A HIERARCHICAL HEURISTIC APPROACH FOR MACHINE LOADING PROBLEMS IN A PARTIALLY GROUPED ENVIRONMENT

A Dissertation

by

JONG HWAN LEE

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2003

Major Subject: Industrial Engineering

A HIERARCHICAL HEURISTIC APPROACH FOR MACHINE LOADING PROBLEMS IN A PARTIALLY GROUPED ENVIRONMENT

A Dissertation

by

JONG HWAN LEE

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Approved as to style and content by:

César O. Malavé (Chair of Committee)	
Amarnath Banerjee (Member)	Sheng-Jen "Tony" Hsieh (Member)
Yu Ding (Member)	Brett A. Peters (Head of Department)

December 2003

Major Subject: Industrial Engineering

ABSTRACT

A Hierarchical Heuristic Approach for Machine Loading Problems

in a Partially Grouped Environment. (December 2003)

Jong Hwan Lee, B.S., Dongguk University;

M.S., Texas A&M University

Chair of Advisory Committee: Dr. César O. Malavé

The loading problem in a Flexible Manufacturing System (FMS) lies in the allocation of operations and associated cutting tools to machines for a given set of parts subject to capacity constraints. This dissertation proposes a hierarchical approach to the machine loading problem when the workload and tool magazine capacity of each machine are restrained. This hierarchical approach reduces the maximum workload of the machines by partially grouping them. This research deals with situations where different groups of machines performing the same operation require different processing times and this problem is formulated as an integer linear problem.

This work proposes a solution that is comprised of two phases. In the first phase (Phase I), demand is divided into batches and then operations are allocated to groups of machines by using a heuristic constrained by the workload and tool magazine capacity of each group. The processing time of the operation is different for each machine group, which is composed of the same identical machines; however, these machines can perform different sets of operations if tooled differently. Each machine and each group

of machines has a limited time for completing an operation. Operations are allocated to groups based on their respective workload limits.

In the second phase (Phase II), demand is divided into batches again and operations are assigned to machines based on their workload and tool magazine capacity defined by Longest Processing Time (LPT) and Multifit algorithms. In Phase II, like Phase I, partial grouping is more effective in balancing the workload than total grouping. In partial grouping, each machine is tooled differently, but they can assist one another in processing each individual operation.

Phase I demonstrates the efficiency of allocating operations to each group. Phase II demonstrates the efficiency of allocating operations to each machine within each group. This two-phase solution enhances routing flexibility with the same or a smaller number of machines through partial grouping rather than through total grouping. This partial grouping provides a balanced solution for problems involving a large number of machines. Performance of the suggested loading heuristics is tested by means of randomly generated tests.

ACKNOWLEDGEMENTS

This dissertation marks the beginning of the end of my Ph.D. studies in Texas. I would be remiss if I didn't acknowledge my debt of gratitude to God (who makes all things possible) as well as to several others, who have made my studies possible.

I'm especially grateful to those who have been my encouragement, my inspiration, my moral support and my spiritual foundation. Amongst my family I would especially like to thank my parents, Kyu Sang Lee and Eun Hee Kwak; my brother, Young Hwan Lee; my sisters, Ji Won Lee and Jungwon Rhee Hermanoff; my brother-in-law, Jeffrey Hermanoff; my wife Bo Sung Park; my lovely daughter Shirlyn Priscilla Rhee; and my precious second daughter, who will be born within a few months. I'm also grateful to my church family and the many friends I've met here.

I would like to express my sincere gratitude to the mentors who have guided me throughout this process and who have greatly influenced me and my work; those mentors include: Dr. César O. Malavé, Dr. Amarnath Banerjee, Dr. Sheng-Jen "Tony" Hsieh, and Dr. Yu Ding. I would also like to express my appreciation to Dr. M. Rita Caso who has supported me and who has shown great concern for my family and my studies.

I'd finally like to give special thanks to Pastor Jong Kim and his family -- I will never forget their love and prayers.

TABLE OF CONTENTS

Page

CHAPTER Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

TABLE Page

CHAPTER I

INTRODUCTION

1.1. MOTIVATION

Several production planning problems are inherent in Flexible Manufacturing Systems (FMS). Briefly, these problems are: (1) selecting compatible part types for simultaneous machining for the upcoming time period; (2) partitioning machines into machine groups, each of which can perform the same operations; (3) determining production ratios for each part type; (4) determining minimum inventory requirements (pallets and fixtures) for maintaining production ratios; (5) allocating operations and cutting tools to limited capacity tool magazines (Stecke 1983).

This research focuses on machine grouping and loading problems for a given set of part types. After product items and their quantities to be manufactured are determined by production planning, the next problem to be solved in production management is that of allocating the workloads to the existing production facilities for manufacturing these products. In general, the capacities (including human power) of the facilities are not infinite. Therefore, in order to actually perform production activities according to the production plan established, it is essential to adjust the workload for each of the facilities and workers in every time period so they do not exceed the given capacity. This decision is called “machine loading”.

This dissertation follows the style and the format of the *International Journal of Production Research.*

The loading problem depends not only on the part type selection problem but also on the grouping problem in the sense that a solution of the grouping problem or the decisions made regarding machine grouping generate the environment for the loading problem. Such machine grouping decisions are related to the number of machine groups, the number of machines in each group and the way in which the machines are grouped.

Grouping consists of tooling physically identical machines in order to allow them to process the same or different sets of operations. There are essentially three means for grouping: “no grouping”, “partial grouping” and “total grouping.” In “no grouping” configurations, each machine is tooled differently and each operation is assigned to only one machine. In “total grouping” configurations, machines are partitioned into groups in which all machines are identically tooled so that they can process the same set of operations (either individually or collaboratively). In “partial grouping” configurations, individual machines are tooled differently, as in the case of “no grouping”; however, multiple machines can be assigned to each operation (i.e. each operation is allocated to one or more machines, Lee and Kim 2000).

When an operation is assigned to multiple machines, a set of tools required for that operation must be loaded onto each machine. This is one of the distinct characteristics of partial grouping. In other sorts of grouping, tools are loaded before operations are assigned, however, in partial grouping, necessary tools are loaded after operations allocated to each machine. It is this characteristic of partial grouping that makes each machine a “virtual cell”.

In most cases in the field of production planning and scheduling, the processing time required to complete a specified operation is set as a constant. In most of the existing research about loading problems with respect to partial grouping, the machine loading models were constructed under the assumption that the processing time is a constant. In practical situations, however, it is possible to vary the processing times by actively changing manufacturing conditions, especially machining speeds. In these cases, some modifications must be made to production planning and scheduling models. To be useful, those models require a new type of heuristic that allows for variation in processing times.

This research is concerned with machine loading models that have variable processing times. Note that by solving the loading problem in a partial grouping environment, partial grouping may prove more flexible than total grouping with the same or a smaller number of machines. Furthermore, through the employment of clustering, we can deal with a large number of machine problems in a more balanced fashion. The performance of these suggested loading heuristics will be tested by means of randomly generated tests.

1.2. PROBLEM DESCRIPTION

The machine loading problem for FMS lies in the allocation of operations and associated cutting tools to machines for a given set of parts subject to capacity constraints. This dissertation proposes a hierarchical approach to the machine loading problem when the workload and tool magazine capacity of each machine are restrained. This hierarchical approach minimizes the maximum workload of the machines by

partially grouping them. This research deals with situations where different groups of machines performing the same operation require different processing times.

The FMS considered in this dissertation consist of several identical machines, each with an automatic tool changer and a tool magazine of a limited capacity. These machines can perform different sets of operations if tooled differently. To perform an operation, one or more tools are required, and each tool requires one or more slots in the tool magazine. In addition, several operations may utilize the same tools (a tool may be used for two or more operations) in the system. This is referred to as "tool sharing" or "tool commonality."

The system has to be reconfigured when a new set of part types is selected, that is, virtual cells may differ for different sets of selected part types. The number of duplicate tools required for an operation is equal to the number of machines to which the operation is assigned. More copies of certain tools may be required because of finite tool lives. In general, it is apparent that total grouping and no grouping are special forms of partial grouping.

This research assumes that a set of part types has been selected for simultaneous production during the upcoming production period, and production quantities for the parts have also been determined. This research also assumes that different machines involved in production require different processing times for the same operation. And we approach this research under the assumption that the machine grouping within each cluster is already finished. Given these assumptions, this research will attempt to assign to each machine the operations required for the production of the selected part types.

The fact that different machines require different processing times for the same operation makes this all the more challenging.

1.3. OBJECTIVES AND CONTRIBUTIONS

The objectives of this dissertation are to investigate the formulation of the loading problem as an integer programming problem, to develop a solution algorithm based on the formulation of the problem and to test solution methodologies. Eventually, these procedures will minimize the maximum workload for each machine.

The most significant contributions of the research described in this dissertation are (1) the development of a good heuristic in a loading problem with variable processing time for each cluster; (2) the decomposition of this problem into two sub-problems: Phase I assigns the operations into clusters. Phase II allocates operations to machines within each cluster; (3) the implementation of different heuristics, according to phase. In Phase I, operations are assigned to clusters while each cluster has a different processing time. In Phase II, operations are allocated to machines within each cluster while every machine within the clusters has the same processing time; (4) the attempt to address the loading problem with respect to each machine's capacity and workload limit, which impact loading problem performance.

1.4. ORGANIZATION OF DISSERTATION

The remainder of this dissertation is organized as follows: CHAPTER II provides an overview of related past work on grouping and loading problems and of the mathematical tools used in this dissertation. CHAPTER III presents the mathematical

formulation of the problem and the overall approach. In CHAPTER IV, heuristic methods are developed, presented and analyzed for computational efficiency. For each sub-problem, a heuristic is either extended or developed based on previous documented viable approaches. CHAPTER V presents results based on randomly generated data (data generated by uniform distribution in order to show the efficiency of the heuristics). CHAPTER VI provides the conclusion of the work and some proposal for future research.

CHAPTER II

LITERATURE REVIEW

There is extensive literature on assembly line and workload balancing in job shops and FMS. Balancing is appropriate for flexible assembly systems as well as automated transfer lines. Stecke and Solberg (1981) employed loading and control policies for a flexible manufacturing system and defined loading and control methods that significantly improve system production rates. Stecke (1986) considered various operation assignment objectives appropriate in FMS and presented a hierarchical framework for considering these objectives. Berrada and Stecke (1986) applied a branch and bound algorithm to solve the workload balancing problem for all machines when each machine's processing time is different. Kim and Yanco (1994) developed a new branch and bound algorithm, based on the work of Berrada and Stecke (1986). This new algorithm was developed to maximize the expected production rate (throughput) of the system and to ensure that actual workload allocation is commensurate with the continuous workload allocation that maximizes throughput.

Stecke and Morin (1985) have shown that if each operation is assigned to only one machine, balancing the workload of each machine maximizes expected production by using symmetric mathematical programming. Stecke and Solberg (1985) showed that if functionally similar machines are pooled into machine groups of equal size, then balancing workloads again maximizes expected production by deploying a closed queueing network.

Shanthikumar and Stecke (1986) showed that maintaining balanced workloads on each machine over time stochastically minimizes work-in-process inventory requirements for FMS that contain only one machine in a group. Kumar et al. (1990) employed the combined grouping and loading problem as a multistage multiobjective optimization model and also employed the "min-max" approach to multiobjective optimization in order to obtain a compromise solution.

Chen and Askin (1990) performed heuristics, based on the separate evaluation of five objectives: workload balance, volume of inter-machine part movement, routing flexibility, tool investment and maximum machine utilization. With respect to balancing machine utilization, the Machine Balance Assignment (MBA) heuristic dominated other heuristics. (Due to the apparent utility of the MBA heuristic in balancing workloads, a modified version of it is used in Phase I of this dissertation.) Chen and Askin concluded that the assignment of operations to machine types according to workload balance parameters is better than the assignment of operations to the most efficient machine type.

Shanker, K. and Srinivasulu (1989) defined the loading problem as the selection of a subset of jobs from a job pool and the allocation of those jobs among machines. They then developed a two-stage branch and backtrack procedure in order to maximize the assigned workload. Later, Mukhopadhyay et al. (1992) suggested a more advanced and reliable heuristic procedure for loading problems that was meant to minimize system imbalances and thereby maximize the throughput originally provided by the procedures proposed by Shanker, K. and Srinivasulu (1989). Choi and Lee (1998) developed heuristic procedures with the two-part objective of minimizing workload imbalances and

maximizing system throughput. Their solution hinged on the rejection factor and virtual total processing time. They demonstrated that they had obtained a better solution than Shanker, K. and Srinivasulu (1989) as well as Mukhopadhyay et al. (1992).

Coffman et al. (1978) applied bin-packing to the multiprocessor scheduling problem. They described the well-known LPT (Longest Processing Time) algorithm and MUTIFIT algorithm in terms of bin-packing techniques. Kim (1993) reported that the minimization of the maximum workload is closely related with the minimization of makespan for a given set of part types. Kim and Yanco (1994) considered the loading problem in order to maximize the throughput for a specified steady-mix of orders. Since it is computationally difficult to find optimal solutions for problems with more than 20 operations, faster heuristics algorithms were developed.

The proposed heuristics in Kim and Yanco (1993) differ from earlier work in two ways. First, multi-pass rather than single-pass bin-packing approaches are used. Second, new assignment rules explicitly consider tool magazine capacity constraints. Two-dimensional bin-packing algorithms were also adapted to approach this problem. Rectangle widths denote the number of tool slots needed, and rectangle heights denote workloads. When balancing the workloads is the objective, LPT and Multifit algorithms are slightly better than the other algorithms. Due to their utility, these algorithms are applied in Phase II of this dissertation.

There are various algorithms for the identical processor minimum-makespan scheduling problem, and for some of them the worst case performance ratios are known. The LPT algorithm has a worst case performance ratio of 4/3-1/3m, where m is the

number of processors (machines), and the Multifit algorithm has a worst case performance ratio of $1.2+(1/2)^k$, where k is the number of iterations in the algorithm. (Coffman et al. (1978), Friesen (1984) and Graham (1969) for details.)

Stecke and Raman (1994) mentioned the problem of determining optimal machine workload in order to minimize mean part flow time with the objective of (un) balancing workloads for the “no grouping”, “total grouping”, and “partial grouping” configurations. They decompose this problem into the subproblems of first forming machine groups and next assigning operations to these groups. They propose a heuristic approach which is a modification of the “first fit decreasing” heuristic for the bin-packing problem.

Lee and Kim (2000) implemented several loading algorithms for flexible-manufacturing systems with partially grouped machines. They formulated the loading problem by means of integer programming and primarily utilized LPT and Multifit algorithms. They described two means of addressing this problem. One approach was direct, whereas the other decomposed the problem into the operation assignment problem and the workload allocation problem. Both approaches implemented LPT and Multifit algorithms, yet a comparison of the results demonstrated that decomposition methods are better that direct ones. Additionally, simulation experiments demonstrated that partial grouping loading plans yielded significantly better performance than total grouping loading plans.

Sarin and Chen (1987) discussed the machine loading and tool allocation problems and developed mathematical models for achieving minimum overall

machining cost by determining the route of parts through machines as well as for allocating appropriate cutting tools to each machine. Bretthauer and Venkataramanan (1990) studied the assignment of operations to machines in flexible manufacturing systems and also studied the impact of alternate routes through these systems. By assigning each operation to more than one machine, alternatives for the parts being produced become available. They present a constrained network model of the machine loading problem and use surrogate and Lagrangian relaxation for solving large-scale problems.

Ram et al. (1990) modeled the problem of machine loading planning and tool allocation in a FMS as a discrete generalized network with simple side constraints, and described an algorithm to yield a solution to this problem. An important aspect of this modeling process is its ease of application to other planning problems in FMS. Moreno and Ding (1993) studied FMS loading and part type selection problems in which each part is processed by a series of operations. They presented two heuristic methods for balancing workloads and meeting due dates. Their first goal was to achieve workload balance and their second was to reduce the number of late part types. The loading and part type selection had to satisfy tooling constraints. Computational results were encouraging and indicated significant improvement over existing methods.

Tang et al. (1995) introduced a framework for a two-phase planning and scheduling model for selecting part types and for assigning required tools to machines for processing. Chen et al. (1995) presented an integer programming model for existing FMS users to select the most cost effective set of parts to run simultaneously on an FMS

during a specified production horizon. Two heuristic solution algorithms were developed by dividing the part-selection procedure into two stages.

Kuhn (1995) formulated the loading problem as a linear mixed 0-1 program in order to minimize the greatest workload assigned to each machine. This involved a heuristic procedure in which operations were assigned to machine tools according to the solution of a parameterized generalized assignment problem with an objective function that approximates the use of tool slots required by the operations assigned to the machines. Rupe and Kuo (1997) provided improved solutions to the generalized tool loading problem by means of a unique solution involving job splitting. By additionally allowing concurrent job and tool changing, a new optimal tooling policy was obtained that proved useful with either the previously developed job scheduling heuristics, or the algorithm.

Denizel and Sayin (1998) studied the part-type selection problem in a Flexible Manufacturing Environment. They formulated a problem in which the objectives were the maximization of the number of part-types selected and the minimization of the measure of total tardiness. Mukhopadhyay et al. (1998) considered the problem of FMS machine loading with the objective of minimizing system imbalances using a simulated annealing approach. New job sequences were generated with a proposed perturbation scheme named “the modified insertion scheme.” These sequences were used in the proposed simulated annealing algorithm in order to arrive at a near global optimum solution.

Nayak and Acharya (1998) proposed a three-stage approach to solving part type selection, machine loading and part type volume determination problems. In contrast to the usual approach of maximizing the part types in each batch, they attempted to maximize the routeing flexibility of the batches. Guerrero et al. (1999) presented a new approach to the loading problem in flexible manufacturing systems. It focused on the existence of alternative routes for each part type and directly determined the optimal number of copies of each tool type to be loaded into each tool magazine. The loading objective was that of balancing machine workloads by using decision variables of routing mix and tool allocation.

