>
<tr>
<th>Data Group#22</th>
<th>Maximum Flow Time</th>
<th>* Mean Flow Time</th>
<th>* Maximum Tardiness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
<td>Numbers</td>
</tr>
<tr>
<td>87.000 - 93.500</td>
<td>988</td>
<td>85.100 - 87.240</td>
<td>5588</td>
</tr>
<tr>
<td>93.500 - 100.000</td>
<td>9874</td>
<td>67.240 - 69.380</td>
<td>49728</td>
</tr>
<tr>
<td>100.000 - 108.500</td>
<td>66228</td>
<td>69.380 - 71.520</td>
<td>189024</td>
</tr>
<tr>
<td>108.500 - 113.000</td>
<td>139742</td>
<td>71.520 - 73.680</td>
<td>399300</td>
</tr>
<tr>
<td>113.000 - 119.500</td>
<td>352884</td>
<td>73.680 - 76.800</td>
<td>621024</td>
</tr>
<tr>
<td>119.500 - 126.000</td>
<td>494504</td>
<td>76.800 - 77.940</td>
<td>822144</td>
</tr>
<tr>
<td>126.000 - 132.500</td>
<td>837800</td>
<td>77.940 - 80.080</td>
<td>758208</td>
</tr>
<tr>
<td>132.500 - 139.000</td>
<td>907938</td>
<td>80.080 - 82.220</td>
<td>530988</td>
</tr>
<tr>
<td>139.000 - 145.500</td>
<td>650808</td>
<td>82.220 - 84.360</td>
<td>212448</td>
</tr>
<tr>
<td>145.500 - 152.000</td>
<td>168824</td>
<td>84.360 - 86.500</td>
<td>40808</td>
</tr>
<tr>
<td>Mean = 130.383</td>
<td>Mean = 77.100</td>
<td>Mean = 65.292</td>
<td></td>
</tr>
<tr>
<td>Variance = 104.777</td>
<td>Variance = 12.550</td>
<td>Variance = 84.013</td>
<td></td>
</tr>
<tr>
<td>Std.Dev = 10.238</td>
<td>Std.Dev. = 3.543</td>
<td>Std.Dev. = 8.898</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mean Tardiness</th>
<th>WIP Inventory</th>
<th>Resource Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>11.800 - 13.900</td>
<td>878</td>
<td>4.667 - 4.850</td>
</tr>
<tr>
<td>13.900 - 16.000</td>
<td>10312</td>
<td>4.860 - 5.032</td>
</tr>
<tr>
<td>18.000 - 18.100</td>
<td>88904</td>
<td>5.032 - 5.215</td>
</tr>
<tr>
<td>18.100 - 20.200</td>
<td>284682</td>
<td>5.215 - 5.388</td>
</tr>
<tr>
<td>20.200 - 22.300</td>
<td>699442</td>
<td>5.388 - 5.581</td>
</tr>
<tr>
<td>22.300 - 24.400</td>
<td>987910</td>
<td>5.581 - 5.763</td>
</tr>
<tr>
<td>24.400 - 28.500</td>
<td>937174</td>
<td>5.763 - 5.948</td>
</tr>
<tr>
<td>26.500 - 28.600</td>
<td>497186</td>
<td>5.948 - 6.129</td>
</tr>
<tr>
<td>28.800 - 30.700</td>
<td>145432</td>
<td>6.129 - 6.311</td>
</tr>
<tr>
<td>30.700 - 32.800</td>
<td>17592</td>
<td>6.311 - 6.494</td>
</tr>
<tr>
<td>Mean = 23.878</td>
<td>Mean = 5.532</td>
<td>Mean = 0.740</td>
</tr>
<tr>
<td>Variance = 7.868</td>
<td>Variance = 0.123</td>
<td>Variance = 0.001</td>
</tr>
<tr>
<td>Std.Dev. = 2.803</td>
<td>Std.Dev. = 0.350</td>
<td>Std.Dev. = 0.028</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Throughput</th>
<th>No. of Tardy jobs</th>
<th>Summation of *</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>0.103 - 0.106</td>
<td>138800</td>
<td>3.000 - 3.700</td>
</tr>
<tr>
<td>0.106 - 0.108</td>
<td>408980</td>
