AN HIERARCHICAL APPROACH TO MACHINE BATCHING, LOADING AND TOOL ALLOCATION PROBLEMS

by

Eleftherios Iakovou

1Research was supported in part by the National Science Foundation grant DDM-8819542.
AN HIERARCHICAL APPROACH TO MACHINE BATCHING, LOADING, 
AND TOOL ALLOCATION PROBLEMS.

Eleftherios Iakovou, Ph.D.
Cornell University 1992

Group technology (GT) is a manufacturing philosophy which strives to capitalize on similar activities. It can affect potentially all areas of a manufacturing organization. One specific application of GT is Cellular Manufacturing (CM), which involves processing collections of similar parts (part families) on dedicated clusters of machines (machine cells). Computer numerically controlled (CNC) machining centers are the basic components in metal parts fabrication systems often found in CM. Typical CNC machining centers have permanently attached tool magazines with limited capacities along with automatic tool changers. A well designed tooling system is essential for the effective use of this equipment.

The issues we are looking at appear at the operational level for a manufacturing system. We first address the issue of scheduling parts on a single machine so that the total setup time is minimized. We then look at problems that appear in multi-machine environments. The problem is to assign parts to machines, cluster parts to families based on their tooling requirements, and then to determine the sequence in
which the parts will be processed. In this case our goals are both setup minimization and workload balance.

To tackle the problem we develop an hierarchical scheme that assigns first the high volume parts to machines, and then the low volume ones. For all levels of the hierarchy, models and solution procedures are developed.

One of the major contributions of this thesis is the inclusion of variability and correlation for the demands among the parts in the models that determine the assignment of parts to machines. So far, no provisions are made in the literature for how similar systems could accommodate wide fluctuations in demand without the machines suffering from significant over- and under- capacity utilization, in the dynamic operation of the system.

In our models we formulate a quadratic integer problem. We develop a novel approach for solving it by showing its equivalence to a maximum network flow problem.

Our approach is an integrative one. It recognizes the decisions that have been made, the computational complexity of the problem, and the various constraints that could be included in the model.
Acknowledgements

I would like to thank Professor John A. Muckstadt for his guidance, encouragement, for his sense of humor and mainly for his vision! I thank him for the time that he spent with me, sharing some of his experience and enthusiasm in working with engineering problems. I thank him especially for helping me keep things in perspective. Professor Muckstadt has been both an advisor and a friend to me.

I would also like to thank Professor Kavindra Malik, for his friendship as well as his guidance and time throughout my last semester at Cornell. I particularly thank Professors Narahari Prahbu and Dean Taylor not only for serving on my committee, but also for the exciting courses in queuing theory and computer-aided design that they taught me.

Thanks go also to Professors David Ruppert and Robin Roundy for what they taught me as well as to Professor D.P. Psinos for his inspiration.

I owe so many things to my family. The completion of this degree is their achievement as well as mine. I thank my parents Thodoros and Evanthia, and my brother Yiannis for their love and support. I do hope that they are as proud of me as I am of them.
I regard myself very fortunate having made wonderful friends, living over the
last four and a half years in Ithaca. In particular I thank Professor Sid, “Happy
Harry”, Resnick for his wit, spirit, friendship, unforgettable courses in Stochastic
Processes, and for all the basketball games we played together.

I also had the privilege to play hoops with the following basketball stars: Enver
Yucesan, Kevin Healy, Lance Waller, Levent Tuncel, Jim Hobert, Dave Goldsman.

Ingri Johanesson, Richard Warkentin, Andrew Forbes, Nikos Pitsianis, Priamos
Georgiadis, Leonardo Zuniga, Dave Bell, Jorge Vera, Julio Stern, Stefan Scholtes,
Uday Rao, Dave Thomas, Neville Nagarwalla, Sridhar Tayur, Sharon Filipowski,
Yufei Wang, Sarah Stock, helped make my time at Cornell a really rewarding expe-
rience. I would also like to thank Janice Post, Patricia English and Kathy King for
their help and smile thus making endless days at the department much nicer.

I would never forget the people that have always been there for me since I came in
Ithaca: Nikos and Sava Plataniotis, Greg Katsoulis, Mike Georgiadis and Panayiotis
Stratakos.

Finally, thanks go to Nikos Gallis for setting an example of self-discipline, deter-
mination as well as fighting and winning spirit, and to Pavlos Sidiropoulos for songs
that have been accompanying me for years.
# Table of Contents

1 Introduction ................................................. 1
   1.1 Group Technology and Cellular Manufacturing ............... 1
   1.2 Focused Factories .......................................... 4
   1.3 The Effects of Reducing Setup Time .......................... 6
   1.4 Research Goals ............................................. 7

2 Models Arising from a Single Machine ......................... 9
   2.1 Introduction .................................................. 9
   2.2 Minimization of the Number of Tool Switches on a Flexible Machine ........................................... 10
   2.3 Formulation of the Problem ................................... 12
   2.4 Characteristics of the Problem ............................... 14
       2.4.1 Computational Results - Complexity ...................... 14
       2.4.2 Problem Reduction ...................................... 15
   2.5 Minimum number of tool switches for a fixed sequence ..... 15
   2.6 Lower Bounds ................................................ 16
       2.6.1 Linear Programming Relaxation of DT ................... 16
       2.6.2 Lagrangian Formulation .................................. 18
       2.6.3 Traveling Salesman Paths ............................... 19
   2.7 Heuristics for Job Scheduling ............................... 21
       2.7.1 Heuristics based on lb(i,j) .............................. 21
       2.7.2 Heuristics based on ub(i,j) .............................. 23
   2.8 Clustering of Jobs .......................................... 24

3 Multiple Machines with Setup Times as a Function of the Number of Switching Instances .................................................. 27
   3.1 Background for the Problem .................................. 27
   3.2 Formulation of the Problem .................................. 29
   3.3 Makespan : Literature Review - Complexity ................ 32
   3.4 Hierarchical Approach ....................................... 33
   3.5 Assignment Problem for the High Volume Parts ............. 36
3.6  A Solution Procedure for HVPA ........................................ 39
  3.6.1 Optimal Selection Model ........................................ 40
3.7  Solving the Lagrangean Dual ........................................... 42
  3.7.1 Features of the Lagrangean Algorithm .......................... 44
3.8  Assignment Problem for the Low Volume Parts .................. 46
3.9  A Solution Procedure for LVPA ....................................... 48
  3.9.1 Introduction ...................................................... 48
  3.9.2 MULTIFIT Algorithm ............................................ 49
  3.9.3 Algorithm ALLOCATE ........................................... 51

4  Multiple Machines with Setup Times as a Function of the Number of Tool Switches. .......................... 57
  4.1 Background for the Problem ......................................... 57
  4.2 Formulation of the Problem ......................................... 58
  4.3 Hierarchical Approach ................................................ 59
  4.4 Assignment of the High Volume Parts .............................. 60
  4.5 Assignment of the Low Volume Parts ............................... 62

5  Conclusions - Further Research ........................................ 65
  5.1 Summary .............................................................. 65
  5.2 Areas for Future Research ........................................... 67

Bibliography ............................................................... 70
List of Figures

3.1 A 20:80 Product/Quantity Pareto Graph .......................... 35
3.2 Optimal Selection Model ........................................... 43
3.3 Network for the HVPA problem ................................... 45
3.4 Network for the LVPA problem ................................. 55
Chapter 1

Introduction

In this thesis we present an approach for machining batches of parts on groups of machines in a way that accounts for machine loads and tool assignments. This allocation is related to principles underlying group technology, cellular manufacturing, and focused factories and to the importance of setup times. These topics have been discussed by many authors during the last two decades. We begin by discussing briefly each of these areas and conclude this opening chapter by defining precisely the goals and organization of this thesis.

1.1 Group Technology and Cellular Manufacturing

Group technology (GT) is a manufacturing philosophy that achieves economies throughout the manufacturing cycle by grouping similar parts into families. It is applied mainly to small- and medium-sized batch production systems. Through classification and coding systems, parts that have similar design and/or manufacturing
characteristics are grouped into families. The formation of part families based on design similarity results in the reduction of component variety. On the other hand, when families are formed based on the parts' manufacturing similarities, there is an impact upon the production process itself. In some cases, families identified by design similarity will also have similar manufacturing requirements; for example, parts in a family may require the same material and have the same specifications in terms of finish.

The manufacture of batches of parts has traditionally taken place in a functional layout where similar machines are placed together in one area of the production facility. In this case, during the production process, batches move through the various workcenters in a specified sequence of operations. Still, the notion of part families can be used in a functional layout, but one of the possible applications of GT is a rearrangement of the production facility. Rather than organizing the facility with machine similarities as the criterion, which is called a process orientation, different machines on which a part family can be produced are grouped together. This is called product orientation. In this case, the machines form a cell. This is the basis for cellular manufacturing (CM). Therefore, CM is a specific application of GT, which involves processing collections of similar parts, called part families, on dedicated clusters of machines, called cells.

CM is currently receiving a lot of attention from both practitioners and academicians. Japan has been using CM extensively as a crucial step to achieve just-in-time (JIT) manufacturing (Monden 1983, Schonberger 1982). Furthermore, CM has become quite popular among US and European manufacturers during the last few
years. It is also the development of new process technologies that support the application of CM. Developments in information systems, such as computerized part coding schemes, also facilitate family recognition and cell formation.

A specific instance of CM is flexible manufacturing systems (FMSs). They are computer-controlled configurations of semi-independent workstations often connected by automated material handling systems. They are designed to manufacture efficiently more than one kind of part at low to medium volumes. Jaikumar and Wassenhove (1989) classify FMSs into three categories based on the interdependence of machine operations for a given part and the space available for the storage of work in process.

1. Tightly coupled cellular systems. They have little intermediate storage between cells and a high degree of interdependence characterized by unique sequences of operations for many parts throughout many machines.

2. Transfer lines. They are similar to tightly coupled cellular systems; but, in this case, all the parts share the same sequence of operations.

3. Integrated, independent cells. They exhibit high storage between cells or low inter-dependence among them.

As numerically controlled machines (NC machines) of greater versatility have become prevalent, there has been an increase in the number of transfer lines and integrated independent cells, while the population of tightly coupled cellular systems has been decreasing. Small inventories are kept so that their flow is decoupled.
Transfer lines are used when the processing times on each machine are low, and integrated independent cells when processing times are high.

Efficient use of these technologies suggests a cell-structured approach to manufacturing.