CHAPTER III

MODEL DEVELOPMENT

3.1. INTRODUCTION

This chapter describes the development of the linear integer programming models for the loading problem. More specific assumptions used in modeling are provided and discussed. Finally, linear integer programming models of the problems are presented.

3.2. ASSUMPTIONS

1. The number of operations, the number of clusters and the number of machines within each cluster are given.
2. Tool magazine capacity and workload limit for each machine are given.
3. Demand, processing time for each cluster and processing time for each operation are generated by uniform distribution.
4. The number of tools needed for each operation and the number of tool slots needed for each tool are generated by uniform distribution.
5. Clustering is finished before loading is started.
6. In each cluster, several identical machines, each with an automatic tool changer and a tool magazine, will be clustered; however, these machines can perform different sets of operations if tooled differently.
7. Processing time is different on each cluster, but processing time is the same within each cluster.

8. If tool magazine capacity or workload limit is overloaded, then the system must be restarted.
9. Setup time and slack time between loadings is ignored.
10. A set of part types has been selected to be produced simultaneously during the upcoming production period.
11. Operations that require different processing times are considered different operations even though they are of the same type.

3.3. MODEL DEVELOPMENT

The purpose of this problem is to assign each operation to machines. In order to assign operations to machines, those operations must be assigned to cluster first, and then they can be assigned to machines in each cluster. The machines used for these FMS have automatic tool changers and a tool magazine of a limited capacity and can work limited time for each day. To execute an operation, one or more tools are required, and each tool requires one or more slots in the tool magazine. Another significant feature of this system is tool sharing or tool commonality. In other words, several operations may share the same tools in the system. Operations require several tools: yet if different operations are allocated to the same cluster or different machines use the same tool, then duplicated tools are not assigned to the same cluster or machine.

If operations require different processing times, they are considered different operations even though they are the same type. For instance, drilling operations for different part types are treated as different operations if their processing times are different -- even though they require the same set of tools. The loading problem allocates

operations and associated tools to machines in order to minimize the maximum workload of the machines subject to tool magazine capacity and workload capacity constraints. An integer linear problem provides a clear description of this loading problem. The following notations are used in its formulation.

3.3.1. INTEGER LINEAR PROGRAMMING

This formulation explains about entire process for assigning operations to machines. A small integer example problem for 3 clusters, 3 machines, and 5 operations was solved to test the integer programming formulation. Later this problem decomposed into two sub problems. We have used the following notations throughout the dissertations. Let:

i index for operation, $i \in$ I;

j index for machine, $j \in$ J;

t index for tools, $t \in$ T;

k index for machine type, $k \in K$;

c index for machine cluster, c $\in C$;

p_i processing time of operation i;

p_{ki} processing time of operation i for machine type k;

C_{cj} capacity of a tool magazine of a machine j in cluster c;

C_c capacity of a tool magazine of a machine cluster c;

W_{cj} the workload for a machine j in the cluster c;

W_c the maximum workload allowed for a cluster c;

D_i demands of operation i;

D_{ci} demands of operation i in the cluster c;

S_t number of tool slots needed for tool t;

b_{cij} batches for operation i that is assigned to machine j in the cluster c.

b_{ic} batches for operation i that is assigned to machine cluster c.

$|J_c|$ number of machine in the cluster c;

$|G|$ number of cluster;

a_{cit} $\begin{cases}1 \text{ if operation i requires tool } t \text{ in the cluster c,}\\ 0 \text{ otherwise.}\end{cases}$

a_{it} $\begin{cases}1 \text{ if operation i requires tool } t,\\ 0 \text{ otherwise.}\end{cases}$

x_{cij} $\begin{cases}1 \text{ if operation i is assigned to machine } j \text{ in the cluster c,}\\ 0 \text{ otherwise.}\end{cases}$

x_{ic} $\begin{cases}1 \text{ if operation i is assigned to machine cluster c,}\\ 0 \text{ otherwise.}\end{cases}$

y_{ctj} $\begin{cases}1 \text{ if tool t is assigned to machine } j \text{ in the cluster c,}\\ 0 \text{ otherwise.}\end{cases}$

y_{tc} $\begin{cases}1 \text{ if tool t is assigned to machine cluster c,}\\ 0 \text{ otherwise.}\end{cases}$

Problem (P)

Minimize Z,

subject to

$$\sum_{k\in K}\sum_{i\in I} p_{ki}b_{ic} \le W_c \qquad \forall \text{ c,} \qquad (1)$$

$$W_c = \sum_{j\in J} W_{cj} \qquad \forall \text{ c,} \qquad (2)$$

$$\sum_{i\in I} p_i b_{cij} \le Z \qquad \forall j, c, \qquad (3)$$

$$\sum_{i\in I} p_i b_{cij} \le W_{cj} \qquad \forall j, c, \qquad (4)$$

$$1 \le \sum_{c \in G} x_{ic} \le |G| \qquad \forall\, i, \qquad (5)$$

$$1 \le \sum_{j \in J} x_{cij} \le |J_c| \qquad \forall\, i,\ c, \qquad (6)$$

$$b_{ic} \le D_i x_{ic} \qquad \forall\, i,\ c, \qquad (7)$$

$$b_{cij} \le D_{ci} x_{cij} \qquad \forall\, i,\ j,\ c, \qquad (8)$$

$$\sum_{c \in C} b_{ic} = D_i \qquad \forall\, i, \qquad (9)$$

$$\sum_{j \in J} b_{cij} = D_{ci} \qquad \forall\, i,\ c, \qquad (10)$$

$$a_{it} x_{ic} \le y_{tc} \qquad \forall\, i,\ c,\ t, \qquad (11)$$

$$a_{cit} x_{cij} \le y_{ctj} \qquad \forall\, i,\ j,\ t,\ c, \qquad (12)$$

$$\sum_{t \in T} s_t y_{tc} \le C_c \qquad \forall\, c, \qquad (13)$$

$$\sum_{t \in T} s_t y_{ctj} \le C_{cj} \qquad \forall\, j,\ c, \qquad (14)$$

$$p_{ki}, b_{cij}, b_{ic} \ge 0 \text{ and integer}, \qquad \forall\, i,\ j,\ c, \qquad (15)$$

$$x_{cij}, y_{ctj}, x_{ic}, y_{tc} \in \{0,1\} \qquad \forall\, i,\ j,\ c,\ t. \qquad (16)$$

In the formulation, constraint set (1) states that operations can be assigned to any machine cluster and workload for each cluster is restricted. The sum of all workloads of the machines in each cluster is defined in constraint set (2). Constraint set (3) and (4) also imply that any operation can be allocated to any machine and in constraint (3) leads to workload balancing, i.e., minimization of the maximum workload, which is represented by Z_c. Constraint set (3) assures that the workload for each machine in each

cluster is bounded. Constraint set (5) denotes that the number of clusters allocated to each operation should not be greater than the total number of clusters, and constraint set (6) represents that the number of machines in the cluster allocated to each operation should not be greater that the total number of machine in the cluster. Constraint set (5) indicates that the batches of operation i can be allocated to cluster c (b_{ic}) only if $x_{ic}=1$, that is, the relationship between b_{ic} and x_{ic}. Constraint set (6) denotes that the batches of operation i can be assigned to machine j (b_{cij}) only if $x_{cij}=1$, that is, the relationship between b_{cij} and x_{cij}. Constraint set (9) ensures that the sum of all batches to be assigned to each cluster in each operation is the demand of each operation. Constraint set (10) illustrates that the sum of all batches in each cluster to be assigned to each machine in the cluster in each operation is the demand of each operation. Constraint set (11) illustrates that the required tools for the operation to be loaded onto each cluster where the operation is allocated. Constraint set (12) is in reference to the tools required for each operation in each cluster that are to be loaded onto each machine where the operations are assigned in each cluster. Constraint sets (13) and (14) bound the number of parts that can be processed on a cluster and a machine within a cluster.

3.3.2. SOLUTION APPROACH

Figure 1 shows the overall conceptual approach of the proposed methodology. The proposed approach can be divided into two phases: In Phase I, operations will be allocated into clusters and in Phase II operations will be assigned into machines within each cluster. The solution of Phase I can effect the solution of Phase II. If workload is

well balanced between clusters, then there is a high possibility that workload will be minimized when we assign operations to machines within clusters.

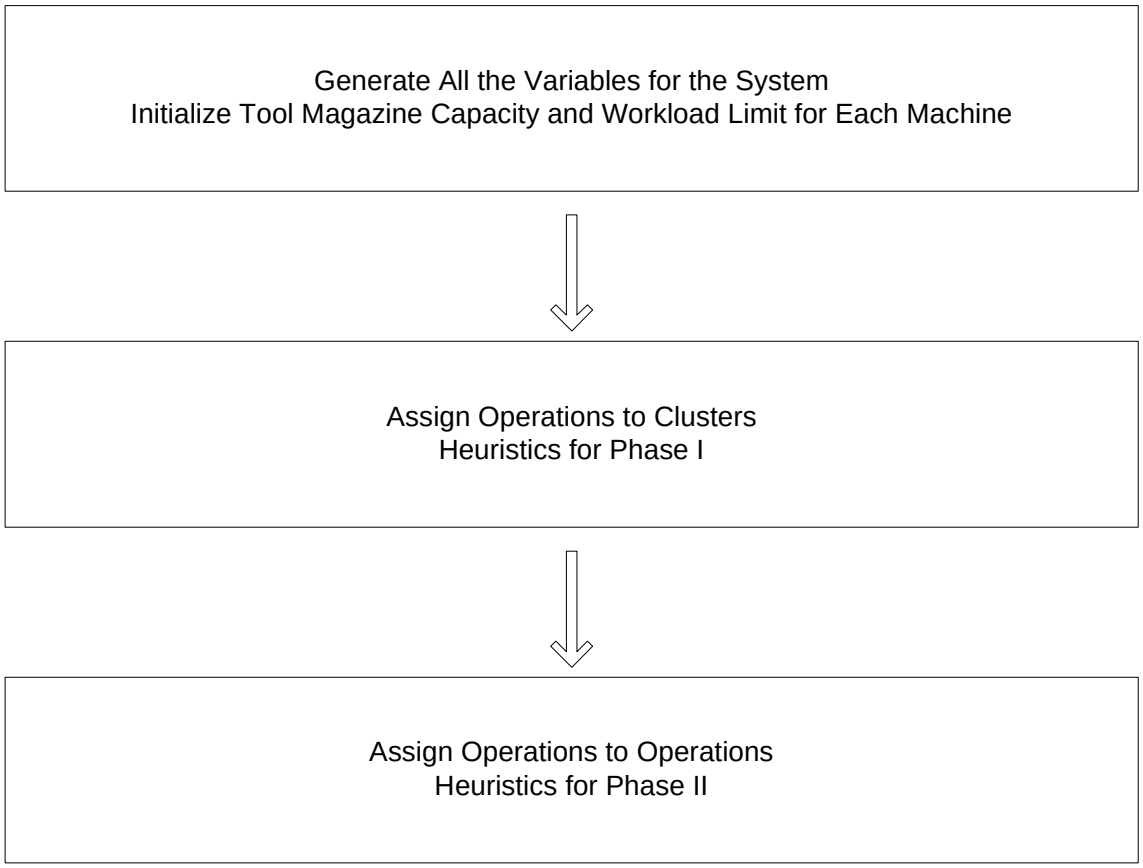

Figure 1 Overall conceptual approach.

3.3.3. INTEGER LINEAR PROGRAMMING FOR PHASE I

The formulation above is decomposed into two formulations for Phase I and Phase II. Phase I deals with the problem of assigning operations to clusters. The formulation for Phase I appear below. Constraint (1) is changed to express the selection of Z, the maximum value between clusters. However, the objective is to minimize the maximum value of Z.

Sub Problem I (S)

Minimize Z,

subject to

$$\sum_{k\in K}\sum_{i\in I} p_{ki} b_{ic} \le Z \qquad \forall \text{ c}, \qquad (17)$$

$$W_c = \sum_{j\in J} W_{cj} \qquad \forall \text{ c}, \qquad (18)$$

$$1 \le \sum_{c\in G} x_{ic} \le |G| \qquad \forall\ i, \qquad (19)$$

$$b_{ic} \le D_i x_{ic} \qquad \forall\ i,\ c, \qquad (20)$$

$$\sum_{c\in C} b_{ic} = D_i \qquad \forall\ i, \qquad (21)$$

$$a_{it} x_{ic} \le y_{tc} \qquad \forall\ i,\ c,\ t, \qquad (22)$$

$$\sum_{t\in T} \text{s}_t y_{tc} \le C_c \qquad \forall\ c, \qquad (23)$$

$$p_{ki}, b_{ic} \ge 0 \text{ and integer}, \qquad \forall\ i,\ j,\ c, \qquad (24)$$

$$x_{ic}, y_{tc} \in \{0,1\} \qquad \forall\ i,\ j,\ c,\ t. \qquad (25)$$

3.3.4. INTEGER LINEAR PROGRAMMING FOR PHASE II

In Phase II, operations are allocated to machines in the cluster (formulation appears below). The batches (b_{ic} in Phase I) assigned to each cluster will be D_{ci} in each cluster. In the constraint (1), Z is the value of workload that will cover all of the machines workload and the objective is minimizing that value of Z.

Sub Problem II (S)

Minimize Z,

subject to

$$\sum_{i \in I} p_i b_{cij} \le Z \qquad \forall\, j, c, \qquad (26)$$

$$\sum_{i \in I} p_i b_{cij} \le W_{cj} \qquad \forall\, j, c, \qquad (27)$$

$$1 \le \sum_{j \in J} x_{cij} \le |J_c| \qquad \forall\, i, c, \qquad (28)$$

$$b_{cij} \le D_{ci} x_{cij} \qquad \forall\, i, j, c, \qquad (29)$$

$$\sum_{j \in J} b_{cij} = D_{ci} \qquad \forall\, i, c, \qquad (30)$$

$$a_{cit} x_{cij} \le y_{ctj} \qquad \forall\, i, j, t, c, \qquad (31)$$

$$\sum_{t \in T} s_t y_{ctj} \le C_{cj} \qquad \forall\, j, c, \qquad (32)$$

$$b_{cij} \ge 0 \text{ and integer}, \qquad \forall\, i, j, c, \qquad (33)$$

$$x_{cij}, y_{ctj} \in \{0,1\} \qquad \forall\, i, j, c, t. \qquad (34)$$

3.4. SUMMARY

This chapter presents the model development for the loading problem. Modeling variables, parameters and coefficients were defined. Then, relationships among variables

were presented. Next, an appropriate objective function was defined, one that was supportive of the overall objective. Last, the loading problem was concisely represented as a two-phase model. The next chapter examines detailed solution strategies for the models presented in this chapter.

CHAPTER IV

SOLUTION METHODOLOGY

4.1. INTRODUCTION

This chapter develops solution procedures for the loading problem workload balancing models of both Phases. The overall conceptual approach is discussed and the heuristics proposed for Phase I are introduced. Next, an alternative heuristic for Phase I is presented. Then an LPT (Longest Processing Time) algorithm and a Multifit algorithm for Phase II are presented. Finally, the proposed heuristics are analyzed for computational efficiency.

4.2. OVERALL CONCEPTUAL APPROACH

This work proposes a solution that is comprised of two phases. In the first phase (Phase I), demand is divided into batches and then operations are allocated to groups of machines by using a heuristic constrained by the workload and tool magazine capacity of each group. The processing time of the operation is different for each machine group composed of the same identical machines; however, these machines can perform different sets of operations if tooled differently. After obtaining batches for each cluster, batch workload will be obtained by multiplying the batch and processing times for each operation. Each machine and each group of machines has a limited time for completing an operation. Operations are allocated to groups based on their respective workload limits. Four heuristics and a Multifit algorithm for each heuristic will be implemented in

Phase I. These heuristics will be compared with each other using the relative performance ratio (the ratio of a solution to a lower bound) as an index.

In the second phase (Phase II), demand, which was allocated in Phase I, is divided into batches again and operations are assigned to machines based on their workload and tool magazine capacity as defined by LPT algorithm and Multifit algorithms. In Phase II, just as in Phase I, partial grouping is applied again, because it proved more effective in workload balancing than total grouping. In partial grouping, each machine is tooled differently, but one or more machines can process each operation.

4.3. DESCRIPTION OF PROPOSED APPROACH

A detailed description of the heuristics for Phase I and Phase II is provided along with some examples that help explain the procedure in practical terms.

4.3.1. HEURISTIC APPROACH FOR PHASE I

In Phase I, operations are assigned to machine clusters. The demand of an operation is divided into the same number of batches for all operations. Batches for each operation are divided by the number of clusters (to the nearest integer). This Phase is required only when operations can be performed on more than one machine type and processing time is different for each cluster. We utilize this latitude to equalize workload. Operations are ordered according to the number of different clusters to which they may be assigned. From the operations with the fewest choices, we select the longest operation (total batch processing time) and assign it to the machine cluster that will end up with nearly equal workloads on each machine cluster.

There are four Heuristics and four Multifit algorithms for each Heuristic in Phase I. The first Heuristic selects the maximum batch workload for each cluster, and then selects the minimum batch workload among them. The second Heuristic selects the minimum batch workload value for each cluster, and then selects the minimum batch workload value among them instead of selecting the maximum batch workload value of the minimum from the cluster types in Step 3. The third Heuristic selects the maximum batch workload value for each cluster and then selects the maximum batch workload value from among them. The fourth Heuristic selects the minimum batch workload value in each cluster and then chooses the maximum batch value among them. By means of these trials, we can determine which of the four Heuristics provides the best performance. The modified Multifit algorithm will be applied with each Heuristic.