<td>3.700 - 4.400</td>
</tr>
<tr>
<td>0.108 - 0.110</td>
<td>807264</td>
<td>4.400 - 5.100</td>
</tr>
<tr>
<td>0.110 - 0.113</td>
<td>870824</td>
<td>5.100 - 5.800</td>
</tr>
<tr>
<td>0.113 - 0.118</td>
<td>761268</td>
<td>5.800 - 6.500</td>
</tr>
<tr>
<td>0.116 - 0.118</td>
<td>328484</td>
<td>6.500 - 7.200</td>
</tr>
<tr>
<td>0.118 - 0.120</td>
<td>185760</td>
<td>7.200 - 7.900</td>
</tr>
<tr>
<td>0.120 - 0.123</td>
<td>409924</td>
<td>7.900 - 8.600</td>
</tr>
<tr>
<td>0.123 - 0.125</td>
<td>289986</td>
<td>8.600 - 9.300</td>
</tr>
<tr>
<td>0.125 - 0.128</td>
<td>6528</td>
<td>9.300 - 10.000</td>
</tr>
<tr>
<td>Mean = 0.112</td>
<td>Mean = 7.124</td>
<td>Mean = 142.392</td>
</tr>
<tr>
<td>Variance = 0.000</td>
<td>Variance = 0.883</td>
<td>Variance = 115.637</td>
</tr>
<tr>
<td>Std.Dev. = 0.004</td>
<td>Std.Dev. = 0.881</td>
<td>Std.Dev. = 10.753</td>
</tr>
<tr>
<td>Data Group#23</td>
<td>* Mean Flow Time</td>
<td>* Maximum Tardiness</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>91.000- 98.200</td>
<td>1777</td>
<td>88.100- 70.540</td>
</tr>
<tr>
<td>98.200- 105.400</td>
<td>17821</td>
<td>70.540- 72.980</td>
</tr>
<tr>
<td>105.400- 112.800</td>
<td>75121</td>
<td>72.980- 75.420</td>
</tr>
<tr>
<td>112.800- 119.800</td>
<td>229398</td>
<td>75.420- 77.880</td>
</tr>
<tr>
<td>119.800- 127.000</td>
<td>384591</td>
<td>77.880- 80.300</td>
</tr>
<tr>
<td>127.000- 134.200</td>
<td>692850</td>
<td>80.300- 82.740</td>
</tr>
<tr>
<td>134.200- 141.400</td>
<td>834000</td>
<td>82.740- 85.180</td>
</tr>
<tr>
<td>141.400- 148.600</td>
<td>748082</td>
<td>85.180- 87.820</td>
</tr>
<tr>
<td>148.800- 155.800</td>
<td>742512</td>
<td>87.820- 90.060</td>
</tr>
<tr>
<td>155.800- 163.000</td>
<td>204840</td>
<td>90.060- 92.500</td>
</tr>
<tr>
<td>Mean = 138.488</td>
<td>Mean = 82.450</td>
<td>Mean = 73.780</td>
</tr>
<tr>
<td>Variance = 156.110</td>
<td>Variance = 14.603</td>
<td>Variance = 88.522</td>
</tr>
<tr>
<td>Std.Dev. = 12.814</td>
<td>Std.Dev. = 3.821</td>
<td>Std.Dev. = 9.409</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mean Tardiness</th>
<th>WIP Inventory</th>
<th>Resource Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>11.900- 14.870</td>
<td>1380</td>
<td>4.602- 4.791</td>
</tr>
<tr>
<td>14.870- 17.840</td>
<td>24332</td>
<td>4.791- 4.879</td>
</tr>
<tr>
<td>17.840- 20.810</td>
<td>153910</td>
<td>4.879- 5.188</td>
</tr>
<tr>
<td>20.810- 23.780</td>
<td>449840</td>
<td>5.188- 5.358</td>
</tr>
<tr>
<td>23.780- 26.750</td>
<td>787144</td>
<td>5.358- 5.645</td>
</tr>
<tr>
<td>26.750- 29.720</td>
<td>891982</td>
<td>5.645- 5.734</td>
</tr>
<tr>
<td>29.720- 32.890</td>
<td>791780</td>
<td>5.734- 5.822</td>
</tr>
<tr>
<td>32.690- 35.860</td>
<td>433976</td>
<td>5.822- 6.111</td>
</tr>
<tr>
<td>35.660- 38.830</td>
<td>88716</td>
<td>6.111- 6.289</td>