Surveys on GT and CM are given by Burbridge (1979) and Hyer and Wemmerlov (1984). There is an extensive set of papers addressing the parts grouping problems. Surveys are provided by King and Nakornchai (1982), Kusiak (1985), Wemmerlov and Hyer (1986), and Vakharia (1987). Most of the classification schemes produce a binary matrix that provides information about the machines required for the processing of each part. The entry \((i,j)\) of the matrix is 1 if machine \(i\) is necessary for the production of part \(j\). This matrix is used for the formation of part groups and equivalently of machine cells. The methods that are used can be classified into:


1.2 Focused Factories

The concept of the focused factory (Skinner 1974) is not mutually exclusive from the notion of group technology. It promotes the idea of focusing a plant on a set of parts that share similar volume, market and technology. In today’s competitive
manufacturing arena, products proliferate rapidly and the product life cycles are becoming shorter. Consequently a facility has to produce at the same time low volume specialty parts, high volume parts with relatively stable demand patterns, along with parts that exhibit a rapidly increasing or decreasing demand rates.

Skinner (1974) proposes the segmentation of a plant into sub-plants dedicated to different groups of products. Of course, this partition might not always be feasible. For example, there might be key equipment that has to be shared by all parts, for instance, heat treating equipment.

The key characteristics of the focused factory as identified by Skinner are:

1. Process technologies. Unproven and uncertain technologies are limited to one per factory. Proven, mature technologies are limited to what their managers can handle.

2. Market Demands. They consist of a set of demands including quality, price, lead times, and reliability specifications.

3. Production Volumes. They are usually of comparable levels, such that material handling techniques, tooling, order quantities, and job contents can be approached with a consistent philosophy. Customer specials and one-of-a-kind orders are to be segregated.

4. Quality levels. A common approach is adopted for setting quality standards and specifications for all of the following attributes: equipment, tooling, training, inspection, job content, material handling system.
5. Manufacturing tools. The goal that each plant has to reach so that it can be competitive is comprised of one set of internally consistent, criteria for success.

The focused factory has many advantages over the complex factory. Repetition and concentration in one area allow its work force and its managers to become effective and experienced in achieving its goal. The focused factory is manageable and controllable.

1.3 The Effects of Reducing Setup Time

The degree of effectiveness of CM is directly related to the amount of required setup time. The hope is that by dedicating cells to produce families of parts, there is a reduction of the time needed to change the tooling on a machine. Since the parts in a family are grouped due to their similarity in terms of manufacturing characteristics, the average amount of time required to change a machine over to working on another batch should be lower when a cell is dedicated to the production of only parts in the family. The degree to which setup times can be minimized determines the effectiveness of the cellular approach. Reducing the average amount of time required for a setup results in multiple benefits such as improved quality control, flexibility and increased effective capacity (see also Schonberger (1982), and Hall (1983)). For example, by spending less time on setups, average flow time and due date performance can be improved.

The capacity of a production facility is often defined relative to its output, the number of units that can be produced per unit time. There are many factors which may effect output capacity:
1. Lot sizes (by changing the number of setups required).

2. Scheduling (minimize the number of required setups).


4. The number of machines and their condition.

5. Quantity and quality of labor.

The impact of all these factors can be investigated using the models that we will discuss. However, we will focus in the lot sizes and scheduling considering the remaining factors fixed.

1.4 Research Goals

Our goal for this research is to develop models that, in manufacturing environments like the ones that are mentioned above, lead to an increase of output capacity. We develop planning and scheduling procedures which are designed to reduce setup times.

The remainder of this thesis is organized as follows. In Chapter 2, we present models that arise from scheduling parts to a single machine. There are $N$ parts to be produced by a single machine, with known tooling requirements. The machine is equipped which has a tool magazine with limited capacity $C$. Because of this, there are times that one or more tools have to be inserted on the tool carousel, so that the machine can process the next part in the sequence. We want to find the sequence in which to process the parts on the machine so that the number of tool switches
is minimized. This is a very hard combinatorial problem. We develop improved procedures for solving it. We show how to find stronger lower bounds. We also develop new heuristic algorithms that sequence the parts on the machine and at the same time load the machine with the proper tools. Computer codes were developed to test the performance of these heuristics.

In Chapters 3 and 4 we examine multiple machine environments. In these chapters we not only construct models but develop computational, mathematical programming based, procedures based on theoretical results and practical considerations. Computer codes are also developed to implement these schemes. One of the contributions of this thesis is the consideration of variability and correlation for the demands among parts when deciding the assignment of parts to machines. In this way, wide swings in machine time demands are avoided. Computationally hard nonlinear integer problems are formulated. We introduce a novel idea that allows us to solve a large class of similar problems by establishing their 1-1 correspondence to a network flow problem.

Finally, Chapter 5 discusses conclusions and recommendations for further research.
Chapter 2

Models Arising from a Single Machine

2.1 Introduction

Recent advances in microelectronics, robotics, and computer integrated manufacturing technology in the metal parts fabrication industries, have given rise to systems so versatile that are able to manufacture a wide variety of parts. Flexible manufacturing systems (FMSs) are groups of Numerically Controlled (NC) Machines that are linked together with an automatic material handling system, and a common hierarchical computer control. These systems are designed to manufacture efficiently parts that are within predetermined families, and which have low to medium production volumes.

One of the most vital requirements in a FMS is the delivery and changing of cutting tools to the CNC machine. Typical CNC machining centers have permanently
attached tool magazines, with capacities often ranging from 30 to 120 cutting tools and are equipped with automatic tool changers.

A well designed tooling system would have all the required cutting tools stored in the magazines of the different machines. But since the total tooling requirements of the parts that are to be fabricated on a machine exceed its tool magazine capacity and because of breakage and cutter wear, tools must be replaced periodically.

One of the most important problems in tool management for flexible machines is how to sequence the parts that are to be produced on a single machine, and with what tools to load the machine so that the total number of tool setups is minimized. We also study other models that appear when the the objective is to minimize the total number of switching instants (i.e. instants at which at least one tool is switched).

2.2 Minimization of the Number of Tool Switches on a Flexible Machine.

We begin by examining the parts sequencing problem on a single machine first formulated by Denardo and Tang (1988) and Bard (1988). Let us now discuss the structure of this problem.

Suppose that N parts are to be processed on an NC machine with an automatic tool interchange device. Each part requires a subset of tools that have to be placed in the tool magazine so that the part can be processed. In total there are M tools needed to produce these N parts. To describe the tooling requirements, a tool part matrix A with dimension M x N is defined, whose entries are
\[ a_{ti} = \begin{cases} 
1, & \text{if part } i \text{ requires tool } t, \\
0, & \text{otherwise.} 
\end{cases} \]

The tool magazine has a limited capacity of \( C \) tool slots. We assume that no part requires more than \( C \) tools for its processing. Tools are transported from the tool storage area and loaded onto the tool magazine when needed. Since the number of tools required to produce all the parts is generally greater than the magazine capacity (\( M > C \)), there are times that is necessary to switch tools when a new part is to be processed. A \textit{switch} is the combination of a tool setup and a tool removal. The setup time in this case, being the time to first load and then calibrate the tools, is proportional to the number of tool switches. The objective is to sequence the parts so that the total time of completion for all parts is minimized. In the scheduling literature this is called \textit{makespan}.

Inherent in such an environment there are the following assumptions:

1. The batch sizes are small.

There are many benefits to small batch sizes. These include flexibility, improved quality, shorter planning horizons, and shorter customer response times. One should note that when batch sizes grow the role of \textit{tool wear} becomes more important. In the models that we are considering, part variety is the dominant factor in the planning decisions. Tool wear is of secondary concern and we assume it does not affect the decision making. Based on a simulation of a particular FMS, Carrie and Pereira found out that in their system tools had
to be changed, on average, ten times more frequently because of tool wear than because of part variety. At the same time they caution against generalizations, pointing out that the individual characteristics of the FMS have to be studied carefully.

2. Tools are changed one at a time.

More that one tool may be setup when changing from the production of one part to the production of its successor. However, when doing this each tool is changed sequentially.

3. Times to change tools are identical for all tools and are constant. The tasks of inserting a new tool in the magazine and withdrawing an old one, would be independent of the specific tool.

4. Each tool occupies one slot in the magazine.

2.3 Formulation of the Problem.

Recall that the objective is to determine the job sequence and the corresponding tool loading that minimizes the number of tool switches, or equivalently, that minimizes the makespan. Let

\[
dx_{jn} = \begin{cases} 
1, & \text{if part } j \text{ is the } n\text{-th part type produced in the sequence,} \\
0, & \text{otherwise,}
\end{cases}
\]

\[
y_{in} = \begin{cases} 
1, & \text{if tool } i \text{ is loaded for the } n\text{-th part in the sequence,} \\
0, & \text{otherwise,}
\end{cases}
\]
\( z_{in} = \begin{cases} 
1, & \text{if tool } i \text{ is inserted in the magazine just before the } n\text{th job is processed.} \\
0, & \text{otherwise,} 
\end{cases} \)

\( i = 1, \ldots, M; \ j = 1, \ldots, N. \)

Then the overall problem can be formulated as the following nonlinear integer problem; which we denote by DT:

(DT):

\[
\text{minimize } \sum_{n=1}^{N} \sum_{i=1}^{M} z_{in} \tag{2.1}
\]

subject to:

\[
\sum_{n=1}^{N} x_{jn} = 1, \ j = 1, \ldots, N \tag{2.2}
\]

\[
\sum_{j=1}^{N} x_{jn} = 1, \ n = 1, \ldots, N \tag{2.3}
\]

\[
a_{ij} x_{jn} \leq y_{in}, \ i = 1, \ldots, M; \ j, n = 1, \ldots, N. \tag{2.4}
\]

\[
\sum_{i=1}^{M} y_{in} \leq C, n = 1, \ldots, N. \tag{2.5}
\]

\[
z_{in} \geq y_{in} - y_{i,n-1}, \ i = 1, \ldots, M, \ j = 1, \ldots, N. \tag{2.6}
\]

\[
x_{jn} \in \{0,1\}, \ j, n = 1, \ldots, N. \tag{2.7}
\]

\[
y_{in}, z_{in} \in \{0,1\}, \ i = 1, \ldots, M, \ n = 1, \ldots, M. \tag{2.8}
\]

Where \( y_{io}, i = 1, \ldots, M, \) are defined by the initial conditions, which are given.

Constraints (2.2) and (2.3) ensure that each part is assigned to exactly one position in the sequence, while constraint (2.5) ensures that no more than \( C \) tools
can be placed on the machine at any instant. Constraint (2.4) assures that if part \( j \) is \( n \)-th in the sequence, then all the tools required by part \( j \) must be on the machine (that is, the time after processing the \( j \)-th part, but before any tools are switched). Constraints (2.6) indicate when tool switches are made.

The objective function counts the number of tool switches. In the next section we will look at different characteristics and at the complexity of the optimization problem (DT).