W_c and C_c are control parameters for the heuristic. When assigning operations, we use W_c and C_c to limit the operation aggregation process. As W_c increases, fewer but longer operations will be fed to the solution procedure. Material handling should decline but workload balancing may become more difficult in the cluster.

We will use the term $\Delta\tau_{ic}$ as a dynamic variable to indicate the number of machine tool slots that must be added to cluster type c to perform operation i. This term is dynamic in the sense that it depends on which tools have already been assigned to c. For example, if two operations use the same tool and the first operation is assigned to clusters, the second operation must then use additional tool slots. In other words, if the concept of tool sharing is used in Phase I, then in Phase II, when we allocate operations within clusters, we will confront the shortage of tool magazines in each machine.

And, the term $\Delta\eta_{ic}$ will be used as a dynamic variable for denoting the workload limit for each cluster. There is a workload limit for each machine and the total workload limit for the machines within each cluster is the workload limit for that cluster. This workload limit value is decreased as it is when assigning operations to clusters. If the workload limit for each cluster has the negative value, then the system is infeasible.

The tool magazine capacity for each cluster defines the system in the same way. If the system is infeasible, then the system stopped and other variables are generated for it. If these problems happen repeatedly, then tool magazine capacities and workload limits for each machine must be considered. Flexible workload limits maximize the probability that all of the operations can be assigned to clusters; however, they also lower utilization. Figure 2 provides a flow chart for the Phase I Heuristic.

4.3.1.1. PHASE I HEURISTIC

The main factor that we need to consider is the workload, because the objective is minimize the maximum workload so that we can minimize the lead time with these results. Figure 2 is the flow chart for Phase I Heuristic.

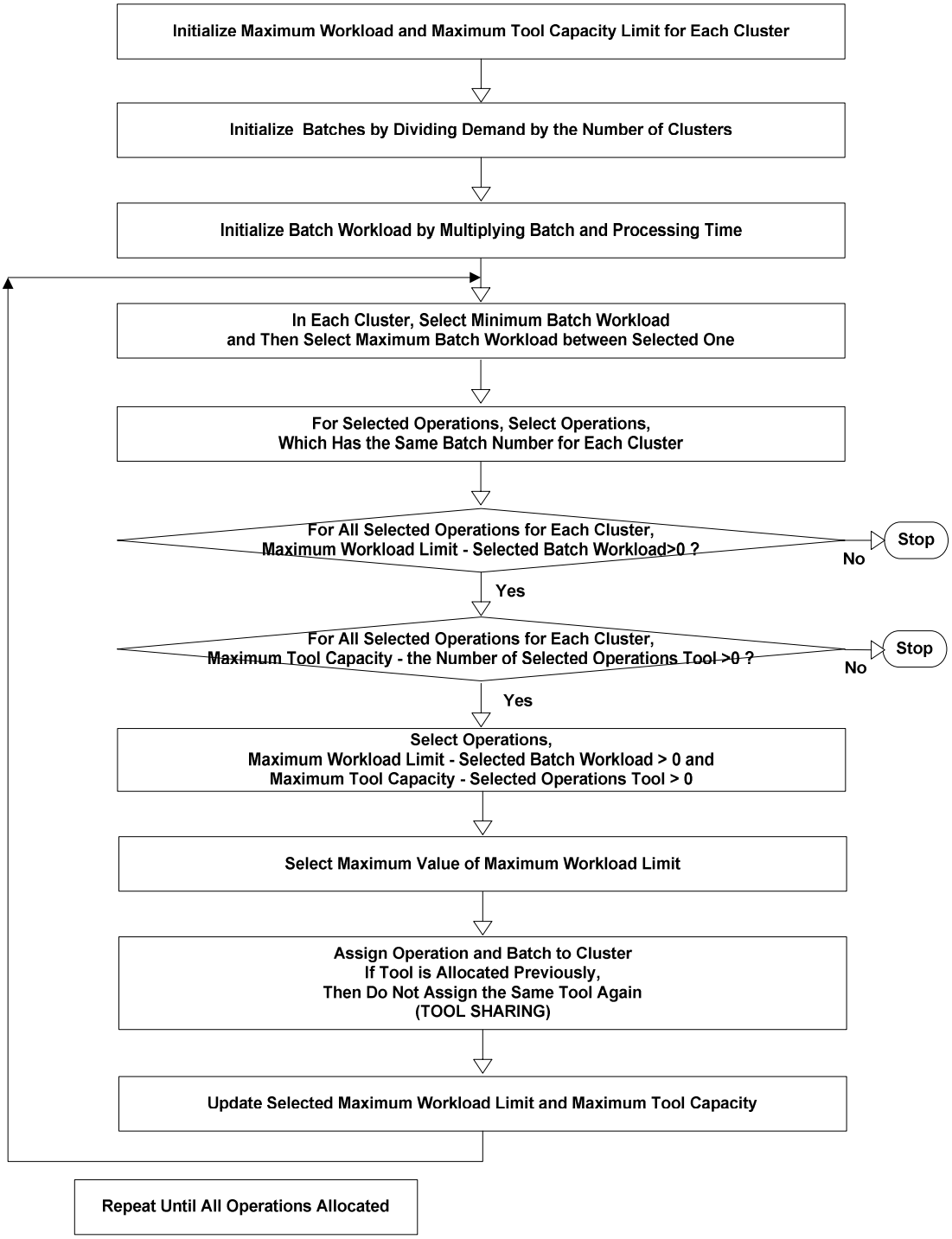

Figure 2 Flow chart of Heuristic I in Phase I.

General procedure for Heuristic I in Phase I

Step 1. Initialize maximum workload and maximum tool capacity.

Step 2. Make batches for each operation by dividing demands of an operation by the number of clusters (Batches must be integers, so if demand=10 and the number of machines is 3, then the batches are 3,3,4).

Step 3. Initialize batch workload by multiplying batch and processing time for each operation.

Step 4. In each cluster, select the minimum batch workload, and then select the maximum batch workload from those previously selected.

Step 5. For the operations that have been selected, select operations that have the same batch number for each cluster.

Step 6. For all selected operations for each cluster, select if maximum workload limit – selected batch workload ($\Delta\eta_{ic}$)> 0 and maximum tool capacity – the number of tool ($\Delta\tau_{ic}$) > 0.

Step 7. Select maximum value of maximum workload limit – selected batch workload ($\Delta\eta_{ic}$).

Step 8. Assign operation to the cluster that has the largest remaining workload capacity -- if tools were assigned previously, then do not allocate again. Update maximum workload limit and maximum tool capacity.

Step 9. Repeat until all operations are allocated.

Example of Heuristic I in Phase I

The sample data below in the Table 1 and Table 2 will be used to illustrate the Heuristic procedures.

Operation number	Demand	Processing time for cluster type			Batches			Workload in cluster A			Workload in cluster B			Workload in cluster C		
		A	B	C				1	2	3	1	2	3	1	2	3
1	9	3	2	4	3	3	3	9	9	9	6	6	6	12	12	12
2	10	2	6	3	3	3	4	6	6	8	18	18	24	9	9	12
3	7	4	2	8	2	2	3	8	8	12	4	4	6	16	16	24
4	6	7	3	4	2	2	2	14	14	14	6	6	6	8	8	8
5	8	6	9	2	2	3	3	12	18	18	18	27	27	4	6	6

Table 1 Loading data for small example to illustrate Heuristic.

Operation number	The number of tools needed for operation	Tool number for each tool	Tool Number	The number of tool slot needed for each tool
1	3	4, 7, 9	1	1
			2	2
2	2	2, 5	3	1
			4	1
3	5	1, 2, 5, 7, 8	5	1
			6	2
4	7	2, 3, 4, 6, 8, 9, 10	7	3
			8	1
5	4	3, 6, 7, 9	9	1
			10	1

Table 2 Other parameters for small example to illustrate Heuristic.

Example procedure of Heuristic I in Phase I

Step 0. Initialize the workload Limit as 500 and the tool magazine capacity as 100.

Step 1. In each cluster, find the minimum workload.

Step 2. Choose the maximum workload found in Step 1. 6 for cluster A, 4 for cluster B and 4 for cluster C.

Step 3. Relate the batch value (found in Step 2, 3 in batches column). This batch can be allocated into cluster A, B, or C.

Step 4. Workload for this batch in each cluster is 6, 18, and 9. Verify that the workload limit and tool capacity limit for each cluster is greater than 0 after assigning this batch to each cluster.

Step 5. Determine whether workload limit and tool capacity limit are greater than 0.

For cluster A, 500-6=494, for cluster B, 500-18=482 and for cluster C, 500-9=491. Tool capacity for each cluster is 100-3=97, since chosen batch is in operation 2. Operation 2 needs 2 tools and these tools need 3 slots.

If a tool is already assigned to a cluster, do not allocate the same tool again. This is the concept of tool sharing. If workload limit and tool capacity limit are less than 0, then this system is infeasible.

Step 6. Between the values chosen in Step 5, choose the one that has the maximum workload remaining value. Cluster A has a workload limit of 494.

Step 7. Assign operations to the cluster that was selected (when an operation is allocated to a cluster, the tools associated with that operation are also allocated). By this

method, one can determine which tools are allocated to which clusters we well as determine how many tool slots are required (Table 2).

Step 8. Update workload limit and tool capacity limit for chosen machine.

Step 9. Repeat until all operations are allocated.

4.3.1.2. ALTERNATIVE HEURISTICS OF PHASE I

There is a flow chart for alternative Heuristics for Phase I in Figure 3. The first alternative selects the minimum workload value for each cluster, and then selects the minimum workload value among them instead of selecting the maximum workload value of the minimum from among cluster types as in Step 3. The second alternative selects the maximum workload value for each cluster and then selects the maximum value from among them. The third alternative selects the minimum workload value in each cluster and then chooses the maximum value from among them. The last alternative is a modified Multifit algorithm, which will be explained in more detail in Phase II. After implementing all of the Heuristics, a relative performance ratio, which is defined by $[(H_h - H_b)] / H_h] \times 100$, will be shown in order to evaluate the results.

When allocating operations to clusters, there are two choices. The first involves the allocation of operations to clusters, which provides the largest remaining workload capacity. The second involves the allocation of operations to the smallest remaining workload capacity. Therefore, for each Heuristic, two alternatives are pursued. The flow chart that appears below is an example of the first alternative. It allocates operations to the cluster with the largest remaining workload capacity.

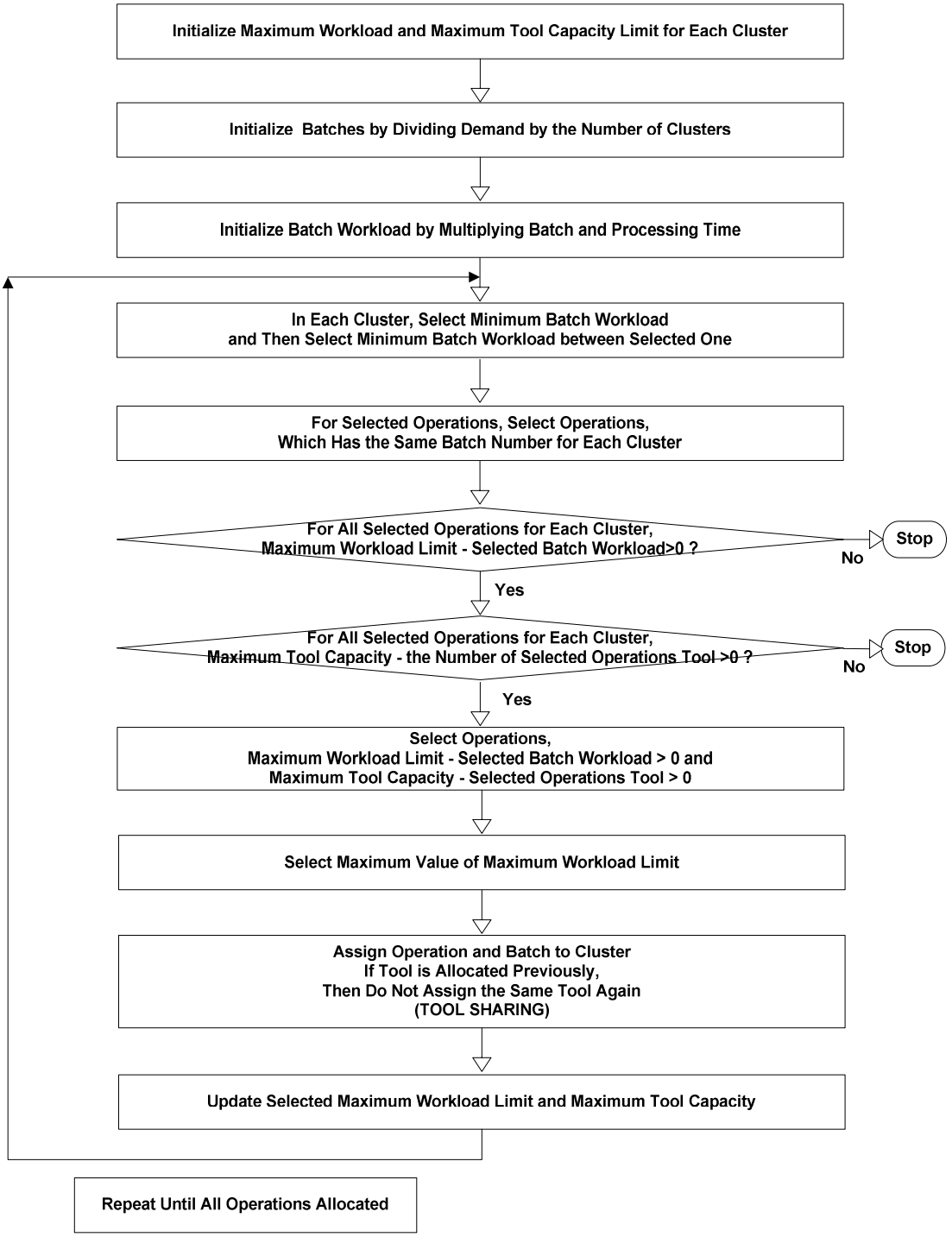

Figure 3 Flow chart of alternative Heuristic II in Phase I.

General procedure of alternative Heuristic II in Phase I

Step 1. Initialize maximum workload and maximum tool capacity.

Step 2. Make batches for each operation by dividing operation demands by the number of clusters (Batches must be integers, so if demand=10 and the number of machines is 3, then the batches are 3, 3, 4).

Step 3. Initialize batch workload by multiplying batch and processing time for each operation.

Step 4. In each operation, select the minimum batch workload, and then select the minimum batch workload from among those previously selected.

Step 5. For the operations that have been selected, select operations that have the same batch number for each cluster.

Step 6. For all operations previously selected for each cluster, select if the maximum workload limit – selected batch workload ($\Delta\eta_{ic}$)> 0 and maximum tool capacity – the number of tools ($\Delta\tau_{ic}$)> 0.

Step 7. Select maximum value of maximum workload Limit – selected batch workload ($\Delta\eta_{ic}$).

Step 8. Assign operations to the cluster that has the smallest remaining workload capacity – if tools were assigned previously, then do not allocate again. Update maximum workload limit and maximum tool capacity.

Step 9. Repeat until all operations are allocated.

Example of alternative Heuristic II for Phase I

The sample data below in the Table 3 will be used to illustrate the Heuristic procedures.

Operation number	Demand	Processing time for cluster type			Batches			Workload in cluster A			Workload in cluster B			Workload in cluster C		
		A	B	C				1	2	3	1	2	3	1	2	3
1	9	3	2	4	3	3	3	9	9	9	6	6	6	1	1	1
2	10	2	6	3	3	3	4	6	6	8	1	1	2	9	9	1
3	7	4	2	8	2	2	3	8	8	1	4	4	6	1	1	2
4	6	7	3	4	2	2	2	1	1	1	6	6	6	8	8	8
5	8	6	9	2	2	3	3	1	1	1	1	2	2	4	6	6

Table 3 Example to illustrate alternative Heuristic II in Phase I.

Example procedure of alternative Heuristic II in Phase I

Step 0. Initialize the workload limit as 500, and the tool magazine capacity as 100.

Step 1. In each cluster, find the minimum workload.

Step 2. Between workloads found in Step 1, choose the minimum. 8 for cluster A, 4 for cluster B and 16 for cluster C.

Step 3. Relate the batch value (found in Step 2, 3 in batches column).This batch can be allocated to cluster A, B, or C.

Step 4. Workload for the batch in each cluster is 8, 4, and 16. After assigning this batch to each cluster, verify that the workload limit and tool capacity limit for each cluster is greater than 0.

Step 5. Determine if the workload limit and the tool capacity limit are greater than 0.

For cluster A, 500-8=492, for cluster B, 500-4=496 and for cluster C, 500-16=484. The Tool Capacity for each cluster is 100-8=92, since the chosen batch is in operation 2. Operation 2 needs 2 tools and these tools need 8 slots.

If a tool is already assigned to a cluster, then do not allocate that tool again. This is the concept of tool sharing.

If workload tool capacity limits are less than 0, then this system is infeasible.

Step 6. Select the maximum workload remaining value from those selected in Step 5. Cluster B has the maximum workload limit (496).

Step 7. Assign operations to the cluster that was selected (when an operation is allocated to a cluster, the tools associated with that operation are also allocated). By this method, one can determine which tools are allocated to which clusters we well as determine how many tool slots are required (Table 2).

Step 8. Update workload and tool capacity limits for the machine selected.

Step 9. Repeat until all operations are allocated.