</tr>
<tr>
<td>38.630- 41.800</td>
<td>5780</td>
<td>6.289- 6.488</td>
</tr>
<tr>
<td>Mean = 27.984</td>
<td>Mean = 5.478</td>
<td>Mean = 0.759</td>
</tr>
<tr>
<td>Variance = 18.154</td>
<td>Variance = 0.120</td>
<td>Variance = 0.001</td>
</tr>
<tr>
<td>Std.Dev. = 4.281</td>
<td>Std.Dev. = 0.348</td>
<td>Std.Dev. = 0.028</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Throughput</th>
<th>No. of Tardy jobs</th>
<th>Summation of *</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>0.087- 0.100</td>
<td>190920</td>
<td>3.000- 3.700</td>
</tr>
<tr>
<td>0.100- 0.102</td>
<td>728480</td>
<td>3.700- 4.400</td>
</tr>
<tr>
<td>0.102- 0.105</td>
<td>811658</td>
<td>4.400- 5.100</td>
</tr>
<tr>
<td>0.105- 0.107</td>
<td>765144</td>
<td>5.100- 5.800</td>
</tr>
<tr>
<td>0.107- 0.110</td>
<td>798896</td>
<td>5.800- 6.500</td>
</tr>
<tr>
<td>0.110- 0.113</td>
<td>243576</td>
<td>6.500- 7.200</td>
</tr>
<tr>
<td>0.113- 0.115</td>
<td>88856</td>
<td>7.200- 7.900</td>
</tr>
<tr>
<td>0.115- 0.118</td>
<td>28424</td>
<td>7.900- 8.600</td>
</tr>
<tr>
<td>0.118- 0.120</td>
<td>5832</td>
<td>8.600- 9.300</td>
</tr>
<tr>
<td>0.120- 0.123</td>
<td>2018</td>
<td>9.300- 10.000</td>
</tr>
<tr>
<td>Mean = 0.105</td>
<td>Mean = 7.256</td>
<td>Mean = 156.210</td>
</tr>
<tr>
<td>Variance = 0.000</td>
<td>Variance = 1.100</td>
<td>Variance = 112.006</td>
</tr>
<tr>
<td>Std.Dev. = 0.004</td>
<td>Std.Dev. = 1.048</td>
<td>Std.Dev. = 10.583</td>
</tr>
<tr>
<td>Data Group#24</td>
<td>Maximum Flow Time</td>
<td>* Mean Flow Time</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>90.300- 97.300</td>
<td>619</td>
<td>74.200- 78.110</td>
</tr>
<tr>
<td>97.300- 104.800</td>
<td>10232</td>
<td>78.110- 78.020</td>
</tr>
<tr>
<td>104.800-111.900</td>
<td>59845</td>
<td>78.020- 79.830</td>
</tr>
<tr>
<td>111.900-119.200</td>
<td>221087</td>
<td>79.930- 81.840</td>
</tr>
<tr>
<td>119.200-126.500</td>
<td>441267</td>
<td>81.840- 83.750</td>
</tr>
<tr>
<td>126.500-133.800</td>
<td>672670</td>
<td>83.760- 85.680</td>
</tr>
<tr>
<td>133.800-141.100</td>
<td>692758</td>
<td>85.680- 87.570</td>
</tr>
<tr>
<td>141.100-148.400</td>
<td>578040</td>
<td>87.570- 89.480</td>
</tr>
<tr>
<td>148.400-155.700</td>
<td>602552</td>
<td>89.480- 91.390</td>
</tr>
<tr>
<td>155.700-163.000</td>
<td>453144</td>
<td>91.380- 93.300</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
<td><strong>138.819</strong></td>
<td><strong>Mean</strong></td>
</tr>
<tr>
<td><strong>Variance</strong></td>
<td><strong>177.280</strong></td>
<td><strong>Variance</strong></td>
</tr>
<tr>
<td><strong>Std.Dev.</strong></td>
<td><strong>13.314</strong></td>
<td><strong>Std.Dev.</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mean Tardiness</th>
<th>WIP Inventory</th>
<th>Resource Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>18.000- 21.550</td>