2.4 Characteristics of the Problem.

2.4.1 Computational Results - Complexity.

Denardo and Tang consider an example with 10 parts \((N=10)\) to be processed on a machine with tool magazine capacity 4 \((C=4)\). These parts require for their processing, a total number of 9 tools \((M=9)\). The tool requirements matrix is given in Table 1 of Denardo and Tang. In this case, the integer program has 190 0-1 variables \((x_{jn} \text{ and } y_{in})\) and 90 continuous variables \((z_{in})\). It took 8 minutes for MPSX/MIP (a standard mixed integer code run on an IBM 370) to produce just 2 feasible solutions. No better solutions were found when the package run for 4 more minutes.

This example indicates the difficulty associated with trying to solve this problem.

Denardo and Tang (1988) observed that the problem can be modelled as a Traveling Salesman Problem with variable edge lengths. Based on this observation they infer that the tool switching problem is NP-hard. Crama et al. (1990) offer a formal proof of this assertion. From the above it is obvious that we cannot hope to find a
polynomial time algorithm, as is actually the case for almost all of the scheduling problems. Therefore some heuristic or approximation scheme must be used.

2.4.2 Problem Reduction.

Denardo and Tang also point out that the size of the problem can be substantially decreased by preprocessing the data. Notice that if part \( j \) requires just a subset of the tools that part \( i \) requires, then by sequencing part \( j \) after part \( k \) entails no additional tool switches. If \( \Delta(k) = \{ j : a_{tj} \leq a_{tk}, \ t = 1, \ldots, M \} \), then one can treat \( k \cup \Delta(k) \) as a single part. Similar ideas will be used in heuristics that we will propose later.

2.5 Minimum number of tool switches for a fixed sequence.

The tool switching problem decomposes into two separate subproblems, sequencing and loading tools. The goals of these problems are:

1. **sequencing**: compute an optimal sequence for the parts.

2. **tool loading**: for the given sequence, determine the tools that should be loaded in the tool magazine, at each instant, so that the total number of tool switches is minimized.

Denardo and Tang showed that the tool loading problem can be solved using the so called *Keep Tool Needed Soonest* (KTNS) policy, in \( O(MN) \) operations. A KTNS
policy has the following properties:

1. At any instant, no tool is inserted unless it is required by the next job.

2. If some tools have to be removed from the magazine, so that tools needed by the next job could be inserted, then the tools needed the soonest for a future part are removed last.

Since the KTNS can be solved optimally in only $O(MN)$ operations, the sequencing problem is the one that is difficult to solve. From now on, for each job sequence we define its cost to be the total number of required tool switches. The minimum number of tool switches for the given sequence is given by the KTNS procedure.

2.6 Lower Bounds

Deriving tight lower bounds, especially for a hard combinatorial problems, is important so that the quality of proposed heuristics can be judged. Furthermore, lower bounds are necessary for the development of exact optimization procedures (for example, a branch and bound procedure). In the next three subsections we discuss different lower bound schemes.

2.6.1 Linear Programming Relaxation of DT

In section (2.3) we introduced the formulation (DT) for the job scheduling and tool loading problems. This formulation is quite ineffective in providing us with good lower bounds. Although the linear programming relaxation, $(DT_{LR})$, provides a lower bound on the optimal number of tool switches, this bound is extremely weak.
Let's look at the following allocation for the \((DT_{LR})\). Set \(x_{jn} = 1/N, y_{in} = C/M, \forall i, \forall n, \forall j\). This allocation satisfies both constraints (2.2) and (2.3), since

\[
\sum_{n=1}^{N} x_{jn} = 1, \quad j = 1, \ldots, N,
\]

\[
\sum_{j=1}^{N} x_{jn} = 1, \quad n = 1, \ldots, N.
\]

Constraint (2.5) becomes

\[
\sum_{i=1}^{M} y_{in} = M(C/M) = C.
\]

For this allocation constraints (2.4) would imply that

\[
1/N \leq C/M \Rightarrow M \leq C \cdot N.
\]

The last relationship is true; if \(M > C \cdot N\), then the problem would be infeasible since it would be impossible to load all the required tools. Therefore constraint (2.5) is satisfied also.

We showed that a trivial allocation \(x_{in} = 1/N, \ y_{in} = C/M\) is a feasible one. Notice that in this case, \(y_{in} - y_{i,n-1} = 0, \forall i, n\) and there is no need for tool switches! Hence the optimal value of the objective function of \((DT_{LR})\) is zero.

One way to improve that bound would be to add more valid inequalities to the formulation of \((DT)\). One such inequality is would be:

\[
\sum_{n=1}^{N} z_{in} \geq 1, \quad t = 1, \ldots, M,
\]
where as before,

\[
    z_{in} = \begin{cases} 
        1 & \text{if tool } i \text{ is inserted in the magazine just before the } n\text{th job is processed.} \\
        0 & \text{otherwise.} 
    \end{cases}
\]

The above is true since all the tools have to be inserted in the tool magazine at least once. These constraints are not satisfied by the optimal solution of \((DT_{LR})\).

Although more valid inequalities could be derived, we believe that the most effective approach for improving the lower bound estimate comes from a different formulation for the tool switching problem. We do not discuss this development here, but rather we intend to derive stronger formulations, as part of our future research.

### 2.6.2 Lagrangian Formulation

One other approach to derive lower bounds on the optimal number of tool switches would be lagrangian relaxation. Two constraints that one might consider to relax would be constraints (2.4) and (2.5). Bard (1988) showed that if those two groups of constraints are relaxed, the resulting subproblems are easy to solve. He later uses this approach to derive different heuristics for the tool switching problem. But it is easy to prove that this scheme does not provide us with any improved bound, since the optimal value of the lagrangean dual equals the optimal value of the linear programming relaxation of \((DT)\). Hence the use of lagrangian relaxation does not show any promise for finding a tighter lower bound.
2.6.3 Traveling Salesman Paths

Lower bounds on the optimal number of tool switches can be derived based on an idea suggested by Denardo and Tang (1988). Denote as $V$ the set of all jobs that are to be sequenced on the machine. Consider a graph $G = (V_0, E_0, lb)$, where $V$ is the set of jobs plus a dummy job 0, with no tooling requirements, that is $T_0 = \phi$. Hence $V_0 = V \cup \{0\}$. $E_0$ is the set of all pairs of nodes in $V$. Every edge has length $lb(i,j)$, where $lb(i,j)$ denotes the fewest possible number of tool switches needed if job $j$ follows job $i$ in the job sequence. This is a lower bound on the actual number of tool switches if $i$ precedes $j$ in the sequence. More precisely, we define

$$lb(i,j) = \max(|T_i \cup T_j| - C, 0),$$

where $T_k$ is the set of tools required by job $k$, $k = 1, \ldots, N$.

Then, the length of a shortest traveling salesman tour in $G$, that starts and ends at node 0, is a lower bound on the total number of tool switches. As Crama et al. (1991) report and our experimentation showed, this bound is extremely weak. The reason is the following. $lb(i,j)$ is quite myopic estimate of the number of tool switches required to process parts $i$ and $j$ sequentially, since it does not account for the whole job sequence. However, this is the best available lower bound on this problem provided in the literature. We extend the lower bounding scheme on this problem in the following discussion. First we introduce the following estimate of the tool switches when jobs $i$, $j$, and $k$ are processed sequentially:
\[ \hat{b}(i,j,k) = \max(|T_i \cup T_j \cup T_k| - C, 0). \]

In this case, \( \hat{b}(i,j,k) \) is less myopic since it considers a partial sequence of not just two, but rather three jobs, in order to estimate the number of tool switches when parts \( i, j, k \) are supposed to be processed one after the other. To extract a new lower bound using the \( \hat{b}(i,j,k) \) we have to construct a graph \( \hat{G} \) properly. For this, let \( V_0 = V \cup \{0\} \) as before, where 0 is a dummy job which requires no tools for its processing. We define \( \hat{V}_0 = \{(i,j) \in V_0 \times V_0, i \neq j\} \) as the set of ordered pairs of \( V_0 \) excluding all pairs of the form \( (i,i) \), \( \forall i = 0, \ldots, N \). We also define the set of arcs \( \hat{E} = \{(i,j),(j,l), i \neq l\} \). An arc from node \( (i,j) \) to node \( (j,l) \) that is a member of \( \hat{E} \), represents the partial job sequence \( i \rightarrow j \rightarrow l \). We assign \( \hat{b}(i,j,l) \) to be the length for that arc, that is a lower bound estimate on the number of tool switches in the sequence \( i \rightarrow j \rightarrow l \).

Consider the minimum weight TSP tour in \( \hat{G} \):
\[(i_0,i_1) \rightarrow (i_1,i_2) \rightarrow (i_2,i_3) \ldots \rightarrow (i_N,i_{N+1}), \text{ with } i_0 = i_N = 0. \]
The weight of this tour is
\[ \sum_{a=0}^{N-1} \hat{b}(i_a,i_{a+1},i_{a+2}). \]
Therefore, the number of tool switches needed for for the partial sequence \( i_b \rightarrow i_{b+1} \) are being computed exactly twice, that is in both \( \hat{b}(i_b,i_{b+1},i_{b+2}) \) and \( \hat{b}(i_{b-1},i_b,i_{b+1}) \). Hence, the half length of the tour is a lower bound on the number of tool switches.

We tested our lower bounding scheme on the test problem that Denardo and Tang (1988) use. Denardo and Tang find the minimum weight TSP tours on \( G(V,E,lb) \) and then apply the KTNS procedure to determine the total number of tool switches.
required by each job sequence. By enumeration among the different TSP tours they try to find the optimal number of tool switches. Running our procedure produces a lower bound (LB) and then by applying the KTNS we find an upper bound (UB). On the Denardo and Tang problem $UB = LB$. This proves that our scheme produced the optimal solution outperforming the procedure developed by Denardo and Tang for their example problems.

2.7 Heuristics for Job Scheduling

2.7.1 Heuristics based on $lb(i,j)$

For the solution of the sequencing subproblem different heuristics have been proposed. The quantity $lb(i, j)$, that was introduced in section 2.6.3, is a lower bound on the number of tool switches incurred between processing jobs $i$ and $j$ in any sequence. Observe that if the tool magazine is always fully loaded (that is $|T_k| = C$, $\forall k$), then $lb_{ij}$ is not just an underestimate of the number of tool switches needed when parts $i$ and $j$ are processed sequentially, but rather is equal to the number of tool switches that are required so that the machine will be able to process part $j$ after part $i$.

Every traveling salesman (TS) path of $G$ corresponds to a job sequence for the tool switching problem. In fact, Denardo and Tang (1988) suggest computing a shortest TS path in $G$ to construct a reasonable heuristic for generating a good sequence. Observe that $lb(i, j) = lb(j, i)$ and therefore the cost matrix for the corresponding TSP is symmetric. In the special case that all jobs require the whole capacity of the tool magazine, the tool switching problem becomes equivalent to the traveling salesman problem, over the graph $G$. 
Based on this, Crama et al. (1991) applied standard TSP heuristics to obtain good solutions to the job sequencing problem. They considered the following four procedures for constructing a short TS path in G:

1. Shortest edge heuristic. This is the heuristic proposed by Denardo and Tang (1988). Its complexity is $O(N^2 \log N)$.