4.3.2. SET UP FOR EACH HEURISTICS

Table 4 presents all of the Heuristics. Four Heuristics are shown in the first column. The second column shows how to select between clusters. The third column describes the means for selecting operations in column two. The last column describes the means for allocating operations to each cluster. The Multifit algorithm will be

implemented for each Heuristic case. A detailed explanation of the Multifit algorithm will be presented in section 4.3.4.2.

4.3.3. CONFIGURATIONS FOR EACH HEURISTICS

Heuristics	Between cluster	Among selected ones	Assign operations to clusters
Heuristic I	Minimum workload	Minimum workload	Largest remaining cluster
Heuristic II	Minimum workload	Minimum workload	Largest remaining cluster
Heuristic III	Maximum workload	Maximum workload	Largest remaining cluster
Heuristic IV	Maximum workload	Minimum workload	Largest remaining cluster

Table 4 Selection rule for operation assignments.

4.3.4. HEURISTIC APPROACH FOR PHASE II

After all of the operations are allocated to each cluster, Phase II employs two Heuristics within each cluster. When allocating operations to machines, operations will be assigned to the machine with the largest remaining workload capacity.

4.3.4.1. LPT (LONGEST PROCESSING TIME) ALOGRITHM

In the first Heuristic, an operation’s processing requirement is divided into the same number of batches for all operations for each machine cluster. These batches are allocated to machines by an LPT algorithm. A flow chart is shown in Figure 4.

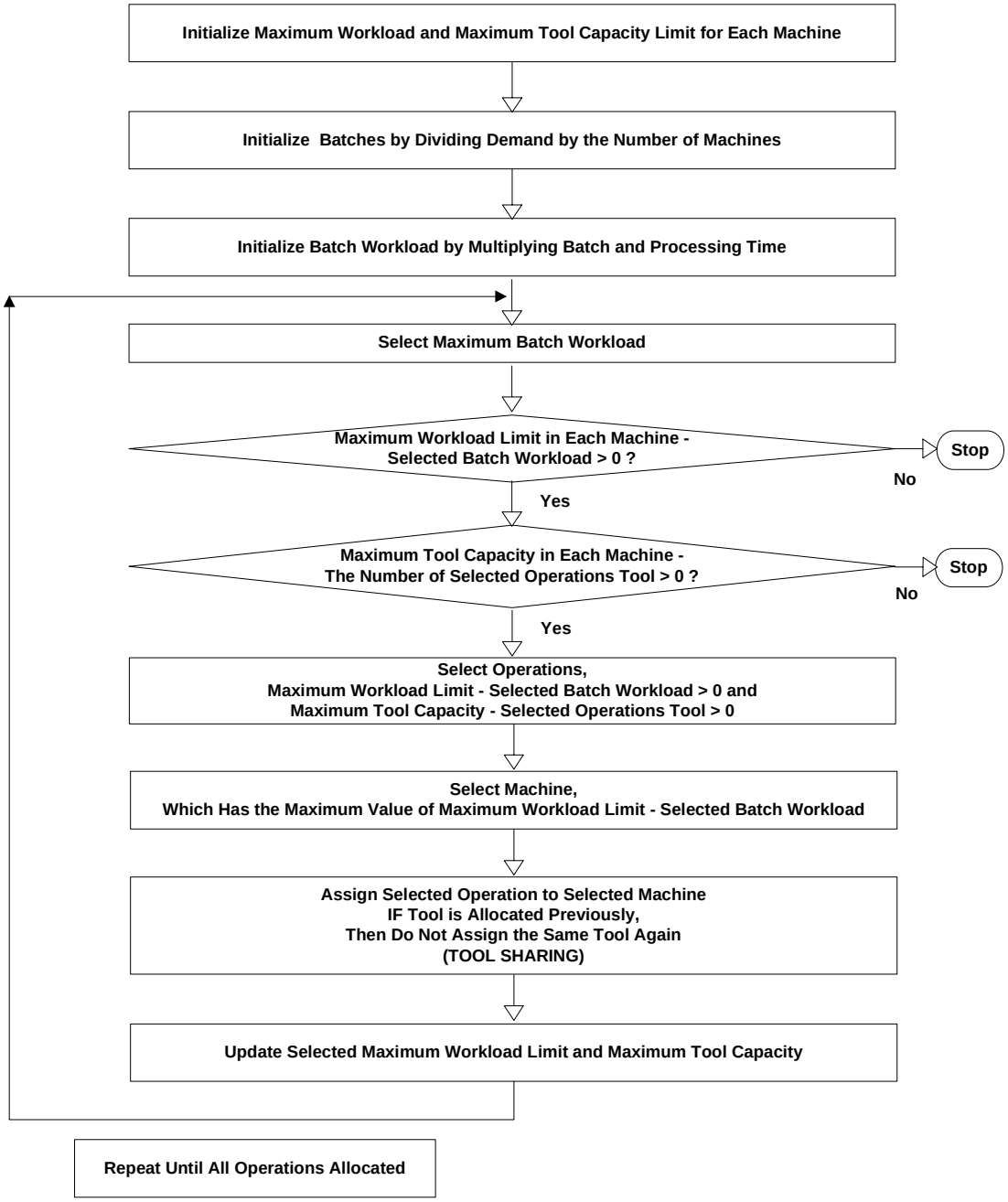

Figure 4 Flow chart for LPT algorithm.

General procedure for LPT algorithm in Phase I

Step 1. Initialize $\eta_m = Q_{cj}, \tau_m = C_{cj}$.

Step 2. Make batches for each operation by dividing demands of an operation by the number of machine.

Step 3. Select the batches with the maximum workload (batch×processing time) among the set of batches not yet allocated to machine.

Step 4. For each unassigned operation, determine the number of feasible machines.

Machine m is feasible if and only if $\eta_m \geq \frac{Q_{cj}}{m}$ and $\tau_m \geq \frac{\Delta\tau_m}{m}$.

Step 5. Assign the selected batch to a machine with the minimum workload allocated to it so far (ties are broken arbitrary).

Step 6. If any operations are not assigned, go to Step 3.

Table 5 is the data to illustrate example procedure for LPT algorithm in Phase II.

Operation number	Demand	Processing time for cluster type	Batches			Workload		
1	9	3	3	3	3	9	9	9
2	10	2	3	3	4	6	6	8
3	7	4	2	2	3	8	8	1
4	6	7	2	2	2	1	1	1
5	8	6	2	3	3	1	1	1

Table 5 Loading data for small example to illustrate LPT algorithm.

Example procedure for LPT algorithm in Phase II

Step 0. Initialize the workload limit as 200 and the tool magazine capacity as 50.

Step 1. In each machine, find the maximum workload that 18 in the workload column.

Step 2. Determine if the workload tool capacity limits are greater than 0.

For machine A, 200-18=182, for machine B, 200-18=182 and for machine C, 200-18=182. Tool capacity for each machine is 50-7=43, since the selected batch is in operation 2. Operation 5 needs 4 tools and these tools need 7 slots.

If a tool is already assigned to a machine, then do not allocate it again. This is the concept of tool sharing. If workload limit and tool capacity limit are less than 0, then this system is infeasible.

Step 6. Select the maximum workload remaining value from Step 2. Cluster A has a workload limit of 182.

Step 7. Assign operations to the machines that were selected (when an operation is allocated to a machine, the tools associated with that operation are also allocated). By this method, one can determine which tools are allocated to which machines we well as determine how many tool slots are required (Table 2).

Step 8. Update the workload and tool capacity limits for the selected machines.

Step 9. Repeat until all operations allocated.

4.3.4.2. MULTIFIT ALGORITHM

A Multifit algorithm is implemented in Phase I, and the detailed procedure is explained here. In the second Heuristic, each batch is allocated to machines by a Multifit algorithm, which is a multi-pass algorithm for bin-packing problems. To find a near optimal solution, it makes repeated trials for batch assignments with different machine capacity values (processing time capacities). A bisection search method provides the smallest machine capacity that can accommodate the allocation of all the batches.

Multifit algorithms are usually used in scheduling problems (Coffman et al. 1978), where they tend to exhibit better performance than LPT algorithms. Multifit algorithms can be applied to a loading problem, when the loading problem is presented as a bin-packing problem (Kim and Yano 1993). As in the case of scheduling problems, Multifit algorithms also tend to exhibit better performance than LPT algorithms. In this algorithm, the First Fit Decreasing (FFD), and Best Fit Increasing (BFI) rules are used to assign operations to machines.

Best Fit Decreasing (BFD) rules are implemented for testing, but the results are unsatisfactory; therefore, we have decided to apply the BFI rule rather than the BFD rule. FFD and BFI sort all operations in a non-increasing order of workloads and allocate them to machines in that order. While FFD allocates each operation to the lowest-indexed machine into which the operation can be allocated without violating the machine capacity, BFI allocates each operation to the machine that will have the largest remaining capacity after the operation is allocated to it. The Multifit algorithm will be stopped when the GAP is less than 0.01. A flow chart is shown in Figure 5.

When applying the Multifit algorithm in Phase I, those assignment methodologies will be used instead of FFD and BFI rules.

$$\text{GAP} = \left[\frac{\text{Upper Bound - Lower Bound}}{\text{Upper Bound}} \right] \times 100$$

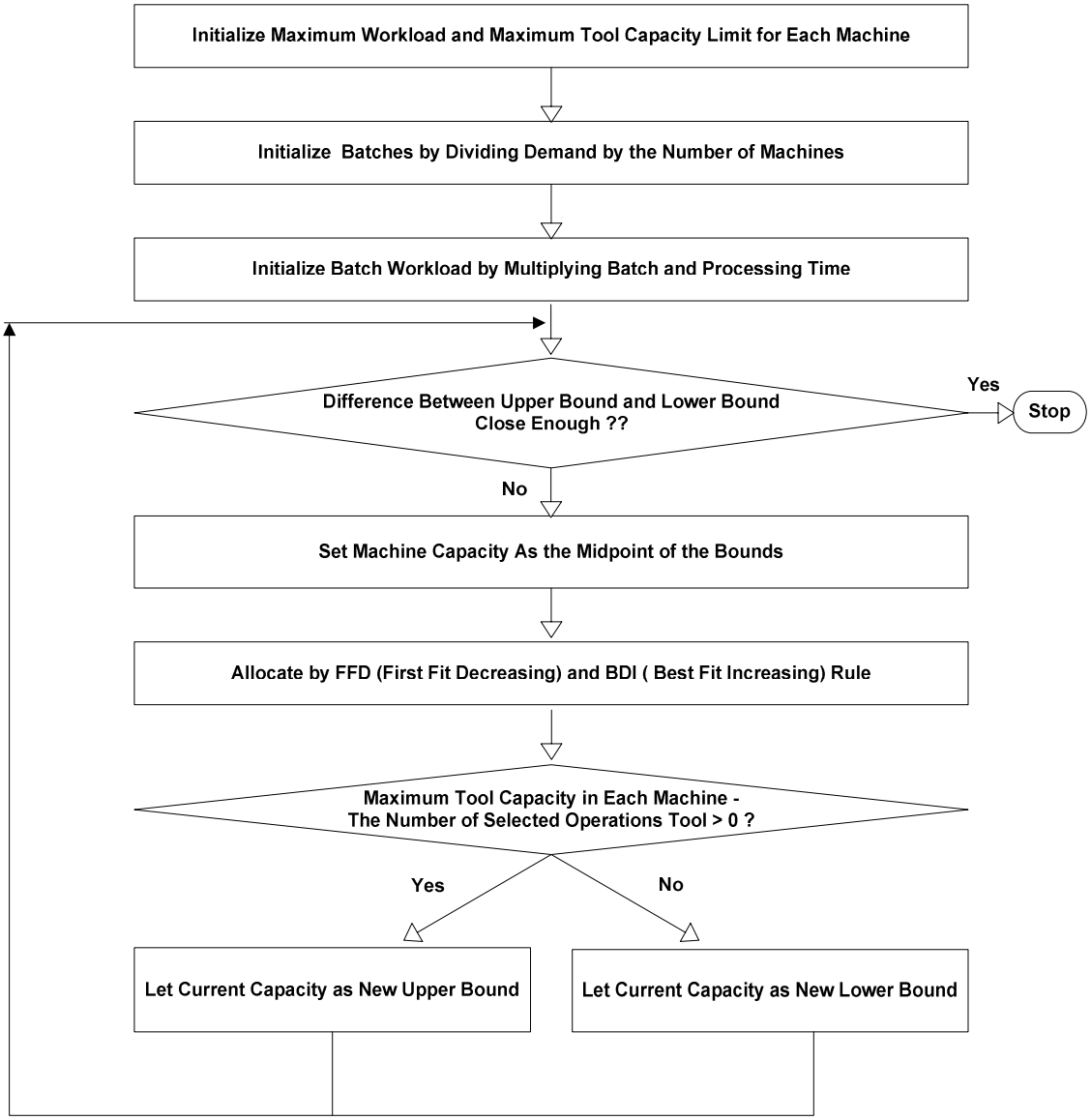

Figure 5 Flow chart for Multifit algorithm in Phase I.

General procedure for Multifit algorithm in Phase I

Step 1. Initialize $\eta_m = Q_{cj}, \tau_m = C_{cj}$.

Step 2. Make batches for each operation by dividing an operation's demands by the number of machines.

Step 3. Initialize the upper and the lower bounds on the machine capacities.

Step 4. If the difference of the lower and the upper bounds is close enough, stop. Otherwise, set the machine capacities to be the midpoints of the bounds.

Step 5. For each unassigned operation, determine the number of feasible machines.

Machine m is feasible if and only if $\eta_m \geq \frac{Q_{cj}}{m}$ and $\tau_m \geq \frac{\Delta\tau_m}{m}$.

Step 6. Allocate the batches to machines by the FFD (and BFI) rule. If all the batches can be assigned by these rules, let the current machine capacity be a new upper bound. Otherwise, let the current capacity be a new lower bound and go to Step 4.

4.4. EFFECTIVENESS OF HEURISTIC METHODS

To measure the effectiveness of each Heuristic, numerical experiments will be conducted for randomly generated data and the computational times recorded and the solution compared with the optimal solution; the results will be examined to determine which types of data instances result in favorable objective function values.

4.5. SUMMARY

This chapter presented heuristic procedures for the loading problem-assigning operations to clusters and allocating operations to machines within clusters. The

heuristics were analyzed for worst-case computational efficiency. In the next chapter, the heuristics will be tested in order to determine actual computational efficiency, system utilization as well as performance standards.

CHAPTER V

DISCUSSION, NUMERICAL RESULTS, CONCLUSION AND RECOMMEDATIONS

This chapter presents the results of the numerical experiments employed to test the heuristics developed in CHAPTER IV for the loading problems.

The following is an overview of the chapter. Loading problem data sets will be generated to demonstrate heuristic performance. It entails an explanation of the means in which performance ratios, relative performance ratios and utilization are obtained for clusters in Phase I and machines in Phase II. A simulation test will determine how many times the program needs to be run.

The tool capacity will be defined by means of a simulation test, then a reasonable workload limit for clusters and for machines will be determined. The number of operations to run will then be determined. Final configurations for running this program will then be made.

Unfortunately, it is extremely difficult to obtain actual data for a wide variety of systems because most are proprietary. In order to overcome this limitation, test problems have been generated randomly to ensure that the resulting data represent real systems relatively well. In fact, the parameters for the problems were derived from the reference author's experience at a manufacturing company. The resulting data reflects FMS within that company.

This data was generated by means of a discrete uniform distribution in the C programming language (appendix B). Non-uniform distributions require information about the mean and variance of the dataset. Since we do not have any information about our dataset, uniform distribution (which assumes that data is evenly distributed over the range) is the most suitable distribution for this problem.

Test case data was randomly generated using the parameters and test levels provided in Table 6 and Table 7. The heuristics were implemented in the C programming language (appendix B) on a Personal Computer.

When implementing these problems, the parameters are undefined. Accordingly, performance and utilization may vary. Good parameters yield good output, while bad parameters yield poor output. Without predefined parameters, good and reasonable output must be determined by means of trial and error. We will discuss the means in which parameters are defined in section 5.2 PROCEDURE.

Parameter	Range of values
Demand	Between 5 and 30
The processing time	Between 1 and 30
The number of tools for each operations	Between 5 and 10
The number of tool slots needed for each tool	1, with probability 0.7 2, with probability 0.1 3, with probability 0.2
Total number of tools used in this tests	80

Table 6 Generating loading problem data sets.

#of cluster	The number of machine	The number of operation	Tool capacity for cluster and machine	Workload limit for the cluster	Workload limit for the machine
3	4	90	110, 140	9200	2300
3	6	140	110, 140	13800	2300
3	8	190	110, 140	18400	2300
4	4	140	110, 140	9200	2300
4	6	200	110, 140	13800	2300
4	8	280	110, 140	18400	2300
5	4	190	110, 140	9200	2300
5	6	280	110, 140	13800	2300
5	8	360	110, 140	18400	2300

Table 7 Configurations for running loading problems.

5.1. DISCUSSION

The heuristics were implemented in the C programming language on a Personal Computer with a 2.40 GHz Intel Pentium 4 processor with 512 MB DDR SDRAM (appendix B).

Performance ratio, relative performance ratio and utilization may vary according to generated numbers. Sometimes, there is no output whatsoever, because randomly generated number can exceed workload or tool capacity limit. In order to further reduce the tolerance, this program was run 150 times for each case.

Solutions from the algorithms are compared with each other using the performance ratio as an index; this is the ratio to a lower bound, i.e., $\sum_{i \in I} (p_i \times D_i) / |C|$ for Phase I, and $\sum_{i \in I} (p_i \times D_i) / |J|$ for Phase II. Furthermore, performance ratios are obtained in this way: $\frac{\text{(Output of Heuristic-Lower Bound)}}{\text{Lower Bound}}$. Lastly, relative performance ration is obtained from this following procedure, first acquire the ratio of a solution to a lower bound and find a best ratio (H_B), which was obtained form this problem. Then a relative performance ratio is defined as $[(H_h - H_B)] / H_h] \times 100$, where H_B is the best performance ratio and H_h is the performance ratio from the Heuristic.