<td>3040</td>
<td>4.631- 4.800</td>
</tr>
<tr>
<td>21.550- 24.100</td>
<td>48552</td>
<td>4.800- 4.999</td>
</tr>
<tr>
<td>24.100- 26.850</td>
<td>243516</td>
<td>4.889- 5.138</td>
</tr>
<tr>
<td>26.850- 29.200</td>
<td>513582</td>
<td>5.138- 5.307</td>
</tr>
<tr>
<td>29.200- 31.750</td>
<td>740288</td>
<td>5.307- 5.476</td>
</tr>
<tr>
<td>31.750- 34.300</td>
<td>788154</td>
<td>5.476- 5.645</td>
</tr>
<tr>
<td>34.300- 36.850</td>
<td>771112</td>
<td>5.645- 5.814</td>
</tr>
<tr>
<td>36.850- 39.400</td>
<td>403612</td>
<td>5.814- 5.983</td>
</tr>
<tr>
<td>39.400- 41.950</td>
<td>102732</td>
<td>5.983- 6.152</td>
</tr>
<tr>
<td>41.950- 44.500</td>
<td>82332</td>
<td>6.152- 6.321</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
<td><strong>32.443</strong></td>
<td><strong>Mean</strong></td>
</tr>
<tr>
<td><strong>Variance</strong></td>
<td><strong>16.790</strong></td>
<td><strong>Variance</strong></td>
</tr>
<tr>
<td><strong>Std.Dev.</strong></td>
<td><strong>3.974</strong></td>
<td><strong>Std.Dev.</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Throughput</th>
<th>No. of Tardy jobs</th>
<th>Summation of *</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervals</td>
<td>Numbers</td>
<td>Intervals</td>
</tr>
<tr>
<td>0.106- 0.109</td>
<td>278840</td>
<td>5.000- 5.500</td>
</tr>
<tr>
<td>0.109- 0.111</td>
<td>708912</td>
<td>5.500- 6.000</td>
</tr>
<tr>
<td>0.111- 0.114</td>
<td>880352</td>
<td>6.000- 6.500</td>
</tr>
<tr>
<td>0.114- 0.117</td>
<td>794448</td>
<td>6.500- 7.000</td>
</tr>
<tr>
<td>0.117- 0.119</td>
<td>568398</td>
<td>7.000- 7.500</td>
</tr>
<tr>
<td>0.119- 0.122</td>
<td>254738</td>
<td>7.500- 8.000</td>
</tr>
<tr>
<td>0.122- 0.125</td>
<td>69894</td>
<td>8.000- 8.500</td>
</tr>
<tr>
<td>0.125- 0.128</td>
<td>47376</td>
<td>8.500- 9.000</td>
</tr>
<tr>
<td>0.128- 0.130</td>
<td>13824</td>
<td>9.000- 9.500</td>
</tr>
<tr>
<td>0.130- 0.133</td>
<td>2180</td>
<td>9.500- 10.000</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
<td><strong>0.115</strong></td>
<td><strong>Mean</strong></td>
</tr>
<tr>
<td><strong>Variance</strong></td>
<td><strong>0.000</strong></td>
<td><strong>Variance</strong></td>
</tr>
<tr>
<td><strong>Std.Dev.</strong></td>
<td><strong>0.004</strong></td>
<td><strong>Std.Dev.</strong></td>
</tr>
<tr>
<td>Data Group#25</td>
<td>* Mean Flow Time</td>
<td>* Maximum Tardiness</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td><strong>Intervals</strong></td>
<td><strong>Numbers</strong></td>
<td><strong>Intervals</strong></td>
</tr>
<tr>
<td>96.000 - 103.200</td>
<td>2875</td>
<td>73.100 - 75.280</td>
</tr>
<tr>
<td>103.200 - 110.400</td>
<td>44155</td>
<td>77.280 - 77.480</td>
</tr>
<tr>
<td>110.400 - 117.600</td>
<td>164924</td>
<td>77.480 - 78.870</td>
</tr>
<tr>