3. Farthest Insertion heuristic with all possible starting nodes. Its complexity is $O(N^4)$ (Golden and Stewart 1985, Johnson and Papadimitriou 1985).

4. Branch and Bound algorithm. This is a branch and bound code that solves TS problems to optimality, but on the other hand is extremely expensive computationally; its complexity is exponential in the worst-case (Volgenant and Jonker 1982).

We should note that using $lb(i,j)$ as the cost for the arcs in $G$ has many drawbacks. The above-mentioned heuristics can be quite myopic, since they do not take into account the whole job sequence to estimate the number of tool switches between pairs of jobs. There are instances that $lb(i,j)$ can be a quite poor estimate of the tool switches between jobs $i$ and $j$. For example, let's examine the case for which each job requires $\leq C/2$ tools. Then

$$|T_i| = |T_j| \leq C/2 \Rightarrow |T_i \cup T_j| - C \leq 0 \Rightarrow lb(i,j) = 0.$$
In such a case, any TSP heuristic that uses \( lb(i, j) \) as a cost estimate for the arcs in \( G \), would pick a random job sequence. Because of these undesirable characteristics of \( lb(i, j) \), we propose in the next section a different length for any arc in \( G \).

### 2.7.2 Heuristics based on \( ub(i, j) \)

We will describe another TSP type heuristic. Consider a graph \( F = (V_1, A, ub) \), where \( V_1 \) is the set of jobs, plus a dummy job 0. Job 0 requires no tools for its processing, that is \( T_0 = \emptyset \). The set of arcs \( A \) consists of all pairs of jobs. We assign as length for the arc \((i, j)\) the following quantity:

\[
ub(i, j) = |T_j \setminus T_i|,
\]

where, as before \( T_l \), is the set of tools that are required by job \( l \), \( l = 1, \ldots, N \). We thus define \( ub(i, j) \) to be the number of tools that job \( j \) requires that are not needed for job \( i \), when job \( i \) immediately precedes \( j \). Therefore, \( ub(i, j) \) is an upper bound on the number of tool switches between jobs \( i \) and \( j \). Then, any traveling salesman path that starts and ends at node 0 defines a sequence for the jobs to be processed on the machine. The cost of this sequence is an upper bound on the number of tool switches that the sequence would produce (as calculated by the KTNS procedure).

\( ub(i, j) \) does not exhibit some of the degenerate properties of \( lb(i, j) \). More specifically \( ub(i, j) \) would not be 0 for two jobs that require different tools, while as we showed before, \( lb(i, j) \) does. If \( ub(i, j) = 0 \), then one of the jobs, say \( j \), requires a subset of the tools that job \( i \) requires; in this case a \( ub \) based heuristic would sequence job \( j \) immediately after job \( i \). Then the heuristic does the preprocessing
that Denardo and Tang proposed, as is described in section 2.4.2. In the special case that all parts require a fully loaded tool magazine for their processing, that is when $|T_i| = C$, $\forall i$, then $ub(i, j) = ub(j, i) = lb(i, j)$ and all three quantities are equal to the actual number of tools switches between $i$ and $j$.

We developed computational tools for testing our heuristic. Notice that the cost matrix for the resulting Traveling Salesman Problem is not symmetric, since in general $ub(i, j) \neq ub(j, i)$. Asymmetric TSP is much harder than the symmetric TSP. For our asymmetric case, we used an enumerative type code developed for the asymmetric TSP by Malik et al. (1990).

Our computational experience is that the $ub$ heuristic does better than the $lb$ heuristics in many instances. Their relative performance depends on the structure of the tool requirements matrix. When the average number of tools required by each part is close to the tool magazine capacity $C$, we will call the matrix dense; in the opposite case the tool matrix will be characterized as sparse. Our computational results showed that the $ub$ heuristic does better than the $lb$ heuristics when the tool matrix is quite sparse. The results agree with our intuition for the problem; under such instances the magazine capacity constraint is not tight, and as we showed earlier in this section, $lb(i, j)$ can give very poor estimates for the required number of tool switches between jobs $i$ and $j$.

### 2.8 Clustering of Jobs

As we saw in the previous section the quantities $ub(i, j)$ and $lb(i, j)$ are equal to the actual number of the tool switches required between $i$ and $j$ when $|T_i| = |T_j| = C$. 
In this case, it is of no importance which jobs have been sequenced before job \(i\) and which after job \(j\) and there is no loss of information by using \(ub(i,j)\) or \(lb(i,j)\). Hence, if all the parts that are to be sequenced on the machine require a fully loaded magazine, for every arc that connects jobs \(i\) and \(j\) we can assign as its cost the exact number of tool switches to process \(j\) immediately after \(i\), and then run a TSP heuristic. Furthermore, most of the FMS environments that we encountered, consist of tool magazines with capacity much larger than the average number of tools required by a part. Motivated by this, we propose the following approach:
Define as a cluster of jobs, a set of jobs that can be processed with at most \(C\) tools in total, that is, a set \(F\) is a cluster if,
\[
| \bigcup_{j \in F} T_j | \leq C.
\]
By its definition, \(F\) is a set of jobs that can be processed with no tool changes. Our aim will be to cluster as many jobs as possible to disjoint clusters. We would like these clusters to require a number of tools as close as possible to the magazine capacity, \(|T_F| \approx C\). As we have seen before, in such a case, both \(ub(i,j)\) and \(lb(i,j)\) provide good estimates of the actual number of tool switches from job \(i\) to job \(j\).

We developed an enumerative type computer code which composes sets of parts that require in total no more than \(C\) tools for their processing. Tool commonality among the jobs is exploited. Finally, we choose disjoint sets that partition the set \(V\) of the jobs. We will treat those clusters now as new “aggregate” jobs, and use the \(ub\) based heuristic for sequencing them on the machine.

Furthermore, for a large number of jobs the clustering procedure decreases the
number of jobs considerably and in this case we can apply, in a computationally efficient way, our lower bound scheme that uses the $\hat{b}(i, j, k)$ estimate.

This approach comes in accordance with some key engineering ideas. Parts or jobs that can be processed with the same tools on the machine are grouped into families. We will discuss composition and assignment problems of families to machines in the next two chapters.

The grouping of parts into families is important when we consider the same manufacturing environment, but we use a different performance criterion, the minimization of the total number of instants at which tools are switched. This performance criterion is useful when the switching time is roughly constant and does not depend on the number of the tool switches. In this case, the partition of the jobs into the fewest possible number of families is desirable. We will discuss this model in multi-machine environments in the next chapter.
Chapter 3

Multiple Machines with Setup Times as a Function of the Number of Switching Instances.

3.1 Background for the Problem.

In this chapter we discuss the problem of scheduling a large set of parts on a Flexible Manufacturing System. The objective considered in this chapter is the minimization of the total completion time.

Consider a Flexible Manufacturing System consisting of a bank of parallel identical machines. Each part can be produced on any one of the machines. Each part requires a subset of tools which must be placed in the machine’s tool carousel before the part can be processed. Each machine has a tool magazine with limited capacity $C$, and in general the number of tools needed to produce all the parts exceeds
this capacity. Therefore, it is sometimes necessary to change tools when a machine switches from one type of part to another. A *switching instant* is an instant at which at least one tool must be switched. In this chapter, the performance criterion for the setup minimization is the minimum total number of switching instants. This performance measure is appropriate when there is an automatic tool interchanging device on a machine that can switch a set of tools simultaneously between the tool magazine and the tool storage area.

The problem is to assign parts to machines and to schedule the processing of these parts on the assigned machine. The objective is to minimize makespan, that is the time to process all parts. The workload for each machine is the total processing time that has been allocated to it plus the setup time, which depends on the way the parts which have been assigned to the machine, are grouped together. The makespan for a machine is the total of the workloads for the parts assigned to the machine. The system makespan is the maximum of these values. This system makespan objective incorporates two goals that rival each other: *setup minimization* and *workload balance*. With this makespan objective we trade off between these two conflicting goals.

As was mentioned in the second chapter, in the single machine case, this objective leads to a natural way of clustering the parts: parts would be assigned to the same group if they can be processed with the same tools in the magazine of the machine. Therefore there is a one-to-one correspondence between these part groups and specific tool configurations of machine magazines. These clusters are *families* of parts that were formed based on the parts’ tooling requirements. Denote by $T_j$
the set of tools required by part $j$; a family is defined as a set of parts $S$ for which $| \bigcup_{j \in S} T_j | \leq C$.  

Once the assignment of parts to machines is made, we can cluster the parts assigned to the same machine into different families, exploiting the commonality in tools for the specific parts.

We should note the assumptions under which this clustering model is applicable. The parts that are to be produced by the manufacturing facility have similar geometry, raw material and require similar fixturing. These parts are assumed to have been grouped together in accordance to the partitioning of a plant into subplants, where each subplant is dedicated to produce similar parts. This is the concept of focused factories. Then it is the differences or similarities in tooling that decide the clustering of parts to families.

### 3.2 Formulation of the Problem.

In this section, we formalize the optimization problem that addresses the assignment of parts to machines. The objective is to allocate parts to machines and then to determine the part families so that the total time to complete processing of all the parts is minimized.

We should comment on the nature of setup times in this manufacturing environment. We are aware of FMSs for which the whole tool carousel is automatically changed when the machine is ready to process a different family of parts. In such a case the main task for the setup, that is loading and calibrating the tools in the magazine, is done off line, and the setup time that would affect makespan consists
only of the time to install the new carousel. Similarly, there are automated machining centers that are equipped with an automatic tool interchanging device that can insert more than one tool at the same time in the tool magazine. In cases like these, we can assume that the setup times are sequence independent, and depend only on the family to which the machine will be switched. In such cases, the setup times can be incorporated into the family processing times, and can be eliminated from the formulation of the problem.

The following notation will be used:

\[ i = \text{index for the parts, } 1, \ldots, N, \]

\[ m = \text{index for the machines, } 1, \ldots, M, \]

\[ f = \text{index for families,} \]

\[ p_i = \text{average total processing time for part } i \text{ (average demand over the planning horizon (PH) times the unit processing time)}, \]

\[ s_f = \text{setup time for family } f, \]

\[ Y_{im} = \begin{cases} 1 & \text{if part } i \text{ is assigned to machine } m, \\ 0 & \text{otherwise} \end{cases}, \]

\[ Z_{fm} = \begin{cases} 1 & \text{if family } f \text{ is assigned to machine } m, \\ 0 & \text{otherwise} \end{cases}, \]

\[ X_{if} = \begin{cases} 1 & \text{if part } i \text{ is assigned to family } f, \\ 0 & \text{otherwise} \end{cases}. \]
\[ a_{it} = \begin{cases} 
1 & \text{if part } i \text{ requires tool } t \text{ to be processed}, \\
0 & \text{otherwise}, 
\end{cases} \]

\( I_m \) = set of parts assigned to machine \( m \),

\( F_m \) = set of families assigned to machine \( m \),

\( MS \) = makespan; the total time to complete the processing of all the parts.