Utilizations for each clusters are obtained in this way by means of the $\frac{\text{Largest Value of Workload for clusters}}{\text{Cluster Workload Limit}}$, machine utilization in each cluster is can be obtained by: $\frac{\text{Largest Output Workload for each Machine in each cluster}}{\text{Machine Worload Limit}}$. We must choose the largest output workload for each cluster and machine since the largest workload can cover all of the operations. If we choose the small output workload, then that workload cannot cover the largest one.

5.2. PROCEDURE

To obtain optimum performance ratio and utilization, we need to know which parameter yield reasonable outputs. First, we need to know how many times to run the program. The means in which the data is generated determines the output value (and also

determines if there is any output). We will begin with the smallest possible number of repetitions, in order to conserve time. To determine this number, we run the program from 100 to 1000 times. We tried this in Heuristic I. The specific condition is defined in Table 8 shows that there is no significant difference between 100 and 1000. Since there are no significant difference between 100 and 1000, we will begin with 150.

Number of run	# of cluster	# of machine	# of operation	# of tool capacity	Cluster workload limit	Machine workload limit	Cluster performance	Cluster utilization	Machine performance	Machine utilization
100	3	4	90	140	9200	2300	0.3093	94.16	0.2978	93.32
150	3	4	90	140	9200	2300	0.3076	92.8	0.295	91.9
200	3	4	90	140	9200	2300	0.3017	93.67	0.2903	92.82
300	3	4	90	140	9200	2300	0.307	92.89	0.2958	92.07
400	3	4	90	140	9200	2300	0.3101	92.75	0.2991	91.95
500	3	4	90	140	9200	2300	0.308	93.71	0.2964	92.86
600	3	4	90	140	9200	2300	0.3082	92.96	0.2965	92.09
700	3	4	90	140	9200	2300	0.3072	93.19	0.2956	92.35
800	3	4	90	140	9200	2300	0.3043	93.39	0.293	92.56
900	3	4	90	140	9200	2300	0.3071	93.1	0.2955	92.25
1000	3	4	90	140	9200	2300	0.3062	93.27	0.2948	92.44

Table 8 Computation experience to test the number of running programs.

Once the number of program repetitions has been decided for each case, the tool capacity can be defined. Tool capacity 100 and 170 case is infeasible. Table 9, describe how performance improves when tool capacity increase from 110 to 140. Tool capacities of 141 to 160 do not exhibit significant difference in performance, so they are,

effectively, a waste of tool capacity. Therefore, we will begin with a tool capacity of 110 for the worst case, and 140 for the best case.

# of cluster	# of machine	# of operation	**# of tool capacity**	Cluster workload limit	Machine workload limit	Cluster performance	Cluster utilization	Machine performance	Machine utilization
3	4	80	**110**	8000	2300	0.3342	94.41	0.3177	93.23
3	4	80	**115**	8000	2300	0.3245	94.61	0.3089	93.47
3	4	80	**120**	8000	2300	0.3183	94.08	0.3026	92.94
3	4	80	**130**	8000	2300	0.3110	94.39	0.2977	93.41
3	4	80	**140**	8000	2300	0.3092	94.12	0.2961	93.16
3	4	80	**150**	8000	2300	0.3102	94.17	0.2971	94.17
3	4	80	**160**	8000	2300	0.3092	93.33	0.2970	92.44

Table 9 Computation experience to test the tool capacity.

The next step is finding a reasonable workload limit for clusters and machines. After fixing tool capacity with 110 and 140, we test the workload from 8000 to 9200. Machine workload limit is cluster workload limit divided by the number of machines. As seen in Table 10, in the below, there is no significant difference between the various numbers of workload limits. With these results, we can predict that workload limit does not effect to performance or relative performance ratio. Therefore, we need to check the utilization for each different workload limit.

# of cluster	# of machine	# of operation	# of tool capacity	**Cluster workload limit**	**Machine workload limit**	Cluster performance	Cluster utilization	Machine performance	Machine utilization
3	4	80	110	**8000**	**2000**	0.3342	94.41	0.3177	93.23
3	4	80	110	**8400**	**2100**	0.3414	92.69	0.3236	92.69
3	4	80	110	**8800**	**2200**	0.3479	90.56	0.3275	89.19
3	4	80	110	**9200**	**2300**	0.3544	87.51	0.3327	86.09
3	4	80	140	**8000**	**2000**	0.3092	94.12	0.2961	93.16
3	4	80	140	**8400**	**2100**	0.3083	91.58	0.2951	91.58
3	4	80	140	**8800**	**2200**	0.3070	88.73	0.2939	87.83
3	4	80	140	**9200**	**2300**	0.3050	85.43	0.2921	85.44

Table 10 Computation experience to test the workload limit.

To find out the number of operations, we have to run the program with varying numbers of operations. When there are 80 operations and the tool capacity is 110 and the cluster workload limit is 8000, utilization is over 90% -- a good number. However, if we continue to execute 80 operations and raise the tool capacity to 140 and the cluster workload limit to 9200, utilization decreases below 90%. Accordingly, when performing 80 operations, a cluster workload limit was set at 8000. Many other cases were dealt with in a similar fashion. As a result of these trials, we found that more than 90 operations are infeasible. Therefore a maximum limit of 90 operations and, in this case, a cluster workload limit of 9200 is reasonable for each tool capacity. This procedure is shown in Table 11.

# of cluster	# of machine	**# of operation**	# of tool capacity	Cluster workload limit	Machine workload limit	Cluster performance	Cluster utilization	Machine performance	Machine utilization
3	4	**80**	110	8000	2300	0.3342	94.41	0.3177	93.23
3	4	**80**	110	9200	2300	0.3544	87.51	0.3327	86.09
3	4	**80**	140	8000	2300	0.3092	94.12	0.2961	93.16
3	4	**80**	140	9200	2300	0.3050	85.43	0.2921	85.44
3	4	**90**	110	9200	2300	0.3384	94.7	0.3238	93.66
3	4	**90**	140	9200	2300	0.3106	93.81	0.2988	92.94
3	4	**100**	**infeasible**						

Table 11 Computation experience to test the number of operations.

Now that the number of operations, the tool capacity, the cluster workload limit and the machine workload limit have been defined, we can perform the test for each different cluster and machine case to get those numbers.

5.3. RESULTS AND ANALYSIS

The results of loading problem computational experiments are subsequently defined. On the tables below, machine workload limit is fixed at 2300 per machine.

Tables 12, 13, and 14 present the loading problem performance ratio for 8 Heuristics in Phase I for each cluster. Heuristic II performs well in the cluster 3 case and Heuristic III performs well in cluster 4 and cluster 5 cases. Tables 15, 16, and 17 show the results of the loading problem performance ratio for 8 Heuristics in Phase II for each

# of machine	# of operation	# of tools	Cluster work-load limit	Heuristic I	Heuristic II	Heuristic III	Heuristic IV	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
4	90	110	9200	0.2426	0.2055	0.2182	0.2492	0.2459	0.2086	0.2210	0.2522
4	90	140	9200	0.2324	0.1970	0.2069	0.2303	0.2350	0.1997	0.2098	0.2331
6	140	110	13800	0.2326	0.2008	0.2150	0.2346	0.2356	0.2037	0.2174	0.2376
6	140	140	13800	0.2341	0.1919	0.2134	0.2325	0.2363	0.1947	0.2156	0.2352
8	190	110	18400	0.2294	0.2047	0.2174	0.2329	0.2318	0.2075	0.2201	0.2356
8	190	140	18400	0.2307	0.1963	0.2081	0.2293	0.2333	0.1986	0.2107	0.2320
Average				0.2363	0.1994	0.2131	0.2348	0.2336	0.2021	0.2158	0.2376

Table 12 Phase I performance in cluster 3.

# of machine	# of operation	# of tools	Cluster work-load limit	Heuristic I	Heuristic II	Heuristic III	Heuristic IV	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
4	90	110	9200	0.2931	0.2713	0.2721	0.2936	0.2956	0.2734	0.2747	0.2959
4	90	140	9200	0.2878	0.2579	0.2639	0.2875	0.2901	0.2609	0.2664	0.2901
6	140	110	13800	0.2957	0.2832	0.2804	0.2939	0.2978	0.2860	0.2831	0.2960
6	140	140	13800	0.2839	0.2658	0.2690	0.2829	0.2860	0.2684	0.2712	0.2851
8	190	110	18400	0.2854	0.2848	0.2749	0.2838	0.2878	0.2872	0.2774	0.2863
8	190	140	18400	0.2845	0.2740	0.2704	0.2844	0.2868	0.2762	0.2729	0.2867
Average				0.2884	0.2729	0.2718	0.2877	0.2907	0.2753	0.2743	0.2900

Table 13 Phase I performance in cluster 4.

# of machine	# of operation	# of tools	Cluster work-load limit	Heuristic I	Heuristic II	Heuristic III	Heuristic IV	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
4	90	110	9200	0.3316	0.3253	0.3209	0.3278	0.3344	0.3276	0.3239	0.3305
4	90	140	9200	0.3216	0.3173	0.3136	0.3269	0.3242	0.3199	0.3169	0.3298
6	140	110	13800	0.3285	0.3343	0.3302	0.3319	0.3308	0.3368	0.3330	0.3348
6	140	140	13800	0.3246	0.3317	0.3180	0.3285	0.3269	0.3339	0.3206	0.3313
8	190	110	18400	0.3304	0.3455	0.3315	0.3320	0.3329	0.3482	0.3342	0.3346
8	190	140	18400	0.3254	0.3428	0.3226	0.3238	0.3282	0.3454	0.3248	0.3264
Average				0.3270	0.3328	0.3228	0.3285	0.3296	0.3353	0.3256	0.3312

Table 14 Phase I performance in cluster 5.

# of machine	# of operation	# of tools	Cluster work-load limit	Heuristic I LPT	Heuristic II LPT	Heuristic III LPT	Heuristic IV LPT	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
4	90	110	9200	0.2343	0.1921	0.2156	0.2440	0.2364	0.1944	0.2178	0.2462
4	90	140	9200	0.2257	0.1905	0.2059	0.2292	0.2280	0.1929	0.2081	0.2315
6	140	110	13800	0.2284	0.1952	0.2142	0.2333	0.2304	0.1978	0.2167	0.2355
6	140	140	13800	0.2292	0.1866	0.2130	0.2313	0.2316	0.1894	0.2154	0.2333
8	190	110	18400	0.2264	0.2007	0.2171	0.2317	0.2287	0.2035	0.2193	0.2343
8	190	140	18400	0.2276	0.1925	0.2078	0.2285	0.2301	0.1952	0.2104	0.2309
Average				0.2286	0.1929	0.2123	0.2330	0.2309	0.1955	0.2146	0.2353

Table 15 Phase II performance in cluster 3.

# of machine	# of operation	# of tools	Cluster work-load limit	Heuristic I LPT	Heuristic II LPT	Heuristic III LPT	Heuristic IV LPT	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
4	90	110	9200	0.2884	0.2628	0.2707	0.2918	0.2906	0.2650	0.2727	0.2938
4	90	140	9200	0.2829	0.2528	0.2631	0.2859	0.2849	0.2550	0.2649	0.2881
6	140	110	13800	0.2921	0.2783	0.2799	0.2929	0.2939	0.2802	0.2818	0.2946
6	140	140	13800	0.2804	0.2619	0.2687	0.2819	0.2822	0.2638	0.2707	0.2840
8	190	110	18400	0.2829	0.2820	0.2747	0.2830	0.2854	0.2842	0.2770	0.2857
8	190	140	18400	0.2822	0.2716	0.2703	0.2838	0.2845	0.2740	0.2726	0.2862
Average				0.2848	0.2682	0.2712	0.2866	0.2869	0.2704	0.2733	0.2887

Table 16 Phase II performance in cluster 4.

# of machine	# of operation	# of tools	Cluster work-load limit	Heuristic I LPT	Heuristic II LPT	Heuristic III LPT	Heuristic IV LPT	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
4	90	110	9200	0.3280	0.3207	0.3202	0.3261	0.3301	0.3227	0.3222	0.3280
4	90	140	9200	0.3183	0.3130	0.3129	0.3256	0.3204	0.3151	0.3150	0.3276
6	140	110	13800	0.3262	0.3312	0.3298	0.3311	0.3280	0.3329	0.3313	0.3327
6	140	140	13800	0.3220	0.3290	0.3176	0.3278	0.3238	0.3306	0.3193	0.3296
8	190	110	18400	0.3286	0.3434	0.3311	0.3316	0.3307	0.3455	0.3332	0.3336
8	190	140	18400	0.3236	0.3407	0.3225	0.3235	0.3259	0.3428	0.3247	0.3257
Average				0.3245	0.3297	0.3223	0.3276	0.3265	0.3316	0.3243	0.3296

Table 17 Phase II performance in cluster 5.

# of machine	# of operation	# of tools	Cluster work-load limit	Heuristic I	Heuristic II	Heuristic III	Heuristic IV	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
4	90	110	9200	0.1023	0.0584	0.0729	0.1102	0.1063	0.0621	0.0762	0.1137
4	90	140	9200	0.0913	0.0494	0.0418	0.0889	0.0945	0.0527	0.0453	0.0922
6	140	110	13800	0.1027	0.0654	0.0522	0.1049	0.1062	0.0688	0.0551	0.1085
6	140	140	13800	0.1069	0.0577	0.0406	0.1051	0.1095	0.0610	0.0432	0.1083
8	190	110	18400	0.0827	0.0533	0.0683	0.0868	0.0855	0.0566	0.0716	0.0900
8	190	140	18400	0.0818	0.0399	0.0683	0.0802	0.0850	0.0427	0.0431	0.0834
Average				0.0946	0.0540	0.0526	0.0960	0.0978	0.0573	0.0558	0.0994

Table 18 Phase I relative performance in cluster 3.

# of machine	# of operation	# of tools	Cluster work-load limit	Heuristic I	Heuristic II	Heuristic III	Heuristic IV	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
4	90	110	9200	0.0788	0.0505	0.0515	0.0795	0.0821	0.0531	0.0549	0.0825
4	90	140	9200	0.0852	0.0468	0.0348	0.0847	0.0882	0.0506	0.0382	0.0881
6	140	110	13800	0.0730	0.0566	0.0447	0.0706	0.0757	0.0602	0.0482	0.0734
6	140	140	13800	0.0694	0.0458	0.0500	0.0680	0.0721	0.0492	0.0528	0.0709
8	190	110	18400	0.0418	0.0411	0.0257	0.0396	0.0451	0.0443	0.0292	0.0431
8	190	140	18400	0.0512	0.0373	0.0325	0.0511	0.0543	0.0402	0.035	0.0542
Average				0.0666	0.0464	0.0399	0.0656	0.0696	0.0496	0.0432	0.0687

Table 19 Phase I relative performance in cluster 4.

# of machine	# of operation	# of tools	Cluster work-load limit	Heuristic I	Heuristic II	Heuristic III	Heuristic IV	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
4	90	110	9200	0.0692	0.0605	0.0543	0.0639	0.0731	0.0636	0.0585	0.0676
4	90	140	9200	0.0576	0.0516	0.0385	0.0650	0.0612	0.0553	0.0432	0.0689
6	140	110	13800	0.0450	0.0532	0.0474	0.0498	0.0483	0.0569	0.0515	0.0539
6	140	140	13800	0.0417	0.0518	0.0324	0.0472	0.0450	0.0548	0.0360	0.0512
8	190	110	18400	0.0470	0.0681	0.0486	0.0493	0.0506	0.0720	0.0524	0.0530
8	190	140	18400	0.0396	0.0643	0.0308	0.0373	0.0435	0.0680	0.0340	0.0410
Average				0.0464	0.0582	0.0420	0.0521	0.0496	0.0618	0.0459	0.0560

Table 20 Phase I relative performance in cluster 5.

# of machine	# of operation	# of tools	Cluster work-load limit	Heuristic I LPT	Heuristic II LPT	Heuristic III LPT	Heuristic IV LPT	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
4	90	110	9200	0.1280	0.0799	0.0729	0.1390	0.1303	0.0825	0.0756	0.1416
4	90	140	9200	0.0961	0.0550	0.0432	0.1002	0.0988	0.0578	0.0458	0.1029
6	140	110	13800	0.1025	0.0640	0.0526	0.1082	0.1048	0.0670	0.0556	0.1108
6	140	140	13800	0.1101	0.0608	0.0424	0.1125	0.1128	0.0641	0.0452	0.1148
8	190	110	18400	0.0800	0.0494	0.0690	0.0864	0.0828	0.0527	0.0716	0.0894
8	190	140	18400	0.0817	0.0400	0.0395	0.0827	0.0846	0.0431	0.0426	0.0856
Average				0.0997	0.0582	0.0533	0.1048	0.1024	0.0612	0.0561	0.1075

Table 21 Phase II relative performance in cluster 3.

# of machine	# of operation	# of tools	Cluster work-load limit	Heuristic I LPT	Heuristic II LPT	Heuristic III LPT	Heuristic IV LPT	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
4	90	110	9200	0.1003	0.0679	0.0513	0.1046	0.1031	0.0708	0.0540	0.1072
4	90	140	9200	0.0881	0.0497	0.0382	0.0919	0.0906	0.0526	0.0405	0.0947
6	140	110	13800	0.0779	0.0599	0.0451	0.0789	0.0802	0.0623	0.0477	0.0810
6	140	140	13800	0.0654	0.0414	0.0501	0.0673	0.0677	0.0438	0.0527	0.0700
8	190	110	18400	0.0385	0.0446	0.0278	0.0460	0.0419	0.0475	0.0309	0.0495
8	190	140	18400	0.0525	0.0386	0.0333	0.0546	0.0556	0.0417	0.0364	0.0578
Average				0.0704	0.0504	0.0410	0.0739	0.0732	0.0531	0.0437	0.0767

Table 22 Phase II relative performance in cluster 4.