<td>117.600 - 124.800</td>
<td>376524</td>
<td>79.670 - 81.880</td>
</tr>
<tr>
<td>124.800 - 132.000</td>
<td>635878</td>
<td>81.880 - 84.050</td>
</tr>
<tr>
<td>132.000 - 139.200</td>
<td>820398</td>
<td>84.050 - 86.240</td>
</tr>
<tr>
<td>139.200 - 146.400</td>
<td>558564</td>
<td>86.240 - 88.430</td>
</tr>
<tr>
<td>146.400 - 153.600</td>
<td>439640</td>
<td>88.430 - 90.620</td>
</tr>
<tr>
<td>153.600 - 160.800</td>
<td>340100</td>
<td>90.620 - 92.810</td>
</tr>
<tr>
<td>160.800 - 168.000</td>
<td>248764</td>
<td>92.810 - 95.000</td>
</tr>
</tbody>
</table>

**Mean = 138.025**  
**Variance = 183.879**  
**Std.Dev. = 13.560**

<table>
<thead>
<tr>
<th>Mean Tardiness</th>
<th>WIP Inventory</th>
<th>Resource Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Intervals</strong></td>
<td><strong>Numbers</strong></td>
<td><strong>Intervals</strong></td>
</tr>
<tr>
<td>21.800 - 24.320</td>
<td>6764</td>
<td>4.470 - 4.860</td>
</tr>
<tr>
<td>24.320 - 26.840</td>
<td>81448</td>
<td>4.860 - 4.891</td>
</tr>
<tr>
<td>26.840 - 29.360</td>
<td>284088</td>
<td>4.891 - 5.101</td>
</tr>
<tr>
<td>29.360 - 31.880</td>
<td>518015</td>
<td>5.101 - 5.311</td>
</tr>
<tr>
<td>31.880 - 34.400</td>
<td>733831</td>
<td>5.311 - 5.522</td>
</tr>
<tr>
<td>34.400 - 36.920</td>
<td>781972</td>
<td>5.522 - 5.732</td>
</tr>
<tr>
<td>36.920 - 39.440</td>
<td>641864</td>
<td>5.732 - 5.842</td>
</tr>
<tr>
<td>39.440 - 41.960</td>
<td>424508</td>
<td>5.842 - 6.152</td>
</tr>
<tr>
<td>41.960 - 44.480</td>
<td>141267</td>
<td>6.152 - 6.363</td>
</tr>
<tr>
<td>44.480 - 47.000</td>
<td>15234</td>
<td>6.363 - 6.737</td>
</tr>
</tbody>
</table>

**Mean = 34.850**  
**Variance = 17.767**  
**Std.Dev. = 4.215**

<table>
<thead>
<tr>
<th>Throughput</th>
<th>No. of Tardy jobs</th>
<th>Summation of *</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Intervals</strong></td>
<td><strong>Numbers</strong></td>
<td><strong>Intervals</strong></td>
</tr>
<tr>
<td>0.110 - 0.113</td>
<td>101890</td>
<td>5.000 - 5.500</td>
</tr>
<tr>
<td>0.113 - 0.115</td>
<td>735256</td>
<td>5.500 - 6.000</td>
</tr>
<tr>
<td>0.115 - 0.118</td>
<td>1042776</td>
<td>6.000 - 6.500</td>
</tr>
<tr>
<td>0.118 - 0.121</td>
<td>948752</td>
<td>6.500 - 7.000</td>
</tr>
<tr>
<td>0.121 - 0.124</td>
<td>498024</td>
<td>7.000 - 7.500</td>
</tr>
<tr>
<td>0.124 - 0.126</td>
<td>148304</td>
<td>7.500 - 8.000</td>
</tr>
<tr>
<td>0.128 - 0.129</td>
<td>118984</td>
<td>8.000 - 8.500</td>
</tr>
<tr>
<td>0.129 - 0.132</td>
<td>31152</td>
<td>8.500 - 9.000</td>
</tr>
<tr>
<td>0.132 - 0.134</td>
<td>2512</td>
<td>9.000 - 9.500</td>
</tr>
<tr>
<td>0.134 - 0.137</td>
<td>1360</td>
<td>9.500 - 10.000</td>
</tr>
</tbody>
</table>

**Mean = 0.119**  
**Variance = 0.000**  
**Std.Dev. = 0.004**

**Mean = 8.541**  
**Variance = 88.801**  
**Std.Dev. = 8.429**