With the above-mentioned notation the overall problem can be stated as follows:

\[
\begin{align*}
\text{minimize } & MS \\
\text{subject to :} & \\
\sum_m Y_{im} = & \sum_m \sum_{f \in F_m} X_{if} = 1, \ \forall i & (3.2) \\
\sum_i \sum_t a_{it} X_{if} \leq & C, \ \forall f & (3.3) \\
\sum_i Y_{im} \rho_i + & \sum_f s_f Z_{fm} \leq MS, \ \forall m & (3.4) \\
X_{if} \leq & Z_{fm}, \ i \in I_m & (3.5) \\
Y_{im} \in & \{0,1\}, \ \forall i, \ \forall m & (3.6) \\
Z_{fm} \in & \{0,1\}, \ \forall f, \ \forall m & (3.7) \\
X_{if} \in & \{0,1\}, \ \forall i, \ \forall f & (3.8)
\end{align*}
\]

Let us discuss the constraints. Constraints (3.2) ensure that every part is assigned to only one machine and at the same time to only one family. Constraints (3.3) make sure that a family is a set of parts that does not require more than \( C \) tools. Constraints (3.4) determine the makespan length, while constraints (3.5)
ensure that when part i is assigned to a family, then the family is assigned to the same machine.

3.3 Makespan: Literature Review - Complexity

Minimizing makespan even for parallel identical machines with no setup times has been shown to be NP complete (Ullman, 1976). For this problem the MULTIFIT heuristic developed by Coffman et al. (1978), based on the "first fit decreasing" bin-packing heuristic, gives a schedule with makespan at most 22% greater than that of the optimal schedule.

The LPT (Longest Processing Time first) heuristic (Graham 1969), first assigns the M longest jobs to separate machines. The remaining jobs are assigned in order of decreasing processing times to the machine that would complete the job first, given the previously assigned workload. It is shown to have worst case bounds of 19/12. Lenstra, Shmoys and Tardos (1987) showed that for unrelated machines with no setups, no polynomial algorithm can achieve a worst case ratio of less than 3/2 unless P = NP.

There is also considerable amount of literature in scheduling identical machines with setups. Geoffrion and Graves (1976) studied the problem in the context of sequence dependent changeover costs, and production costs. Their model arises in chemical processes environments. Parker et al. (1977) with objective to minimize total changeover cost, use a Vechicle Routing Heuristic since their model is a Generalized Assignment Problem. Tang (1988) and Wittrock (1990) give heuristics for
minimizing the makespan on parallel unrelated machines that require minor setups between part types of the same family and major setups between part types of different families.

3.4 Hierarchical Approach

As it is discussed in the previous section, there is a polynomial time algorithm even for the simplest makespan problems. The problem that we are addressing is more complicated due to tooling requirements, tool magazine constraints and grouping of the parts into families. We notice, though, that for a given assignment of parts to machines, we know how to cluster parts to families. Once the families are determined the problem is to assign families to machines to minimize makespan.

We will approach these and related problems hierarchically by breaking them into subproblems.

Furthermore, in our decision-making we want to consider the stochastic nature of the demands for the different parts and their effect on the distribution of the assigned workload on each machine. Models in related literature assume a stable and unvarying demand in order to assign jobs to machines. No provision is made for how such systems could accommodate wide fluctuations in demand without the machines suffering from a crucial amount of over- and under-capacity utilization in the dynamic operation of the system. To do this, we have to examine carefully the demand patterns for the parts. In discrete metal parts manufacturing environments, a typical pattern of monthly demand can be expressed by using the Product-Quantity Pareto graph in Figure 3.1. The graph shows that 20% of the parts account
for roughly 80% of the total production volume (that is, the P/Q ratio is 20 : 80). In fact, P/Q ratios of 10:90 or 5:90 are not unusual. It is obvious that the impact of the high volume parts (HVP) on the machine workloads would be higher than the impact of the Low Volume Parts (LVP). Furthermore, the statistical problems involved in estimating variability parameters for the High Volume Parts are much easier to solve.

This leads us to a two stage hierarchical approach. In the first stage, we assign the high volume parts to the machines and in the second stage we assign the low volume ones.

We first allocate the high volume parts to the machines based on the information we have for the correlation among demands for the various parts. At the same time we would consider the tooling requirements for parts that are to be assigned to the same machine. We would be hesitant to assign two parts, even if they are negatively correlated, to the same machine, as long as they require radically different tool sets.

Hence in this stage we determine \( Y_{im} \) for every part \( i \) that has high demand rate. The objective function represents the desire to assign parts to machines so that the dual effects of positive correlation of high demand parts and high setup requirements will be minimized. This assignment problem is formulated as a quadratic assignment problem.

In the second stage we have to assign the remaining low volume parts to the machines, taking into consideration the allocations made in stage one. Given the remaining available capacity following the stage one assignment, the goal is to allocate the remaining parts to machines so that the newly added parts require as few
Figure 3.1: A 20:80 Product/Quantity Pareto Graph
additional tools as possible. This should help minimize setup time requirements.

3.5 Assignment Problem for the High Volume Parts

The allocation of the high volume parts to the machines is based on the correlation of the demands for the various parts. It is undesirable to assign two parts that have high positive correlation to the same machine. If this occurs we would experience wide swings in the need for machine hours from day-to-day or week-to-week. To avoid this, we would like to assign parts that are negatively correlated to the same machine so that the workload for the machine is smoothed throughout the planning horizon. Correlation of demand among the low volume parts does not affect the distribution for the workload of a machine as much.

For each High Volume Part the following data are available at this point: average demand in machining time over the planning horizon, the set of tooling requirements, and a variance - covariance matrix. Therefore, from the diagonal entries of the variance - covariance matrix we can obtain the variance of demand for each part and from the \((\cdot)_{ij}\) entry, where \(i \neq j\), the correlation for the demands between parts \(i\) and \(j\).

In the first stage, we are defining \(W_{ij}\) to be a reward when parts \(i\) and \(j\) are both assigned to the same machine. \(W_{ij}(\cdot)\) should be a function of the correlation for the demands of parts \(i\) and \(j\) weighted by the average processing times for parts \(i\) and \(j\). In this case a candidate would be:
\[ W_{ij} = (\text{Corr}(i, j) - 1)^2 \times (p_i + p_j)/P_{HV}, \]

where \( P_{HV} \) is the total processing time for all the HVPs. The function \( W_{i,j} \) being quadratic, it assigns greater reward the smaller the correlation gets. The higher reward will be assigned when the correlation becomes -1, while there is no reward for correlation +1.

Furthermore, \( W_{ij} \) should also be a function of the different tools that the two parts require. In the case that the tool magazine is much bigger than the number of tools required by each part, then an appropriate function of \( UB(i, j) \) could be used, as is defined in the second chapter. In cases that the tool magazine capacity is closer to the cardinality of the average tooling requirements set, then a function of \( LB(i, j) \) would be appropriate. In certain instances it might be appropriate to use a convex combination of the two.

Finally, \( W(\cdot) \) should take positive values and it should weight both correlation information as well as information about the required tools. This weight would depend on the nature of the particular problem and the relative importance of exploiting tool commonality when assigning parts to machines versus the significance of smoothing the workload on each machine.

If all the HVPs have similar tooling requirements, then correlation is really the key factor in assigning parts to machines. In this case \( W(\cdot) \) should be a decreasing, positive function of correlation ,as is shown in Figure 3.2. Let

\[
Y_{im} = \begin{cases} 
1 & \text{if part } i \text{ is assigned to machine } m, \\
0 & \text{otherwise ,}
\end{cases}
\]
$p_i$ is the average machine time that is required for processing part $i$.

With the above-mentioned notation, we formulate the following part assignment problem:

$$
\max \sum_{m=1}^{M} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} W_{ij} Y_{im} Y_{jm}
$$

subject to:

$$
\sum_{m=1}^{M} Y_{im} = 1, \ i = 1, \ldots, N
$$

$$
\sum_{i=1}^{N} p_i Y_{im} \leq CAP_m, m = 1 \ldots M
$$

$$
Y_{im} \in \{0, 1\}, \ i = 1, \ldots, N, m = 1, \ldots, M.
$$

All parts are to be produced in an interval of planning horizon PH time units (e.g. a week). This is the planning horizon for which the average processing times $p_i$, of the parts are estimated. Each machine may experience non-negligible downtime due to both planned events (e.g. routine maintenance) and unexpected ones such as failures. Since the assignment of parts to machines is unknown at this point, we have no knowledge about the formation of part families for each machine. Since the setup time depends on the families that have been assigned to each machine, we subtract from the machine time an estimate of its setup time. In this case, the setup time would be significantly smaller that then length of the planning horizon. Then denote $CAP_m$ to be the total available time on machine $m$, where $CAP_m < PH$.

The objective is the maximization of the total reward.
Let's examine the problem constraints. Constraints (3.9) make sure that each part is assigned to one machine. Constraints (3.10) are machine capacity constraints.

Ahmadi and Tang (1989) discuss the benefits of representing a quadratic assignment problem as in (HVPA). They suggest that heuristics based on lagrangean relaxation are used to obtain feasible solutions. These solutions are used later for the development of a simulated annealing scheme.

We will propose a different solution method for the problem. We exploit the structure of the problem. The approach is sound theoretically, is intuitively appealing and is computationally tractable.

3.6 A Solution Procedure for HVPA

Recall the problem (HVPA):

\[
\max \sum_{m=1}^{M} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} W_{ij} Y_{im} Y_{jm} \quad (3.13)
\]

Subject to:

\[
\sum_{m=1}^{M} Y_{im} = 1, \, i = 1, \ldots, N \quad (3.14)
\]

\[
\sum_{i=1}^{N} p_i Y_{im} \leq CAP_m, \, m = 1 \ldots M \quad (3.15)
\]

\[Y_{im} \in \{0, 1\}, \, i = 1, \ldots, N, \, m = 1, \ldots, M. \quad (3.16)\]

We are going to relax the assignment constraints (3.14) and the machine capacity constraints (3.15) using as Lagrangean multipliers \(\lambda_i\) and \(\mu_m\) respectively. Then the
relaxed problem can be written as:

\[
L(\lambda, \mu) = \max \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} W_{ij} Y_{im} Y_{jm} - \sum_{i=1}^{N} \sum_{m=1}^{M} (\mu_m p_i + \lambda_i) Y_{im}
\]

s.t.

\[
Y_{im} \in \{0, 1\}, \; i = 1, \ldots, N, \; m = 1, \ldots, M,
\]

where \(\mu_m \geq 0\), \(\forall m\) while \(\lambda_i\) is unrestricted.