# of machine	# of operation	# of tools	Cluster work-load limit	Heuristic I LPT	Heuristic II LPT	Heuristic III LPT	Heuristic IV LPT	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
4	90	110	9200	0.0663	0.0561	0.0554	0.0635	0.0692	0.0589	0.0582	0.0663
4	90	140	9200	0.0679	0.0606	0.0409	0.0779	0.0707	0.0634	0.0438	0.0806
6	140	110	13800	0.0473	0.0506	0.0496	0.0542	0.0498	0.0530	0.0518	0.0564
6	140	140	13800	0.0405	0.0505	0.0343	0.0487	0.0431	0.0527	0.0367	0.0514
8	190	110	18400	0.0475	0.0690	0.0481	0.0517	0.0505	0.0721	0.0511	0.0547
8	190	140	18400	0.0410	0.0652	0.0321	0.0408	0.0442	0.0682	0.0352	0.0439
Average				0.0504	0.0587	0.0434	0.0561	0.0531	0.0614	0.0461	0.0589

Table 23 Phase II relative performance in cluster 5.

# of machine	# of operation	# of tools	Cluster work-load limit	Heuristic I	Heuristic II	Heuristic III	Heuristic IV	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
4	90	110	9200	0.9359	0.9076	0.9113	0.9430	0.9434	0.9112	0.9146	0.9467
4	90	140	9200	0.9386	0.9020	0.9230	0.9316	0.9419	0.9051	0.9263	0.9350
6	140	110	13800	0.9576	0.9403	0.9358	0.9630	0.9614	0.9437	0.9387	0.9669
6	140	140	13800	0.9558	0.9292	0.9437	0.9485	0.9585	0.9324	0.9463	0.9519
8	190	110	18400	0.9649	0.9515	0.9649	0.9646	0.9678	0.9549	0.9683	0.9680
8	190	140	18400	0.9626	0.9425	0.9568	0.9707	0.9660	0.9452	0.9600	0.9741
Average				0.9526	0.9289	0.9392	0.9536	0.9560	0.9321	0.9424	0.9571

Table 24 Phase I utilization in cluster 3.

# of machine	# of operation	# of tools	Cluster work-load limit	Heuristic I	Heuristic II	Heuristic III	Heuristic IV	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
4	90	110	9200	0.9620	0.9518	0.9564	0.9622	0.9655	0.9545	0.9598	0.9653
4	90	140	9200	0.9615	0.9494	0.9349	0.9539	0.9646	0.9532	0.9381	0.9574
6	140	110	13800	0.9538	0.9411	0.9362	0.9473	0.9566	0.9447	0.9397	0.9501
6	140	140	13800	0.9444	0.9315	0.9353	0.9433	0.9472	0.9348	0.9380	0.9462
8	190	110	18400	0.9724	0.9762	0.9591	0.9701	0.9758	0.9795	0.9625	0.9736
8	190	140	18400	0.9735	0.9609	0.9608	0.9696	0.9766	0.9638	0.9642	0.9728
Average				0.9613	0.9518	0.9471	0.9577	0.9644	0.9551	0.9504	0.9609

Table 25 Phase I utilization in cluster 4.

# of machine	# of operation	# of tools	Cluster work-load limit	Heuristic I	Heuristic II	Heuristic III	Heuristic IV	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
4	90	110	9200	0.9710	0.9678	0.9599	0.9675	0.9752	0.9711	0.9642	0.9714
4	90	140	9200	0.9674	0.9729	0.9515	0.9655	0.9711	0.9767	0.9561	0.9695
6	140	110	13800	0.9489	0.9549	0.9554	0.9487	0.9522	0.9586	0.9594	0.9528
6	140	140	13800	0.9427	0.9562	0.9405	0.9451	0.9460	0.9593	0.9441	0.9491
8	190	110	18400	0.9515	0.9692	0.9517	0.9528	0.9551	0.9733	0.9556	0.9565
8	190	140	18400	0.9453	0.9640	0.9363	0.9497	0.9491	0.9678	0.9394	0.9534
Average				0.9545	0.9642	0.9492	0.9549	0.9581	0.9678	0.9531	0.9588

Table 26 Phase I utilization in cluster 5.

# of machine	# of operation	# of tools	Cluster work-load limit	Heuristic I LPT	Heuristic II LPT	Heuristic III LPT	Heuristic IV LPT	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
4	90	110	9200	0.9256	0.8925	0.9080	0.9361	0.9281	0.8950	0.9106	0.9389
4	90	140	9200	0.9305	0.8947	0.9216	0.9264	0.9333	0.8973	0.9242	0.9292
6	140	110	13800	0.9521	0.9336	0.9346	0.9611	0.9547	0.9366	0.9376	0.9640
6	140	140	13800	0.9496	0.9230	0.9431	0.9468	0.9525	0.9262	0.9458	0.9493
8	190	110	18400	0.9608	0.9465	0.9644	0.9629	0.9637	0.9498	0.9671	0.9661
8	190	140	18400	0.9586	0.9379	0.9563	0.9694	0.9617	0.9410	0.9594	0.9725
Average				0.9462	0.9214	0.9380	0.9505	0.9490	0.9243	0.9408	0.9533

Table 27 Phase II utilization in cluster 3.

# of machine	# of operation	# of tools	Cluster work-load limit	Heuristic I LPT	Heuristic II LPT	Heuristic III LPT	Heuristic IV LPT	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
4	90	110	9200	0.9555	0.9406	0.9542	0.9595	0.9585	0.9435	0.9569	0.9623
4	90	140	9200	0.9547	0.9427	0.9336	0.9516	0.9574	0.9456	0.9359	0.9546
6	140	110	13800	0.9488	0.9345	0.9353	0.9457	0.9512	0.9368	0.9378	0.9479
6	140	140	13800	0.9395	0.9264	0.9346	0.9418	0.9419	0.9288	0.9372	0.9445
8	190	110	18400	0.9688	0.9721	0.9585	0.9689	0.9722	0.9751	0.9616	0.9725
8	190	140	18400	0.9701	0.9575	0.9604	0.9685	0.9733	0.9606	0.9635	0.9717
Average				0.9563	0.9456	0.9461	0.9560	0.9591	0.9484	0.9488	0.9589

Table 28 Phase II utilization in cluster 4.

# of machine	# of operation	# of tools	Cluster work-load limit	Heuristic I LPT	Heuristic II LPT	Heuristic III LPT	Heuristic IV LPT	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
4	90	110	9200	0.9657	0.9610	0.9586	0.9648	0.9687	0.9638	0.9615	0.9676
4	90	140	9200	0.9625	0.9666	0.9503	0.9634	0.9654	0.9695	0.9532	0.9662
6	140	110	13800	0.9455	0.9502	0.9545	0.9473	0.9480	0.9526	0.9567	0.9496
6	140	140	13800	0.9389	0.9521	0.9397	0.9438	0.9414	0.9543	0.9420	0.9464
8	190	110	18400	0.9486	0.9658	0.9508	0.9519	0.9516	0.9690	0.9539	0.9549
8	190	140	18400	0.9425	0.9607	0.9359	0.9489	0.9456	0.9637	0.9389	0.9521
Average				0.9506	0.9594	0.9483	0.9533	0.9535	0.9622	0.9510	0.9561

Table 29 Phase II utilization in cluster 5.

Phase	# of cluster	Heuristic I	Heuristic II	Heuristic III	Heuristic IV	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
I	3	0.2363	0.1994	0.2131	0.2348	0.2336	0.2021	0.2158	0.2376
I	4	0.2884	0.2729	0.2718	0.2877	0.2907	0.2753	0.2743	0.2900
I	5	0.3270	0.3328	0.3228	0.3285	0.3296	0.3353	0.3256	0.3312

Table 30 Performance of Phase I.

Phase	# of cluster	Heuristic I	Heuristic II	Heuristic III	Heuristic IV	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
II	3	0.2286	0.1929	0.2123	0.2330	0.2336	0.2021	0.2158	0.2376
II	4	0.2848	0.2682	0.2712	0.2866	0.2907	0.2753	0.2743	0.2900
II	5	0.3245	0.3297	0.3223	0.3276	0.3296	0.3353	0.3256	0.3312

Table 31 Performance of Phase II.

Phase	# of cluster	Heuristic I LPT	Heuristic II LPT	Heuristic III LPT	Heuristic IV LPT	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
I	3	0.0946	0.0540	0.0526	0.0960	0.0978	0.0573	0.0558	0.0994
I	4	0.0666	0.0464	0.0399	0.0656	0.0696	0.0496	0.0432	0.0687
I	5	0.0464	0.0582	0.0420	0.0521	0.0496	0.0618	0.0459	0.0560

Table 32 Relative performance of Phase I.

Phase	# of cluster	Heuristic I LPT	Heuristic II LPT	Heuristic III LPT	Heuristic IV LPT	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit
II	3	0.0997	0.0582	0.0533	0.1048	0.1024	0.0612	0.0561	0.1075
II	4	0.0704	0.0504	0.0410	0.0739	0.0732	0.0531	0.0437	0.0767
II	5	0.0504	0.0587	0.0434	0.0561	0.0531	0.0614	0.0461	0.0589

Table 33 Relative performance of Phase II.

Phase	# of Tool	Heuristic I	Heuristic II	Heuristic III	Heuristic IV	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit	Average
I	110	0.2855	0.2728	0.2734	0.2866	0.2881	0.2754	0.2761	0.2893	0.2809
I	140	0.2806	0.2639	0.2651	0.2807	0.2830	0.2664	0.2677	0.2833	0.2738

Table 34 Performance by the number of tools in Phase I.

Phase	# of Tool	Heuristic I LPT	Heuristic II LPT	Heuristic III LPT	Heuristic IV LPT	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit	Average
II	110	0.2817	0.2674	0.2726	0.2851	0.2838	0.2696	0.2747	0.2872	0.2777
II	140	0.2769	0.2598	0.2646	0.2797	0.2790	0.2621	0.2668	0.2819	0.2714

Table 35 Performance by the number of tools in Phase II.

Phase	# of Tool	Heuristic I	Heuristic II	Heuristic III	Heuristic IV	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit	Average
I	110	0.0714	0.0563	0.0517	0.0727	0.0748	0.0597	0.0553	0.0762	0.0648
I	140	0.0698	0.0496	0.0427	0.0700	0.0730	0.0530	0.0429	0.0734	0.0593

Table 36 Relative performance by the number of tools in Phase I.

Phase	# of Tool	Heuristic I LPT	Heuristic II LPT	Heuristic III LPT	Heuristic IV LPT	Heuristic I Multifit	Heuristic II Multifit	Heuristic III Multifit	Heuristic IV Multifit	Average
II	110	0.0765	0.0602	0.0524	0.0814	0.0792	0.0630	0.0552	0.0841	0.0690
II	140	0.0715	0.0513	0.0393	0.0752	0.0742	0.0542	0.0421	0.0780	0.0607

Table 37 Relative performance by the number of tools in Phase II.

The number of tool	Average run time
110	0.3648
140	0.2820

Table 38 Average runtime by the number of tools.

The number of cluster	Average run time
3	0.1336
4	0.3016
5	0.5349

Table 39 Average runtime by the number of clusters.

Heuristics	Run time
Heuristic I	0.3175
Heuristic II	0.3153
Heuristic III	0.3319
Heuristic IV	0.3289

Table 40 Average runtime by Heuristics.

cluster. The application of the Heuristic II and the LPT algorithms performs well in the cluster 3 and cluster 4 case, while Heuristic III performs well in the cluster 5 case.

Tables 18, 19, and 20 present the loading problem relative performance ratio for 8 Heuristics in Phase I for each cluster. Heuristic III performs well in the cluster 3, 4 and 5 cases. Tables 21, 22, and 23 show the loading problem relative performance ratio for 8 Heuristics in Phase II for each cluster. The implementation of the Heuristic III and LPT algorithms performs better than the other Heuristics in all cluster cases.

Tables 24, 25, 26, 27, 28 and 29 show the utilization for all clusters and machines. As the number of operations increases, the utilization also increases. Therefore, we can maximize the utilization by increasing the number of operations. As the number of operations increase, workload also increases. So, we can increase the number of operations up to the cluster and machine workload limits.

Tables 30, 31, 32 and 33 present the average and relative performance for all configurations in each cluster. In regard to tool capacity, a large tool capacity provides better performance than a small tool capacity. This result has been demonstrated already in many previous papers. A loose tool capacity can be more freely allocated to clusters and machines in all cases. If tool capacity is bound too tightly, tool capacity bounds must be satisfied first and the application of algorithms becomes difficult.

Tables 34, 35, 36 and 37 show the performance ratio and relative performance ratio according to the number of tools in each Phase. When the tool capacity is loose, the performance and relative performance ratios are better. Table 38 presents the run time according to tool capacity. When the tool capacity is loose, run time is short, because

operations can find machines easily. Table 39 shows the run time according to the number of clusters. As the number of clusters increase, run time also increases. Table 40 presents the run time according to each Heuristic. Different Heuristics do not exhibit significantly different run times.

Given these results, one can infer that if performance in Phase I is good, then performance in Phase II will be similarly good. Accordingly, we can also infer that a Phase I solution can effect a Phase II solution. Many people have solved the Phase II problem in various ways, but this dissertation is the first known attempt to solve the Phase I problem. In fact, Phase I performance can be further improved if a better heuristic can be determined.

Although the loading problem may have to be solved several times in order to obtain a production and setup plan for a given set of orders, it is not likely that the problem will have to be solved very frequently or quickly for real time decisions in most real systems.

We can get a heuristic output in a reasonably short run time, so we can apply this system in a real factory. We can control cluster and machine utilization by increasing or decreasing the number of operations. Since we want to complete as many operations as possible, we need to choose the number of operations that maximizes utilization.

Overall, Heuristic II and Heuristic III performed better than any other Heuristic. To reiterate, there are different loading problem environments that result from the three ways machines are grouped in FMS (i.e., no grouping, partial grouping, and total grouping). Different grouping methods generate different loading problem situations,

which affect the performance of loading plans in various ways. Therefore, we compared the algorithms suggested in this study with existing algorithms developed for the total grouping configuration via a series of simulation experiments. Since no grouping is known to be inferior to total grouping in system throughput, the no grouping configuration is not included in the comparison (Stecke and Solberg 1985).

It has been shown that partial grouping performs better than other groupings (Lee and Kim 2000). This might be because partial grouping and corresponding loading plans help to cope with system disturbances (e.g. machine breakdowns) more easily by providing more routing flexibility.

5.4. CONCLUSION AND EXTENTIONS

This chapter summarizes the contributions of this dissertation and attempts to reveal more areas of future research.

CHAPTER II presented a survey of the literature describing the problems inherent in the loading problem. CHAPTER III developed a formal model as well as integer linear programming formulations for the problems inherent in the loading problem. There are two integer linear programming formulations for the sub-problems. CHAPTER IV proposed a Heuristic for each formulation presented in CHAPTER III. CHAPTER V presented computational results from testing the Heuristics that were presented in CHAPTER IV.

This research focused on the loading problem with 3, 4, and 5 clusters and 4, 6, and 8 machines within those clusters. The goal of this research was to obtain a balanced workload in order to minimize the maximum workload. We expect this result will reduce

lead times and operating costs. Furthermore, such outputs may be obtained in a reasonable running time. In order to obtain this balanced workload, we introduced a partial grouping method. By applying this method, we can divide the demand into small batches so that demand can be distributed in small batches to the machines. Several good algorithms were suggested for the loading problem that results from distributing batches.

To compare the suggested algorithms with an existing one, computational experiments were performed on randomly generated test problems. Research on the loading problem for the partial grouping configuration may be more important since the partial grouping configuration not only provides better performance than the total grouping configuration but it also provides a more practical or realistic alternative for obtaining pooling effects from a small to a large number of machines.

In analyzing this loading problem, this research has made several contributions. The first was the identification and formal modeling of the loading problem. The integer linear programming models presented were extensions of known models for the loading problem. The main problem was initially formulated into an integer linear programming model and then it was decomposed into two sub-problems, which were also formulated into integer linear programming models. These two sub-problems were solved by means of a hierarchical approach.

A second contribution was the ability to deal with different processing times in different clusters for the same operation. This reflects the scenario in which an operation can be processed on machines of different types. In such a scenario, loading algorithms

based on bin packing algorithms could not be used. Therefore, a new algorithm had to be developed.

A third contribution was the development of heuristic procedures for the loading problem. Since existing bin-packing procedures could not be used, a newly invented assignment algorithm was implemented. The heuristics were analyzed for theoretical worst-case performance.

A fourth contribution was the implementation of a two-phase method that made large-scale loading problems possible. Previous loading problems were limited to small numbers of machine problems; however, a two-phase approach made it possible to solve a large number of machine problems. And we could know which Heuristic is performed better than the other Heuristics in each Phase by accomplishing relative performance ratio.

Finally, the heuristics procedures have been tested using randomly generated data. The heuristics were computationally efficient for problems of varying size and the results were interesting from a theoretical and practical perspective. However, the heuristics should be tested using actual case study data.

This research can be extended in several ways. It would be interesting to combine the loading algorithm with those of other related problems such as the part type selection problem or the scheduling problem. Since the part type selection and loading problems are interrelated with each other, the two problems may have to be solved simultaneously. In which case, the loading algorithm suggested in this dissertation can be used as a sub-routine for an algorithm for part type selection.