But now \(L(\lambda, \mu)\) has the same form as a problem that considered by Rhys (1970) and solved elegantly by Balinski (1970). Balinski showed its equivalence to a maximum flow problem. He established a 1-1 correspondence between feasible solutions of the original problem and the cuts having finite capacity for the equivalent flow network.

Since the model developed by Rhys will be essential in solving the HVPA, and it will also be used later for the assignment of the low Volume parts we will review it in the next section.

### 3.6.1 Optimal Selection Model

Let us consider a capital budgeting problem with the following characteristics:

1. There are \(M\) potential projects, each of which must be undertaken fully or not at all.

2. The undertaking of project \(i\) requires the performance of a known subset \(S_i\) of a set of \(N\) activities. Set \(S_i\) will be called the enabling set for project \(i\). The enabling sets are not mutually exclusive. If an activity belongs to several
enabling sets, then a single performance of the activity will satisfy at the same
time all the projects which require it.

3. Performing activity j results in a fixed cost $c_j$.

4. Undertaking project i results in a profit $b_i$.

The budgeting problem is to select projects that maximize profit.

Let

$$x_j = \begin{cases} 
1 & \text{if activity } j \text{ is performed}, \\
0 & \text{otherwise}, 
\end{cases}$$

$$y_i = \begin{cases} 
1 & \text{if project } i \text{ is undertaken}, \\
0 & \text{otherwise}. 
\end{cases}$$

Then the budgeting problem can be formulated as the following integer program:

$$\max \sum_{i=1}^{M} b_i y_i - \sum_{j=1}^{N} c_j x_j$$

Subject to:

$$y_i \leq x_j, \text{ for } j \in S_i \text{ and } 1 \leq i \leq M.$$ (3.18)

Define a selection to be a collection of subsets from the set of projects, $S$, together
with all the points of the set of the activities, $\Omega$ which belong to this set. The value
of the selection would be then the sum of the profits of the subsets from $S$ minus
the sum of the costs of the points of $\Omega$ in the selection. Then the problem can be restated as that of finding a selection of maximal value.

Balinsky (1970) showed that the problem can be solved directly in graphical or network terms. Consider the directed bipartite network (NETW) whose nodes consist of a source $m$, a sink $n$, and a node for each point $\omega$ of $\Omega$ and for each subset $S_i$ of $S$. The arcs of the network consist of $(m, S_i), S_i \in S$, with capacity $b_i$, $(S_i, \omega), S_i \in S, \omega \in \Omega$, with infinite capacity, and $(\omega, n), \omega \in \Omega$, with capacity $c_\omega$. Recall that a cut is a partition of the set of nodes into two sets, say $M$ and $\bar{M}$, with the source $m \in M$ and the sink $n \in \bar{M}$, and that the value of a cut is the sum of the capacities on arcs $(i,j)$ with $i \in M$ and $j \in \bar{M}$. In Figure (3.2) the network (NETW) is shown, with the corresponding capacities for the arcs.

Lemma 1: There is a one-to-one correspondence between cuts containing no arc of type $(S_i, \omega)$ and selections.

Lemma 2: The minimum cut corresponds to the selection of maximum value.


Therefore, running a maximum flow algorithm that uses a labelling procedure in the network (NETW) would solve the problem since it would maximize the flow and at the same time identify a minimum cut and therefore a selection of maximum value.

3.7 Solving the Lagrangean Dual

For any $\mu_m$ and $\lambda_i$ such that $c_{im} = \mu_mp_i + \lambda_i \geq 0$ ($i = 1, \ldots N; m = 1, \ldots, M$) the Lagrangean function consists of a positive and a negative part, as the one we
Figure 3.2: Optimal Selection Model
just described in the the capital budgeting example in the previous section. So, by a suitable application of Rhys' model, we know that maximizing such a function can be done in polynomial time using a maximum network flow algorithm. Let us describe now the network $G_{X,\mu}$ that is associated with the problem $L(\lambda, \mu)$.

The vertices are the source $s$, the sink $t$, and the two sets of vertices

$$\Phi = \{Y_{im}Y_{jm}, i, j = 1, \ldots, N, m = 1, \ldots, M\} \text{ and}$$

$$\Psi = \{Y_{im}, i = 1, \ldots, N, m = 1, \ldots, M\}.$$

The capacities on the arcs of the network are as follows. Arcs emanating from the source, $(s,Y_{im}Y_{jm})$ have capacity $W_{ij}$, arcs to the sink $(Y_{im}, t)$ have capacity $\mu_{m}p_{i} + \lambda_{i}$, and arcs $(Y_{im}Y_{jm}, Y_{jm})$ along with $(Y_{jm}Y_{jm}, Y_{jm})$, have infinite capacity. In Figure (3.3) we formulate the network, and insert the capacities for the different arcs.

There are several options for solving the dual problem. We chose the heuristic of subgradient optimization because of its record of success in the literature and because of its suitability for multiple side constraints. Surveys of lagrangean relaxation are provided by Fisher, Northup, and Shapiro (1975), Fisher (1981), Shapiro (1979), and Geoffrion (1974).

### 3.7.1 Features of the Lagrangean Algorithm

Let us note some of the fautes that the lagrangean dual shoud exhibit:

1. The arc capacities have to be positive and therefore the quantity $\mu_{m}p_{i} + \lambda_{i}$, has to be maintained positive throughout all the lagrangean iterations.
Figure 3.3: Network for the HVPA problem
2. As the lagrangean multipliers $\lambda_i$ and $\mu_m$ are updated from iteration to iteration, the relaxed problem's objective is an upper bound on the objective of the unrelaxed problem. The optimal solution for the relaxed problem does not necessarily satisfy the two relaxed constraints. A greedy heuristic that provides a feasible solution gives us a lower bound at the same time.

3. After any iteration, the greatest lower bound ($R^{LB}$) and the least upper bound ($R^{UB}$) over all iterations thus far, allow us to compute a percentage error

$$\frac{(R^{UB} - R^{LB})}{R^{LB}} \times 100\%$$

The heuristic terminates the first time one of the following occurs:

- The percentage error is less than 5%
- The percentage error has not decreased for a prespecified number of iterations.
- The total number of iterations exceeds a prespecified number.

### 3.8 Assignment Problem for the Low Volume Parts

By assigning the high volume parts to machines, we also establish a superset of tools assigned to each machine. We will call these tools the seed tools for every machine. Our goal is to allocate the low volume parts based on the composition of the seed tools and on the effective remaining capacity of each machine. Our objective will be to assign all remaining parts to machines by adding as few new tools as possible to each machine, and without exceeding the machine capacity constraints.
To formulate the problem we will use the following notation:

\[
U_{tm} = \begin{cases} 
1 & \text{if tool } t \text{ is added to machine } m, \\
0 & \text{otherwise,}
\end{cases}
\]

\[
Z_{im} = \begin{cases} 
1 & \text{if part } i \text{ is added to machine } m, \\
0 & \text{otherwise.}
\end{cases}
\]

For every machine, we will call \( S_m \) the set of tools that have been assigned to it, up to that stage. Initially this consists of the tools assigned from the solution to the stage one problem. For every part we denote by \( T_i \) the set of its tooling requirements. Then the overall problem can be formulated as the following integer program; which we denote by LVPA:

\[
\text{(LVPA):} \quad \min \sum_m \sum_{t \in S_m} U_{tm} \quad \quad (3.19)
\]

s.t:

\[
U_{tm} \geq Z_{im}, \ t \notin S_m, \ t \in T_i \quad \quad (3.20)
\]

\[
\sum_m Z_{im} = 1, \text{ for every } i. \quad \quad (3.21)
\]

\[
\sum_i p_i Z_{im} \leq CAP_m \quad \quad (3.22)
\]

Constraints (3.21) force every low volume part to be assigned exactly to one machine, while constraints (3.22) are machine capacity constraints. Recall from the previous sections that certain high volume parts along with the tools that they
require, have been assigned to each machine. Constraints (3.20) via the objective function, count the number of new the tools that have to be added to the machine, because of the low volume parts that are assigned to it. The objective is to minimize the total number of the new tools.

In the next section we develop a solution scheme for this problem, that appears in the second stage of our hierarchical decision-making process.

3.9 A Solution Procedure for LVPA

3.9.1 Introduction

Problem LVPA has two objectives while it assigns low volume parts to machines. Primarily it strives to minimize the makespan and secondarily to add as few new tools to machines as possible while exploiting tool commonality of the parts.

Let us look at the makespan objective first. For this problem we are given a set of N jobs with integral processing times $p_i$ to be scheduled on M identical machines. As we mentioned before, the minimum makespan problem is NP-complete; therefore it is extremely unlikely that there exist efficient algorithms to find a schedule that achieves the optimal makespan. We will denote the optimal value of the makespan, for a given instance of processing times and number of machines, by $OPT_{MS}$. Because of the complexity of the problem, it is natural to consider algorithms that are guaranteed to produce solutions close to the optimum. Polynomial-time algorithms that produce solutions that are at most $(1 + \epsilon)$ of the optimal value are called $\epsilon$-approximation algorithms. Minimizing makespan is one of the problems that have been studied the most in the theory of approximation algorithms for NP-hard problems.
The first class of algorithms that have been proposed for the minimum makespan problem is the class of list processing algorithms. According to this class of algorithms, the jobs are ordered in a list, and the next job on the list is assigned to the next machine that will become idle. Graham (1966), showed that any such algorithm gives a schedule that has makespan at most \((2 - 1/m)OPT_{MS}\). Graham again (1969), showed that if the jobs are ordered with the Longest Processing Time rule (LPT), then the produced schedule has makespan at most \((4/3 - 1/3m)OPT_{MS}\).

A closely related problem is the bin-packing problem: there are \(N\) pieces of size \(p_i\), with \(p_i \in [0, 1]\). The objective is to pack the pieces into bins, under the constraint that the sum of the pieces packed to a specific bin would not exceed 1, so that the number of bins used is minimized.

Coffman et al. (1978) exploited the relationship between these two problems deriving their MULTIFIT algorithm for the minimum makespan problem. The MULTIFIT algorithm comes as an extension of the FIRST FIT DECREASING bin-packing problem. It is proved that it provides a schedule with makespan at most \(1.22OPT_{MS}\). MULTIFIT-based algorithms have the best known bounds, among algorithms that are polynomial in the length of the input.

We will use a MULTIFIT-based algorithm with additional considerations to assign the low volume parts to machines.