Additional tooling costs, such as tool purchase costs and other tool-related costs, should also be considered in order to provide more flexibility in grouping. For example, determining the number of tool copies for each tool type is another important decision problem in operating FMS. Tradeoffs between the costs and the benefits of having more tools must be analyzed in order to solve such tool requirements planning or tool provisioning problems.

A well-balanced workload system is the goal, and partial grouping is a critical means of obtaining that goal. Partial grouping yields a more balanced workload because it entails the subdivision of demands into several batches. If we can divide demands into smaller batches and then allocate these batches to machines then we can get a more balanced workload. Accordingly, knowing the constitution of a batch can make a big difference.

There are alternative means for selecting operations. For example, at the first step of the Heuristic, we chose either the minimum or maximum workloads between clusters; however, selecting minimum or maximum workloads between operations might also be justified. These alternatives may or may not yield significantly different results. Nonetheless, the implementation of several alternatives may help illuminate the loading problem in FMS.

In Phase I and Phase II, when implementing a MULTIFIT algorithm, operations may be assigned to the smallest remaining capacity clusters instead of to the largest remaining capacity clusters

REFERENCES

Ammons, J.C., Lofgren, C.B., and McGinnis, L.F., 1985, A large scale machine loading problem in flexible assembly. *Annals of Operations Research*, **3**, 319-322.

Askin, R.G. and Standridge, C. R., 1993, *Modeling and Analysis of Manufacturing Systems*. (New York: John Wiley & Sons).

Berrada, M. and Stecke, K.E., 1986, A branch and bound approach for machine load balancing in flexible manufacturing systems. *Management Science*, **32** (10), 1316-1335.

Bretthauser, K. M. and Venkataramanan, M. A., 1990, Machine loading and alternate routing in a flexible manufacturing system. *Computers and Industrial Engineering*, **18** (3), 341-150.

Carrie, A.S. and Perera, D. T. S., 1986, Work scheduling in FMS under tool availability constraints. *International Journal of Production Research*, **24** (6), 1299-1308.

Chen, Y-J and Askin, R.G., 1990, A multiobjective evaluation of flexible manufacturing system loading system heuristics. *International Journal of Production Research*, **28** (5), 895-911.

Chen, F. F., Ker, J-I., and Kleawpatinon, K., 1995, An effective part-selection model for production planning of flexible manufacturing systems. *International Journal of Production Research*, **33** (10), 2671-2683.

Choi, S-H and Lee J.S., 1998, A heuristic approach to machine loading problem in non-preemptive flexible manufacturing systems. *International Journal of Industrial*

Engineering, **5** (2), 105-116.

Coffman, Jr., E.G., Garey, M.R., and Johnson, D.S., 1978, An application of bin-packing to multiprocessor scheduling. *Society for Industrial and Applied Mathematics Journal on Computing*, **7** (1), 1-17.

Denizel, M. and Sayin, S., 1998, Part-type selection in flexible manufacturing systems: a bicriteria approach with due dates. *Journal of Operational Research Society*, **49**, 659-669.

Dobson, G., 1984, Scheduling independent tasks on uniform processors. *Society for Industrial and Applied Mathematics Journal on Computing*, **13** (4), 705-710.

Friesen, D. K., 1984, Tighter bounds for the multifit processor scheduling algorithm. *Society for Industrial and Applied Mathematics Journal on Computing*, **13** (1), 170-181.

Friesen, D. K., 1987, Tighter bounds for LPT scheduling on uniform processors. *Society for Industrial and Applied Mathematics Journal on Computing*, **16** (3), 554-560.

Friesen, D. K. and Langston, M. A., 1983, Bounds for MULTIFIT scheduling on uniform processors. *Society for Industrial and Applied Mathematics Journal on Computing*, **12** (1), 60-70.

Friesen, D. K. and Langston, M. A., 1986, Variable sized bin packing. *Society for Industrial and Applied Mathematics Journal on Computing*, **12** (1), 60-70.

Garey, M.R. and Graham, R. L., 1975, Bounds for multiprocessor scheduling with resource constraints. *Society for Industrial and Applied Mathematics Journal on Computing*, **4** (2), 187-200.

Garey, M.R., Graham, R. L., Johnson, D. S., and Yao, A. C., 1976, Resource constrained scheduling as generalized bin packing. *Journal of Combinatorial Theory(A)*, **21**, 257-298.

Garey, M.R. and Johnson, D. S., 1981, Approximation algorithm for bin packing problems. In G. Ausiello and M. Lucertini, (Eds.), *A Survey, in Analysis and Design of Algorithms in Combinatorial Optimization*, (Springer-Verlag, New York).

Gonzalez, T., Ibarra, O. H., and Sahni, S., 1977, Bounds for LPT schedules on uniform processsors. *Society for Industrial and Applied Mathematics Journal on Computing*, **6** (1), 155-166.

Graham, R.L., 1969, Bounds on multiprocessor timing anomalies. *Society for Industrial and Applied Mathematics Journal on Applied Mathematics*, **17**, 416-429.

Greene, T. J. and Sadowski, R. P., 1986, A mixed integer program for loading and scheduling multiple flexible manufacturing cells. *European Journal of Operational Research*, **24**, 379-386.

Guerrero, F., Lozano, S., Koltai, T., and Larraneta, J., 1999, Machine loading and part type selection in flexible manufacturing systems. *International Journal of Production Research*, **37** (6), 1303-1392.

Ham, I., Hitomi, K., and Yoshida, T., 1985, *Group Technology: Application to Production management*, (Boston: Kluwer-Nijhoff Pub.).

Ibarra, O. H. and Kim, C. E., 1977, Heuristic algorithms for scheduling independent tasks on nonidentical processors. *Journal of the Association for Computing Machinery*, **24** (2), 280-289.

Kato, K., 1995, An integrated approach for loading, routing, and scheduling in flexible manufacturing systems. *IEEE*, **31**, 299-310

Kim, Y-D., 1988, An iterative approach for system setup problem of flexible manufacturing systems. Ph. D. Dissertation, The University of Michigan.

Kim, Y-D., 1993, A study on surrogate objectives for loading a certain type of flexible manufacturing systems. *International Journal of Production Research,* **31**, 381-392.

Kim, Y-D and Yanco, C.A., 1993, Heuristic approaches for loading problems in flexible manufacturing systems. *IIE Transactions*, **25** (1), 26-39.

Kim, Y-D and Yanco, C.A., 1994, A new branch and bound algorithm for loading problems in flexible manufacturing systems. *International Journal of Flexible Manufacturing Systems*, **6**, 361-382.

Kou, L. T. and Markosky, G., 1977, Multidimensional bin packing algorithms. *IBM Journal of Research and Development*, **21**, 443-448.

Kuhn, H., 1995, A heuristic algorithm for the loading problem in flexible manufacturing systems. *The International Journal of Flexible Manufacturing Systems*, **7**, 229-254.

Kumar, P., Tewari, N.K., and Singh, N., 1990, Joint consideration of grouping and loading problems in a flexible manufacturing system. *International Journal of*

Production Research, **28**, 1345-1356.

Kunde, M. and Steppat, H., 1985, First fit decreasing scheduling on uniform processors. *Discrete Applied Mathematics*, **10**, 165-177.

Kusiak, A., 1985, Loading models in flexible manufacturing systems. Flexible Manufacturing. In Raouf, A. and Ahmad, S. I. (Eds.), *Recent Development on FMS, Robotics, CAD/CAM, CIM*, (Elsevier Science Publishers Amsterdam, B. V.), pp. 119-132.

Langston, M. A., 1987, A study of composite heuristic algorithms. *Journal of Operational Research Society,* **38** (6), 539-544.

Lashkari, R. S., Dutta, S. P., and Padhye, A. M., 1987, A new formulation of operation allocation problem in flexible manufacturing systems: Mathematical modeling and computational experience, *International Journal of Production Research,* **25** (9), 1267-1283.

Lee, D-H and Kim, Y-D., 2000, Loading algorithms for flexible manufacturing systems with partially grouped machines. *IIE Transactions*, **32**, 33-47.

Maruyama, K., Chang, S. K., and Tang, D. T., 1977, A general packing algorithm for multidimensional resource requirement. *International Journal of Computer and Information Sciences*, **6** (2), 131-149.

Moreno, A. A. and Ding, F-Y., 1993, Heuristics for the FMS-Loading and part type selection problems, *The International Journal of Flexible Manufacturing Systems*, **5**, 287-300.

Mukhopadhyay, S. K., Midha, S., and Murli Krishna, V., 1992, A heuristic procedure for

loading problems in flexible manufacturing system. *International Journal of Production Research,* **30**, 2213-2228.

Mukhopadhyay, S.K., Singh, M. K., and Srivastava R., 1998, FMS machine loading: a simulated annealing approach. *International Journal of Production Research,* **36** (6), 1529-1547.

Nayak, G. K. and Acharya, D., 1998, Part type selection, machine loading and part type volume determination problems in FMS planning. *International Journal of Production Research,* **36** (7), 1801-1824.

Nemhauser, G.L. and Wolsey, L.A., 1988, Integer and combinatorial optimization, (New York: John Wiley & Sons).

Rajagopalan, S., 1986, Formulation and heuristic solutions for parts grouping and tool loading in flexible manufacturing systems. *Proceedings of the 2nd ORSA/TIMS Conference on Flexible Manufacturing Systems*, Ann Arbor, MI, pp. 312-314.

Ram, B., Sarin, S., and Chen, C. S., 1990, A model and a solution approach for the machine loading and tool allocation problem in a flexible manufacturing system, *International Journal of Production Research*, **28** (4), 637-645.

Rupe, J. and Kuo, W., 1997, Solutions to a modified tool loading problem for a single FMM. *International Journal of Production Research*, **35** (8), 2253-2268.

Saad, M. S., Baykasoglu A., and Gindy, N.Z. N., 2002, A new integrated system for loading and scheduling in cellular manufacturing. *International Journal of Computer Integrated Manufacturing*, **15** (1), 37-49.

Sarin, S. C. and Chen, C. S., 1987, The machine loading and tool allocation problem in a flexible manufacturing system. *International Journal of Production Research*, **25** (7), 1081-1094.

Shanker, K. and Srinivasulu, A., 1989, Some solution methodologies for loading problems in a flexible manufacturing systems. *International Journal of Production Research*, **27** (6), 1019-1034.

Shanker, K. and Tzen, Y-J., 1985, A loading and dispatching problem in a random flexible manufacturing system. *International Journal of Production Research*, **23**, 579-595.

Shanthikumar, J.G. and Stecke, K.E., 1986, Reducing Work-In-Process inventory in certain types of flexible manufacturing systems. *European Journal of Operational Research*, **26** (2), 266-271.

Stecke, K.E., 1983, Formulation and solution of nonlinear integer production planning problem for flexible manufacturing systems. *Management Science*, **29** (3), 273-288.

Stecke, K.E., 1986, A hierarchical approach to solving machine grouping and loading problems of flexible manufacturing systems. *European Journal of Operational Research*, **24**, 369-378.

Stecke, K.E. and Morin, T. L., 1985, The optimality of balancing workloads in certain types of flexible manufacturing systems. *European Journal of Operational Research*, **20** (1), 68-82.

Stecke, K.E. and Raman, N, 1994, Production planning decisions in flexible

manufacturing systems with random material flows. *IIE* Transactions, **26** (5), 2-17

Stecke, K.E. and Solberg, J.J., 1981, Loading and control policies for a flexible manufacturing system. *International Journal of Production Research*, **19**, 481-490.

Stecke, K.E. and Solberg, J.J., 1985, The optimality of unbalancing both workloads and machine group sizes in closed queueing networks of multiserver queues. *Operations Research*, **33** (4), 882-910.

Stecke, K. E. and Talbot, F. B., 1985, Heuristics for loading flexible manufacturing systems. In Raouf, A. and Ahmad, S.I. (Eds.), *Flexible Manufacturing: Recent Developments in FMS, Robotics, CAD/CAM, CIM*, (Elsevier Science Publishers Amsterdam, B. V.), pp. 73-85.

Tang, L-L, Yih Y., and Liu C-Y., 1995, A framework for part type selection and scheduling in FMS environments. *International Journal of Computer Integrated Manufacturing*, **8** (2), 102-115.

Yao, A. C., 1980, New algorithms for bin packing. *Journal of the Association for Computing Machinery*, **27** (2), 207-227.

Yao, D. D., 1985, Some properties of the throughput function of closed networks of queues. *Operations Research Letters*, **3** (6), 313-317.

Yao, D. D. and Kim, S. C., 1987a, Some order relations in closed networks of queues with multiserver stations. *Naval Research Logistics*, **34** (1), 53-66.

Yao, D. D. and Kim, S. C., 1987b, Reducing the congestion in a class of job shops. *Management Science*, **33** (9), 1165-1172.

APPENDIX A

C COMPUTER LISTING FOR LOADING DATA GENERATION

Programming Language: ANSI c
Programming Environment: PC using Pentium 4 processor 2.40 Ghz

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

#define Z 10000
#define Capa 80      // or 100
#define oper_num 20  // or 30, 40
#define max 500
#define mach 4         // or 6, 8
#define TOOL 80
#define NEWTOOL 10

void generating_number();
void run_terminate();

// # of operations: 20, 30, 40
// # of machines: 4, 6, 8
// Capacity of a tool magazine: 80 or 100
// Processing time DU(1,10)
// # of tools needed for each operation DU(5,10)
// # of tool slots needed for each tool 1,2,3 with prob of 0.7, 0.1, 0.2

int p_time[max];
int num_tools[max];
int demand[max];
int opertool[max][TOOL];
int ot[max][NEWTOOL];
int lslot[TOOL];

clock_t start, finish;

FILE *ofp;

void main(void)
{
        ofp=fopen("output.out","w");

        generating_number();
```

```
	start=clock();

	run_terminate();
}

void generating_number()
{
	int i,j;
	int otool;
	int count;
	int temp;
	int ottemp[max];

	srand((unsigned)time(NULL));

	for (i=0;i<oper_num;i++)
	{
		p_time[i]=1+rand()%10
	}
	for (i=0;i<oper_num;i++)
	{
		demand[i]=5+rand()%26;
		//generate demand by Uniform Dist. [5,30]
	}

	for (i=0;i<oper_num;i++)
	{
		ottemp[i]=5+rand()%6;
	}
	for (i=0;i<oper_num;i++)
	{
		count=0;
		for (j=0;j<ottemp[i];j++)
		{
		otool=rand()%TOOL;

		if(opertool[i][otool]==0)
		{
			opertool[i][otool]=1;
			count++;
		}
		else j--;
		}
	}
	for (i=0;i<oper_num;i++)
	{
		for (j=0;j<NEWTOOL;j++)
```

```
                {
                        ot[i][j]=-1;
                }
        }
        for (i=0;i<oper_num;i++)
        {
                count=0;
                for (j=0;j<TOOL;j++)
                {
                        if (opertool[i][j]==1)
                        {
                                ot[i][count]=j;
                                count++;
                        }
                }
        }
        for(i=0;i<TOOL;i++)
        {
                temp=rand()%100;

                if(temp<70) lslot[i]=1;
                else if(temp>=70&&temp<90) lslot[i]=3;
                else lslot[i]=2;
        }
}
void run_terminate()
{
        double duration;
        finish=clock();
        duration=(double)(finish-start)/CLOCKS_PER_SEC;
        fprintf(ofp, "run time = % .10f second \n",duration);
}
```

APPENDIX B

C COMPUTER LISTING FOR HEURISTICS

Programming Environment
Heuristic I
5 Clusters
6 Machines
280 Operations
110 Cluster Tool Capacity
110 Machine Tool Capacity
13800 Cluster Workload
2300 Machine Workload

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

#define cluster 5
#define batches 5
#define machine 6
#define ops_num 280
#define clus_capa 110
#define machine_tool_capa 110
#define cluster_work 13800
#define machine_work_limit 2300

#define max 1000
#define max_num 1000000
#define min_num 0
#define TOOL 80
#define NEWTOOL 10
#define FOUND 1
#define NOTFOUND 0
#define whole_block ops_num*batches

void run_terminate();
void const_batch();
void get_workload();
void min_cluster();
void max_cluster();
void assign_cluster();
void LPT();
int LPT_1();
```

```
void machine_assign(int);
void tool_check(int);
void multifit();

int p_time[ops_num][cluster];
int demand[ops_num];
int min_ptime[ops_num];
int ottemp[ops_num];
int opertool[ops_num][TOOL];
int ot[ops_num][NEWTOOL];
int lslot[TOOL];
int batch[ops_num][cluster];
int quot[max];
int remain[max];
int clu_workload[ops_num][cluster][cluster];
int min_clus_work[cluster], min_oper[cluster], min_batch[cluster];
int max_clus_work, max_oper, max_batch, max_clus;
int clus_work[cluster];
int tool_capa[cluster];
int final_workload[cluster];
float utilization[cluster];
int end_batch[cluster][ops_num*batches];
int machine_batch[cluster][ops_num];
int num_whole_block[cluster];
int tool_name[cluster][TOOL];
int num_tool_name[cluster];
int clus_machine_batch[cluster][ops_num][machine];
int machine_batch_workload[cluster][ops_num][machine];
int max_machine;
int m_work_limit[cluster][machine];
int m_tool_capa[cluster][machine];
int m_workload_judge[cluster][machine];
int m_tool_name[cluster][machine][TOOL];
int m_max_clus_work[cluster];
int m_max_oper[cluster];
int m_max_machine[cluster];
int condition;
int total_tool_num;
int m_total_tool_num;
int machine_num;
int same_tool;
int temp_work_judge;
int judge_tool_num;
int m_temp_tool_slot[cluster][machine];
int m_num_tool_name[cluster][machine];
int m_tool_judge[cluster][machine];
int m_temp_work;
int m_real_workload[cluster][machine];
```

```
int m_answer_workload[cluster];
float m_performance[cluster];
float m_utilization[cluster];
float machine_work;
int UB[cluster];
int LB[cluster];
int current_capa[cluster];
int max_LPT[cluster];
float multi_performance[cluster];
float multi_utilization[cluster];

struct STORE_DATA
{
        int oper;
        int batch;
        int pre_cluster;
};


struct STORE_DATA ALL_TOGETHER[cluster][whole_block];

clock_t start, finish;

FILE *c_ofp, *m_ofp, *multi_ofp, *time_ofp;

void main(void)
{
        int i,j,k,answer_workload,LB, LB_cluster,LB_machine;
        float performance,mean_utilization,sum_utilization;

        c_ofp=fopen("c_output.out","a");
        m_ofp=fopen("m_output.out","a");
        multi_ofp=fopen("multi_output.out","a");
        time_ofp=fopen("time_output.out","a");

        start=clock();
        const_batch();

        get_workload();

        for (j=0; j<cluster;j++)
        {
                clus_work[j]=cluster_work;
                num_whole_block[j]=-1;

                tool_capa[j]=clus_capa;
                num_tool_name[j]=0;
                for(i=0;i<TOOL;i++){
```

```
                tool_name[j][i]=-1;
        }
}

for (j=0; j<cluster;j++)
{
        for (i=0; i<ops_num; i++)
        {
                machine_batch[j][i]=-1;
        }
}

for (j=0; j<ops_num*batches; j++){
        min_cluster();
        max_cluster();
        assign_cluster();

}

for(i=0;i<ops_num;i++){
        min_ptime[i]=max_num;
        for(j=0;j<cluster;j++){
                if(p_time[i][j]<min_ptime[i]){
                        min_ptime[i]=p_time[i][j];
                }
        }
}

LB=0;
for(i=0;i<ops_num;i++){
        LB=LB+(min_ptime[i]*demand[i]);
}

LB_cluster=LB/cluster;
LB_machine=LB / (cluster*machine);

for(j=0; j<cluster; j++){
        final_workload[j]=cluster_work-clus_work[j];
}

sum_utilization=0;
for(j=0;j<cluster;j++){
        utilization[j]= (final_workload[j]/float(cluster_work))*100;
        sum_utilization=sum_utilization+utilization[j];
}

answer_workload=min_num;
for(j=0;j<cluster;j++){
```

```
		if(final_workload[j]>answer_workload){
			answer_workload=final_workload[j];
		}
}

mean_utilization=float (answer_workload)/cluster_work;

performance=(float(answer_workload)-LB_cluster) / LB_cluster;

LPT();

for(j=0;j<cluster;j++){
		for(k=0;k<machine;k++){
			m_num_tool_name[j][k]=0;
			for(i=0;i<TOOL;i++){
				m_tool_name[j][k][i]=-1;
			}
		}
}

for (i=0; i<ops_num*machine; i++){
		condition=LPT_1();
		if (condition==1)
			break;
}

for(j=0;j<cluster;j++){
		for(k=0;k<machine;k++){
			m_real_workload[j][k]=machine_work_limit-m_work_limit[j][k];
		}
}

for(j=0;j<cluster;j++){
		m_answer_workload[j]=min_num;
}
for(j=0;j<cluster;j++){
		for(k=0;k<machine;k++){
			if(m_real_workload[j][k]>m_answer_workload[j]){
				m_answer_workload[j]=m_real_workload[j][k];
			}
		}
}
for(j=0;j<cluster;j++){
		machine_work = float(cluster_work) / machine;
		m_utilization[j]=float(m_answer_workload[j]) / machine_work;
}

for(j=0;j<cluster;j++){
```

```
		m_performance[j]=(float(m_answer_workload[j])-LB_machine) / LB_machine;
	}

	multifit();

	fprintf(c_ofp,"%f		%f\n", performance,mean_utilization);
	for(j=0;j<cluster;j++){
		fprintf(m_ofp,"%f		%f\n", m_performance[j],m_utilization[j]);
		fprintf(multi_ofp,"%f		%f\n", multi_performance[j],multi_utilization[j]);
	}

	run_terminate();
	exit(0);
}

void const_batch()
{

	int i,j;
	for(i=0;i<ops_num;i++)
	{
		for(j=0;j<cluster;j++)
		{
			batch[i][j]=demand[i]/cluster;
		}
			quot[i]=demand[i]/cluster;
			remain[i]=demand[i]-quot[i]*cluster;
		for(j=0;j<remain[i];j++)
		{
			batch[i][j]=batch[i][j]+1;
		}
	}
}
void get_workload()
{
	int i,j,k;

	for (j=0;j<cluster;j++)
	{
		for (i=0;i<ops_num;i++)
		{
			for (k=0;k<batches;k++)
			{
				clu_workload[i][j][k]=batch[i][k]*p_time[i][j];
			}
		}
	}
}
```

```
void min_cluster()
{
	int i,j,k;

	for (j=0;j<cluster;j++)
	{
		min_clus_work[j]=max_num;
		min_oper[j]=-1;
		min_batch[j]=-1;
		for (i=0;i<ops_num;i++)
		{
			for (k=0;k<batches;k++)
			{
				if (clu_workload[i][j][k]!=-1){
				if (clu_workload[i][j][k]<min_clus_work[j]){
					min_clus_work[j]=clu_workload[i][j][k];

					min_oper[j]=i;
					min_batch[j]=k;
					}
				}
			}
		}
	}
}
void max_cluster()
{
	int j;

	max_clus_work=min_num;
	max_clus=-1;
	max_oper=-1;
	max_batch=-1;

	for (j=0; j<cluster;j++)
	{
		if (min_clus_work[j]>max_clus_work){
			max_clus_work=min_clus_work[j];
			max_oper=min_oper[j];
			max_batch=min_batch[j];
			max_clus=j;
		}
	}
}

void assign_cluster()
{
```

```
	int i,j, k,temp_work, cluster_num;
	int judge_work_num[cluster];
	int temp_work_judge,judge_tool_num;
	int tool_judge[cluster];
	int temp_tool_slot[cluster];
	int same_tool;

	static long int num_check = 0;

	for (j=0;j<cluster;j++){
		judge_work_num[j]=-1;
	}
	temp_work_judge=0;

	for (j=0;j<cluster;j++){
		if(clus_work[j]-clu_workload[max_oper][j][max_batch]>0){
			judge_work_num[j]++;
			temp_work_judge++;
		}
	}
	if (temp_work_judge==0){
		printf("Out of All Cluster workload\n");
		exit(1);
	}
	judge_tool_num=0;

	for (j=0;j<cluster;j++){
		if(judge_work_num[j]==0){
			temp_tool_slot[j]=0;
			for (i=0; i<ottemp[max_oper]; i++){
				temp_tool_slot[j]=temp_tool_slot[j]+lslot[ot[max_oper][i]];
			}
			for (i=0; i<ottemp[max_oper]; i++){
				for (k=0; k<TOOL; k++){
					if (tool_name[j][k]!=-1){
						if(ot[max_oper][i]==tool_name[j][k]){
							temp_tool_slot[j]=temp_tool_slot[j]-
lslot[ot[max_oper][i]];
						}
					}
				}
			}
			tool_judge[j]=tool_capa[j]-temp_tool_slot[j];
		}
		else{
			tool_judge[j]=0;
		}
		if(tool_judge[j]>0){
```

```
                judge_work_num[j]++;
                judge_tool_num++;
        }
}

if (judge_tool_num==0){
        printf("Out of Tool Capa even if Workload is available\n");
        exit(1);
}
j=0;
while(judge_work_num[j]<=0){
        j++;
}
temp_work=clus_work[j]-clu_workload[max_oper][j][max_batch];
cluster_num=j;
for(j=cluster_num+1;j<cluster;j++)
{
        if(judge_work_num[j]==1){
                if(temp_work<clus_work[j]-clu_workload[max_oper][j][max_batch]){
                cluster_num=j;
                temp_work=clus_work[j]-clu_workload[max_oper][j][max_batch];
                }
        }
}

total_tool_num=0;
while(total_tool_num<TOOL){
        if(tool_name[cluster_num][total_tool_num]==-1){
                break;
        }
        else{
                total_tool_num++;
        }
}
if(total_tool_num==TOOL+1){
        printf("ERROR out of tool space\n");
        exit(1);
}
for(j=0;j<ottemp[max_oper];j++){
        same_tool=NOTFOUND;
        i=0;
        while(i<total_tool_num){
                if(ot[max_oper][j]==tool_name[cluster_num][i]){
                        same_tool=FOUND;
                        break;
                }
                else{
                        i++;
```

```
                        }
                }
                if(i==total_tool_num){
                        tool_name[cluster_num][total_tool_num]=ot[max_oper][j];
                        total_tool_num++;
                        if(total_tool_num==TOOL+1){
                                printf("ERROR OVERFLOW\n");
                                exit(1);
                        }
                }
        }

        tool_capa[cluster_num]=tool_judge[cluster_num];

        clus_work[cluster_num]=temp_work;

        for(j=0; j<cluster; j++){
                clu_workload[max_oper][j][max_batch]=-1;
        }

        if (machine_batch[cluster_num][max_oper]==-1){
                machine_batch[cluster_num][max_oper]=batch[max_oper][max_batch];
        }
        else{

        machine_batch[cluster_num][max_oper]=machine_batch[cluster_num][max_oper]+batc
h[max_oper][max_batch];
        }

        num_whole_block[cluster_num]++;
        ALL_TOGETHER[cluster_num][num_whole_block[cluster_num]].batch=max_batch;
        ALL_TOGETHER[cluster_num][num_whole_block[cluster_num]].oper=max_oper;
        ALL_TOGETHER[cluster_num][num_whole_block[cluster_num]].pre_cluster=max_cl
us;

        for(j=0; j<cluster; j++){
                for(i=0; i<num_whole_block[j]; i++){
                        end_batch[j][i]=batch[ALL_TOGETHER[j][i].oper][max_clus];
                }
        }
}

void LPT()
{

        int i,j,k;

        for(j=0;j<cluster;j++){
```

```
		for(i=0;i<ops_num;i++){
			for(k=0;k<machine;k++){
				clus_machine_batch[j][i][k]=machine_batch[j][i]/machine;
			}
			quot[i]=machine_batch[j][i]/machine;
			remain[i]=machine_batch[j][i]-quot[i]*machine;
			for(k=0;k<remain[i];k++){
			clus_machine_batch[j][i][k]=clus_machine_batch[j][i][k]+1;
			}
		}
	}

	for(j=0;j<cluster;j++){
		for(i=0;i<ops_num;i++){
			for(k=0;k<machine;k++){

	machine_batch_workload[j][i][k]=clus_machine_batch[j][i][k]*p_time[i][j];
			}
		}
	}

	for(j=0;j<cluster;j++){
		for(i=0;i<ops_num;i++){
			for(k=0;k<machine;k++){
				if(machine_batch_workload[j][i][k]==0){
					machine_batch_workload[j][i][k]=-1;
				}
			}
		}
	}

	for(j=0;j<cluster;j++){
		for(k=0;k<machine;k++){
			m_work_limit[j][k]=machine_work_limit;
			m_tool_capa[j][k]=machine_tool_capa;
		}
	}
}


int LPT_1()
{
	int i,j,k,temp;

	for(j=0;j<cluster;j++){
		m_max_clus_work[j]=min_num;
		m_max_oper[j]=-2;
		m_max_machine[j]=-1;
```

```
	}

	for(j=0;j<cluster;j++){
		for(i=0;i<ops_num;i++){
			for(k=0;k<machine;k++){
				if(machine_batch_workload[j][i][k]>m_max_clus_work[j]){

	m_max_clus_work[j]=machine_batch_workload[j][i][k];
							m_max_oper[j]=i;
							m_max_machine[j]=k;
				}
			}
		}
	}

	temp=0;
	for(j=0;j<cluster;j++){
		if(m_max_clus_work[j]==-1){
			temp++;
		}
	}
	if(temp==cluster){
		return 1;
	}

	for(j=0;j<cluster;j++){
		if(m_max_clus_work[j]>0){

			machine_assign(j);


		}
	}
	return 0;
}

void machine_assign(int j){

	int i,k,g;

	for(k=0;k<machine;k++){
		m_workload_judge[j][k]=-1;
	}

	temp_work_judge=0;

	for(k=0;k<machine;k++){
		if(m_work_limit[j][k]-m_max_clus_work[j]>0){
```

```
                m_workload_judge[j][k]++;
                temp_work_judge++;
        }
}

if(temp_work_judge==0){
        printf("Out of All cluster workload in the machine\n");
        exit(4);
}

judge_tool_num=0;

for(k=0;k<machine;k++){
        if(m_workload_judge[j][k]==0){
                m_temp_tool_slot[j][k]=0;
                for(i=0;i<ottemp[m_max_oper[j]];i++){

m_temp_tool_slot[j][k]=m_temp_tool_slot[j][k]+lslot[ot[m_max_oper[j]][i]];
                }
                for(i=0;i<ottemp[m_max_oper[j]];i++){
                        for(g=0;g<TOOL;g++){
                                if(m_tool_name[j][k][g]!=-1){

if(ot[m_max_oper[j]][i]==m_tool_name[j][k][g]){

m_temp_tool_slot[j][k]=m_temp_tool_slot[j][k]-lslot[ot[m_max_oper[j]][i]];
                                                }
                                        }
                                }
                }
                m_tool_judge[j][k]=m_tool_capa[j][k]-m_temp_tool_slot[j][k];
        }
        else{
                m_tool_judge[j][k]=0;
        }
        if(m_tool_judge[j][k]>0){
                m_workload_judge[j][k]++;
                judge_tool_num++;
        }
}

if(judge_tool_num==0){
        printf("Out of Tool Capacity in the machine even if workload is available\n");
        exit(5);
}

k=0;
while(m_workload_judge[j][k]<=0){
```

```
                k++;
        }

        m_temp_work=m_work_limit[j][k]-m_max_clus_work[j];
        machine_num=k;

        for(k=machine_num+1;k<machine;k++){
                if(m_workload_judge[j][k]==1){
                        if(m_temp_work<m_work_limit[j][k]-m_max_clus_work[j]){
                                machine_num=k;
                                m_temp_work=m_work_limit[j][k]-m_max_clus_work[j];
                        }
                }
        }

        tool_check(j);

        m_tool_capa[j][machine_num]=m_tool_judge[j][machine_num];
        m_work_limit[j][machine_num]=m_temp_work;
        machine_batch_workload[j][m_max_oper[j]][m_max_machine[j]]=-1;

}

void tool_check(int j){

        int i,k;

        m_total_tool_num=0;
        while(m_total_tool_num<TOOL){
                if(m_tool_name[j][machine_num][m_total_tool_num]==-1){
                        break;
                }
                else{
                        m_total_tool_num++;
                }
        }

        if(m_total_tool_num>TOOL){
                printf("ERROR out of tool space in the machine\n");
                exit(3);
        }
        for(k=0;k<ottemp[m_max_oper[j]];k++){
                same_tool=NOTFOUND;
                i=0;
                while(i<m_total_tool_num){
                        if(ot[m_max_oper[j]][k]==m_tool_name[j][machine_num][i]){
                                same_tool=FOUND;
                                break;
```

```
                        }
                        else{
                                i++;
                        }
                }
                if(i==m_total_tool_num){

        m_tool_name[j][machine_num][m_total_tool_num]=ot[m_max_oper[j]][k];
                        m_total_tool_num++;
                        if(m_total_tool_num>TOOL){
                                printf("ERROR OVERFLOW in the machine\n");
                                exit(5);
                        }
                }
        }
}

void multifit()
{

        int i,j,k;

        for(j=0;j<cluster;j++){
                UB[j]=0;
                LB[j]=0;
        }

        for(j=0;j<cluster;j++){
                for(i=0;i<ops_num;i++){
                        if(machine_batch[j][i]!=-1){
                                UB[j]=UB[j]+machine_batch[j][i]*p_time[i][j];
                        }
                }
        }

        for(j=0;j<cluster;j++){
                printf(" UB is %d \n", UB[j]);

                LB[j]=UB[j]/machine;

                printf(" LB is %d \n", LB[j]);
        }

        for(j=0;j<cluster;j++){
                current_capa[j]=(UB[j]+LB[j])/2;
        }

        for(j=0;j<cluster;j++){
```

```
		for(k=0;k<machine;k++){
			m_real_workload[j][k]=machine_work_limit-m_work_limit[j][k];
		}
	}

	for(j=0;j<cluster;j++){

		max_LPT[j]=min_num;
	}

	for(j=0;j<cluster;j++){
		for(k=0;k<machine;k++){
			if(m_real_workload[j][k]>max_LPT[j]){
				max_LPT[j]=m_real_workload[j][k];
			}
		}
	}

	for(j=0;j<cluster;j++){
		while(UB[j]-LB[j]>710){
			current_capa[j]=(UB[j]+LB[j])/2;
			if(LB[j]<=max_LPT[j] && max_LPT[j]<=current_capa[j]){
				UB[j]=current_capa[j];
			}
			else{
				LB[j]=current_capa[j];
			}
		}
	}

	for(j=0;j<cluster;j++){
		multi_performance[j]=  ( UB[j] - LB[j] ) / float (LB[j]);
		multi_utilization[j]=float(LB[j]) / machine_work;
	}
}
```

VITA

NAME	Jong Hwan Lee
EDUCATIONAL BACKGROUND	B.S. Industrial Engineering 1997
	Dongguk University
	M.S. Industrial Engineering 1999
	Texas A&M University
	Ph.D. Industrial Engineering 2003
	Texas A&M University
PERMANENT ADDRESS	6106 Triangle Dr., Columbia MD 20144