### 3.9.2 MULTIFIT Algorithm

Our MULTIFIT algorithm uses a binary search on the makespan. Initially, upper and lower bounds on the makespan are computed. Then, at each iteration, the
mean of the two bounds is used as a candidate makespan, $MS$. Then an allocation algorithm (ALLOCATE) tries to compute a feasible allocation for $MS$; that is an allocation for which all jobs are completed before $MS$. If a feasible allocation is achieved, then the upper bound ($MS_{UB}$) is set to $MS$. Otherwise, the lower bound ($MS_{LB}$) is set to $MS + 1$. The search is terminated when the two bounds coincide.

The initial lower bound will be set to zero. We will use as an upper bound the maximum time capacity for each machine. This could be the length of the planning horizon (PH) we are looking at (for example, this could be a week or a month). This is the length of the time for which the processing times $p_i$'s, have been estimated. The makespan we are looking for, is such that $MS \leq PH$. Our goal is to assign the set of the low volume parts to $M$ machines. There is a bound $k$ on the desired number of iterations. Then the MULTIFIT algorithm proceeds as follows:

1. Set $MS_{LB} \leftarrow 0$;
   
   $MS_{UB} \leftarrow PH$;
   
   $I \leftarrow 1$;

2. If $I > k$, halt.
   
   Otherwise, set $MS \leftarrow [MS_{UB}(I - 1) + MS_{LB}(I - 1)]/2$.

3. If ALLOCATE assigns all parts then, set $MS_{UB}(I) \leftarrow MS$;
   
   $MS_{LB}(I) \leftarrow MS_{LB}(I - 1)$;
   
   $I \leftarrow I + 1$;
   
   and go to 2.

4. If ALLOCATE cannot assign all the parts, set
\[ MS_{LB}(I) \leftarrow MS ; \]
\[ MS_{UB}(I) \leftarrow MS_{UB}(I - 1); \]
\[ I \leftarrow I + 1; \]

and go to 2.

3.9.3 Algorithm ALLOCATE

We now construct the procedure ALLOCATE, which at every iteration of the MULTIFIT algorithm tries to allocate parts (among the low volume ones), to machines, given a candidate makespan \( MS \). The objective is to assign as many parts as possible to each machine, adding as few new tools and satisfying the capacity determined by the current makespan estimate, \( MS \).

Recall that high volume parts have already been assigned to machines. The workload that is incurred from this assignment differs from machine to machine, and therefore for the current \( MS \) and the remaining capacity \( CAP_m \) is different for each machine. ALLOCATE would go through all the machines sequentially, in decreasing order of assigned workload, and allocate parts given the \( CAP_m, \forall m \).

We are going to use the following notation:

\[
X_i = \begin{cases} 
1 & \text{if part } i \text{ is chosen to be assigned,} \\
0 & \text{otherwise,}
\end{cases}
\]

\[
R_t = \begin{cases} 
1 & \text{if tool } t \text{ is chosen to be assigned to the machine,} \\
0 & \text{otherwise.}
\end{cases}
\]
\( \xi_t = \begin{cases} 
0 & \text{if tool } t \text{ belongs to the set of the seed tools}, \\
1 & \text{otherwise}. 
\end{cases} \)

\( T_i \) is the set of tooling requirements for part \( i \).

At every step of the MULTIFIT algorithm, ALLOCATE is called. ALLOCATE would go through all the machines sequentially, and allocate parts given the remaining capacities \( CAP_m \)'s that result from the current \( MS \). After every assignment of a part to a machine, a list that includes all unassigned parts is updated, along with the remaining capacity for the machine.

For a specific machine the problem can be formulated as the following integer program:

\[
\text{max } \sum_{i} X_i \\
\text{s.t:} \\
X_i \leq R_t, \text{ if } t \in T_i, \\
\sum_{t} \xi_t R_t \leq M, \\
\sum_{i} p_i X_i \leq CAP_m. 
\]

Constraints (3.24) ensure that if part \( i \) is picked then all tools that it requires have to be picked too. Constraint (3.26) is the machine capacity constraint. Constraint (3.25) imposes a constraint on the total number of new tools that are assigned to the machine, where \( M \) is just a parameter. A tool will be characterized as new, if it
is not one of the tools (seed tools) that have been allocated to the machine already as a result of the assignment of the high volume parts, done in the first stage of our optimization procedure. Through the variable $\xi_t$ only the allocation of new tools is considered.

The above optimization problem has a special structure, which we will try to exploit. We relax constraints (3.24) and (3.25) using lagrangean multipliers $\alpha$ and $\beta$ respectively.

Then the lagrangean dual is:

$$L(\alpha, \beta) = \max \sum_i (1 - \beta p_i) X_i - \alpha \sum_t \xi_t R_t$$

s.t.

$$X_i \leq R_t, \text{ if } t \in T_i.$$ 

But now $L_{\alpha, \beta}$ has the same form as the optimal selection problem that was reviewed in section (3.6.1), and used in the allocation of the high volume parts to machines. Observe that the lagrangean function consists again of a positive and a negative part, as the objective function of the optimal selection problem does. Therefore, we know that by applying Rhys’ model, we can maximize such a function in polynomial time using a maximum network flow algorithm. Let’s describe now the network $\Theta_{\alpha, \beta}$ that is associated with the problem $L_{\alpha, \beta}$.

The network $\Theta(m)_{\alpha, \beta}$ is defined as follows: Its vertices are the source $s$, the sink $r$, and the two sets of vertices $\Phi = \{i : i$ is an unassigned low volume part $\}$ and $\Psi = \{t : a_{ti} = 1, \ i \in \Phi\}$, that is, the set of tools that are required by the unassigned low volume parts.
The capacities on the arcs of the network are determined as follows. Arcs emanating from the source, \((s,i)\) have capacity \(b_i = (1 - \beta p_i)\), arcs leading to the sink, \((t,r)\) have capacity \(\alpha\), and arcs going from set \(\Phi\) to \(\Psi\), \((i,t)\) for \(t \in T_i\), have infinite capacity. In Figure (3.4) we formulate the network. On every arc we assign its capacity.

We will use subgradient optimization to solve the dual problem. Throughout the subgradient procedure, we have to maintain the quantity \((1 - \beta p_i)\) positive. This a requirement so that the dual problem can be modelled as a network flow problem.

Recall that at each iteration of the MULTIFIT algorithm we assign parts to each machine and then proceed to the next machine sequentially. For every machine \(m\), with the lagrange multipliers \(\alpha\) and \(\beta\) known, we solve a maximum flow algorithm on the network \(\Theta(m)_{\alpha,\beta}\). This algorithm will also identify the minimum cut. According to Lemma 2 of section 3.6.1, the minimum cut identifies the parts along with their tools, that are to be chosen to be assigned to the machine; this selection is the optimal one, that is, it maximizes \(L_{\alpha,\beta}\).

The parts that are chosen to be assigned to machine \(m\) are then withdrawn from the set \(\Phi\) of the low volume parts that are still to be assigned. At the same time the set \(\Psi\) that includes the corresponding tools for the remaining parts is updated, too. These updated sets are then used for the formation of \(\Theta(m+1)_{\alpha,\beta}\) at the next step of the algorithm (the assignment of parts to the next machine).

To reduce the computational effort involved in assigning the LVPs we propose the following procedure. As it can been seen in Figure (3.1) the majority of the LVPs (approximately 50% of the total number of parts) have marginal effect on
Figure 3.4: Network for the LVPA problem
the cumulative demand. Therefore we partition the set of LVPs into two subsets. The first consists of the LVPs with the relatively highest contribution to the total cumulative demand (approximately 30% of the parts), and the second with the LVPs that contribute the least to the total demand. For the first subset of parts the MULTIFIT algorithm along with the procedure ALLOCATE will be used, as is described above. For the remaining ones a simpler procedure, rather than ALLOCATE, can be used at each iteration of the MULTIFIT algorithm. This procedure would be a FIRST FIT DECREASING heuristic with the extra requirement that the tooling constraint (3.25) should be satisfied also, every time a part is allocated to a machine. If this is not the case then the heuristic would try to allocate the part to the next machine on the list, that is the one with the immediately higher assigned workload.

This concludes the assignment of the LVPs to machines. To summarize, at the first stage of the hierarchy we assigned the high volume parts to machines based on the correlation among the parts and on similarities on tooling. With the HVPs and their corresponding tools assigned, we proceed to the next stage, assigning the LVPs to machines so that as few new tools as possible are added to each machine.
Chapter 4

Multiple Machines with Setup Times as a Function of the Number of Tool Switches.

4.1 Background for the Problem

This chapter discusses the problem of scheduling a large set of parts on a FMS so as to minimize the total completion time. The nature of the problem is quite different though from the problem addressed in the previous chapter, because we consider a different performance criterion.

We assume that the FMS consists of a set of parallel identical machines. Each part can be processed by any one of the machines in this group. Each part requires a subset of available tools. The tools required to manufacture the part must be placed in the machine’s tool magazine prior to the part being processed. Each machine has
a tool magazine with a limited capacity $C$, and in general the number of tools needed to produce all the parts exceeds this capacity. Therefore, it is sometimes necessary to change tools when a machine switches from one type of part to another, that is, a setup time is incurred. This setup time is a function of the number of tool switches, and is the time that is required to switch (insert and remove) the tools on the carousel, calibrate them and accommodate the proper fixtures.

When the assignment of parts to machines is known, then the problem reduces to the single machine scheduling problem whose objective is to minimize the number of the tool switches, as discussed in Chapter 2.

4.2 Formulation of the Problem

In this section, we formalize an optimization problem addressing the assignment of parts to machines and then their sequencing. The objective of the problem is to allocate parts to machines and then to determine the order of processing the parts on the corresponding machines, so that the total time to complete the processing of all parts is minimized. To achieve this, we have to consider the setup times that are functions of the number of tool switches that are required to process all the parts that have been assigned to an individual machine.

The following notation is required to formulate the problem:

$i = \text{index for the parts, } 1, \ldots, N.$

$p_i = \text{average total processing time for part } i \text{ (average demand over the planning period times the unit processing time).}$

$m = \text{index for the machines, } 1, \ldots, M.$
\[ \sigma_A \] is a sequence of parts that are members of the set A.

\[ SU(\sigma_A) \] is the setup time that is associated with processing the parts that are included in set A with the sequence \( \sigma \). Let

\[
Y_{im} = \begin{cases} 
1 & \text{if part } i \text{ is assigned to machine } m, \\
0 & \text{otherwise ,}
\end{cases}
\]

\( I_m \) is the set of parts assigned to machine \( m \), and

\( MS \) is the total time to complete the processing of all the parts.

Then the problem is stated as follows:

(PAM):

\[
\begin{align*}
\text{minimize } & MS \\
\text{subject to :} & \\
\sum_{m=1}^{M} Y_{im} & = 1, \ i = 1, \ldots, N \\
\sum_{i=1}^{N} (Y_{im}p_i) + SU(\sigma_{I_m}) & \leq MS, \\
Y_{im} & \in \{0, 1\}, \ i = 1, \ldots, N, m = 1, \ldots, M.
\end{align*}
\]

4.3 Hierarchical Approach

As it is mentioned in the previous chapter, there is no hope to obtain a polynomial time algorithm even for the simplest makespan problem. It is clear that the problem we are addressing is more complicated, since even the sequencing of parts to a single machine is an extremely hard combinatorial problem, as shown in Chapter 2. We
will therefore approach the problem (PAM) and related problems hierarchically. The structure of the hierarchy is the same with that of Chapter 3, while the different nature of the setup times is considered.

Furthermore, in our decision-making we want to consider the stochastic nature of the demands for the parts and their effect on the distribution of the workload that is assigned to each machine. Since the manufacturing environment remains the same with the one described in Chapter 3, we still distinguish the parts into high volume parts (HVPs) and low volume parts (LVPs). The HVPs have stronger impact on the machine workloads than the LVPs. We will use a two stage procedure. In the first stage, we assign the high volume parts to the machines and in the second stage we assign the low volume ones.

4.4 Assignment of the High Volume Parts

We first allocate the high volume parts to the machines based on the information we have for the correlation among demands for the various parts. As in the model studied in Chapter 3, it is undesirable to assign two parts that have high positive correlation to the same machine since that would lead to wide swings in the need for machine hours from day-to-day or from week-to-week. At the same time correlation cannot be the sole criterion for assigning HVPs to machines. As it is explained in section 3.5 we should also consider the difference in tool requirements among the parts.

At this stage we determine $Y_{im}$ for every part $i$ that has high demand rate. This is accomplished by solving a series of problems. To begin with, we approximate the
exact setup time $SU$. In reality, the setup time depends on the order in which the parts are processed. Given this assignment of parts to machines, which is based on the approximation, we then solve the problem of sequencing the parts to run on each machine. The techniques developed in the previous chapter are used to solve this sequencing problem. This sequence provides a better estimate of the setup time.

To find the exact number of tool switches that results from the above sequence the KTNS procedure of Chapter 2 is used. Using this more accurate estimate we repeat the assignment step.

This assignment problem of parts to machines is formulated as a quadratic assignment problem. The objective function represents the desire to assign parts to machines so that the effect of positive correlation of high demand parts on meeting customer demand will be minimized. We will use the same notation with the the notation used in Chapter 3. $W_{ij}$ would be the reward if parts $i$ and $j$ are assigned to the same machine. Then the part assignment problem would be:

$$\max \sum_{m=1}^{M} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} W_{ij} Y_{im} Y_{jm}$$

(4.5)

Subject to:

$$\sum_{m=1}^{M} Y_{im} = 1, \; i = 1, \ldots, N$$

(4.6)

$$\sum_{i=1}^{N} p_i Y_{im} + SU \leq CAP_m, \; m = 1 \ldots M$$

(4.7)

$$Y_{im} \in \{0,1\}, \; i = 1, \ldots, N, \; m = 1, \ldots, M.$$ 

(4.8)

To solve the quadratic problem we will use the solution procedure that was developed in section 3.6, by establishing a 1-1 correspondence between this problem
and a maximum flow network. By identifying the minimum cuts on the network, the best \textit{selection} is given, that is the assignment of parts to machines that maximizes the objective function (4.5).

4.5 Assignment of the Low Volume Parts

In the second stage we have to assign the remaining low volume parts to the machines, taking into consideration the allocations made in stage one. Given the remaining available capacity following the stage one assignment, the goal is to allocate the remaining parts to machines so that the newly added parts require as few additional tools as possible. This should help minimize setup time requirements.

We will use the same notation as in chapter 3.

\[
U_{im} = \begin{cases} 
1 & \text{if tool } t \text{ is added to machine } m \\
0 & \text{otherwise}
\end{cases}
\]

\[
Z_{im} = \begin{cases} 
1 & \text{if part } i \text{ is added to machine } m \\
0 & \text{otherwise}
\end{cases}
\]

\(SU_m\) is the setup time that depends on the order in which the parts that are assigned to the machine are processed.

For every machine, we will call \(S_m\) the set of tools that have been assigned to it, up to that stage. Initially this consists of the tools assigned from the solution to the stage one problem. For every part we denote by \(T_i\) the set of its tooling requirements. Then we want to:
\[ \min \sum_{m} \sum_{t \in S_m} U_{tm} \quad (4.9) \]

s.t:

\[ U_{tm} \geq Z_{im}, \ t \notin S_m, \ t \in T_i \quad (4.10) \]

\[ \sum_{m} Z_{im} = 1, \text{ for every } i. \quad (4.11) \]

\[ \sum_{i} p_i Z_{im} + S U_m \leq C A P_m \quad (4.12) \]

To allocate the LVPs we propose the following scheme. Start with an approximation of the set up time for each machine. In reality, the setup time depends on the order in which the parts are processed. Now use the MULTIFIT algorithm described in detail in section 3.9 to get an assignment of all the LVPs to machines. Since MULTIFIT is quite expensive computationally, we will not include it in an iterative scheme. With the assigned LVPs, find the machine among the \( M \) machines with the heaviest workload, that is the "bottleneck" one; this is the one that determines the makespan. Find now the part assigned to the bottleneck that by reassigning it to another machine, would decrease the makespan the most. For every part an estimate of the savings on makespan can be calculated: \( SAV(i) = p_i + \Delta SU \), where \( \Delta SU \) is a rough estimate on the decrease on setup time incurred by taking part \( i \) out of the ones assiged to the machine. This can be done by running the KTNS procedure with part \( i \) in the sequence of the parts that are to be processed on the machine, and then running the KTNS on the same sequence but excluding now part \( i \). The difference of the two gives an estimate of the number of tool switches that can be "saved". Since KTNS takes \( O(MN) \) the process of finding the part
with the maximum savings would take $O(MN^2)$. If computational time is not an issue, a TSP heuristic could be run. Then the number of the tool switches incurred by the TSP tour can calculated through the KTNS procedure.

The part $i$ that was taken out of the set of the parts assigned to the bottleneck, will be assigned to the machine with the smallest resulting workload, as long as the new makespan is less than the makespan before the swap of the parts. The procedure will be repeated until there is no swap that would improve the makespan.
Chapter 5

Conclusions - Further Research

5.1 Summary

In this thesis we address an approach for machining batches of parts on a group of machines considering the effects on machine loads and tool assignments. This allocation is related to principles conveyed by Group Technology and Cellular Manufacturing. In environments like these the reduction of setup time is crucial for increasing the effective capacity of the system.

In the Chapter 2, we formulate models and develop computational tools for the sequencing of parts on a single machine. The tool requirements for every part are known, and the machine has a tool magazine with limited capacity. We want to sequence the parts in an effective way so that the number of tool switches to process all parts is minimized. The problem is an extremely hard combinatorial problem. We propose a scheme that provides us with lower bounds that are tighter than the ones that already exist in the literature. Furthermore, we developed heuristic algorithms
that perform the sequencing of the parts on the machine and the corresponding loading of the tools on the machine. Different algorithms were developed for the case that the objective criterion is not to minimize the number of tool switches but rather to minimize the number of times that tools have to be switched. This criterion can be applicable in certain manufacturing environments.

In Chapters 3 and 4, we look at problems that appear in flexible multi-machine systems. We address issues that appear at the lower, operational level of the hierarchy. We have decided which parts are to be produced by the manufacturing facility. The parts are quite similar in terms of geometry, raw material and required fixturing. Then, we will compose parts into families based on their tooling characteristics. The problem is to assign parts to machines, cluster parts to families, and then to determine the sequence in which the parts will be produced. Our decision-making spans a planning horizon (a week, a month), for which we estimate the average demand in machining time for all the parts. The above-mentioned problems are very complex. Decisions are made sequentially and their impacts are interwoven. We develop an approach that considers the complexity of the problem as well as the sequential nature of the decision-making process.

Most of the models in literature assume a stable demand in order to assign parts to machines without considering the dynamic operation of the system. Wide fluctuations in demand in machining time would cause the machines to be under- and over-utilized at certain time periods. One of the major contributions of this thesis is the fact that we account for the variability of demands on the distribution of the workload that has been assigned on a machine. We develop an hierarchical
scheme that assigns parts to machines while it considers both the variability of demands for the different parts along with tool commonality. In the first level of the hierarchy, we assign parts that have high demand over the planning horizon. For this, a quadratic integer program is developed. This problem while being very important in the job grouping literature, is well known for its complexity. We solve it in a novel way, by establishing a 1-1 correspondence to a maximum flow network problem. Finally, at the second level the low volume parts are assigned using state of the art algorithms for the makespan problem, modified to accommodate tool commonality among the parts and the assignment of the high volume parts that took place in the first level of the hierarchy.

5.2 Areas for Future Research

There are several areas regarding our methodology for which additional research could prove fruitful.

We first intend to find formulations of the tool switching problem for the single machine case, that would give tight lower bounds. We also intend to further explore the performance of the maximum flow algorithm that was developed for the solution of similar quadratic problems.

In our current models we have not set the desired levels of capacity. We shall investigate how randomness in demand affect our decisions for optimal capacity levels. The environment under study would include multiple machines. We would assume that demands for the different parts are correlated, while there is also correlation in the demand for the same part from period to period. By looking at the aggregate
demand of parts in machining time, as a single commodity we can establish the equivalence of this problem with that of a dam problem from the queueing theory literature. One should recall that for every dam model there is a corresponding queueing model. For example, when the demand for the parts is stationary from period to period, but there is correlation among the demand for parts in the same time period, the aggregate demand of the parts can be fit by a k-Erlangian distribution, using its first two moments, and the problem is equivalent to the $E_k/D/1$ queueing model. This would be a rather naive approach to the problem though, since the k-Erlangian distribution could underestimate the variance of the demand. We would be interested in finding the limiting probability distribution of the inventory level. When the system is behind capacity, backlogging is incurred. An important issue is how much safety stock should be built so that the system operates effectively. There is a trade-off among the different levels of capacity. Lower levels of system capacity imply lower investment and higher utilization, but on the other hand, the expected backlogging cost is higher.

When we want to model correlation in demands from period to period, then we have to use tools from Renewal Theory. The correlation from period to period in the first place, can be modelled with an underlying Markov Chain. Depending on the state $i$, of the Markov Chain, the aggregate demand comes from a probability density $\phi_i(x)$.

Using such models we want to investigate the relationship between aggregate capacity and inventory. After the aggregate inventory level is estimated we have to disaggregate it among the different families of parts. This disaggregation scheme is
a topic for future study, too.
Bibliography


AN HIERARCHICAL APPROACH TO MACHINE
BATCHING, LOADING, AND TOOL ALLOCATION
PROBLEMS.

A Dissertation
Presented to the Faculty of the Graduate School
of Cornell University
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

by
Eleftherios Iakovou
January 1992