AN ABSTRACT OF THE THESIS OF

Adulwit Sonthinen for the degree of Master of Science in Industrial Engineering presented on November 21, 1995. Title: An Efficient Methodology for Scheduling General Flexible Manufacturing Systems.

Abstract approved: Redacted for Privacy

Rasaratnam Logendran

Flexible Manufacturing Systems (FMSs) have evolved rapidly over the last decade. One important factor that influences the performance of FMS is scheduling. Scheduling in an FMS differs from that in a conventional job-shop because each operation of a job may be performed by any one of several machines. Most studies of scheduling in FMS in the past have assumed that each part has a unique process plan. However, in an actual manufacturing environment, it is possible for each part to have more than one process plan and each operation required of a part can be performed on alternative machines.

In this thesis, the problem of scheduling parts in a General Flexible Manufacturing System (GFMS) in static environments is investigated when each part can have alternative process plans and each operation required of a part can be performed on alternative machines. The model and a solution algorithm have been developed for the problem. The mathematical model is formulated as the mixed-(binary) integer programming problem, and is proven NP-hard in the strong sense. An implicit-enumerative algorithm such as the branch-and-bound technique could not be used to find the optimal solution within a reasonable time even for a small-sized problem. The heuristic algorithm based on the concept known as tabu search is proposed to efficiently find the optimal/near-optimal solution for large-sized problems within a reasonable computational time.

The steps involved with the heuristic algorithm and its application to an example problem are presented. Six different versions of tabu search-based heuristics (TSH 1-TSH 6) are compared to investigate the impact of using long-term memory and the use of fixed versus variable tabu-list sizes. A statistical experiment, based on randomized-block design, is carefully constructed to test the performance of the heuristics on 4 problem

structures ranging from 4 parts to 14 parts. The results reveal that the tabu search-based heuristic with fixed tabu-list size and long-term memory (TSH 3) is preferred to other heuristics as the problem size increases.

This research substantiates the fact that there is clearly a need for efficient heuristics to solve the complex scheduling problems in FMSs. One cannot rely on the implicit-enumerative procedure such as the branch-and-bound technique to solve problems that have practical significance.

An Efficient Methodology
for Scheduling General Flexible Manufacturing Systems

by

Adulwit Sonthinen

A THESIS

submitted to

Oregon State University

in partial fulfillment of

the requirements for the

degree of

Master of Science

Completed November 21, 1995

Commencement June, 1996

Master of Science thesis of **Adulwit Sonthinen** presented on **November 21, 1995**

APPROVED:

Redacted for Privacy

Major Professor, representing Industrial Engineering

Redacted for Privacy

Chair of Department of Industrial and Manufacturing Engineering

Redacted for Privacy

Dean of Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for Privacy

Adulwit Sonthinen, Author

ACKNOWLEDGMENT

A great many people have made important contributions to this thesis. My sincere thanks go to my Major Professor, Dr. Rasaratnam Logendran who originated this work and advised me continually throughout the pursuit of my research. His invaluable contributions made this work possible. I would like to thank all of my committee members; Dr. Ken Funk, my Minor Professor, Dr. Sabah U. Randhawa, my committee member, and Dr. Roger C. Graham who took time off his busy schedules to serve as my graduate council representative.

I am grateful to the staff of the Department of Industrial and Manufacturing Engineering for the use of the laboratory and office facilities.

My appreciation and thanks to all authors whose works I have cited throughout this thesis. Such articles have added more value to this research.

I would also like to thank all my friends for their support and encouragement. Special thanks go to Ms. Vimalin Puvanunt for her help in C-programming software, Mr. Budhi Saleh for his generous help in statistical analysis, and Mr. Vichien Tangpanyapinit for everything you have done for me.

My deepest gratitude is reserved for my parents. If not for them, I would not have been able to successfully complete my Masters degree at OSU.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

LIST OF TABLES (Continued)

Table **Page**

LIST OF TABLES (Continued)

Table **Page**

LIST OF APPENDIX TABLES

Table **Page**

LIST OF APPENDIX FIGURES

Figure **Page**

An Efficient Methodology For Scheduling General Flexible Manufacturing Systems

1. INTRODUCTION

Manufacturing industries continue to become more complex mostly due to global competition, cost and profitability pressures, and rapidly advancing technology. Competitive forces are driving manufacturing firms to design and implement production systems which are more flexible in terms of product variety and more efficient at the same time (Demeyer et al., 1989). Conventional methods of manufacturing could not respond to the constantly changing customer requirements. The flexibility or the speed at which systems can react to and accommodate these changes is the essential factor for manufacturers to stay competitive. To this end, flexible manufacturing systems (FMSs) can provide the desired flexibility to rapidly respond to market changes through the utilization of automatic machine tools, automatic material handling system and integrated computer controlled system.

FMSs are regarded as one of the recent developments of machine tools and material handling automation for multi-product small-batch production system. An FMS can be defined as a computer controlled manufacturing system using numerically controlled (NC) machines that are linked together with an automatic material handling system to process a variety of parts in random order under control of an integrated central computer. The central computer provides the control of machine tools and workstations, the transfer control of components and tooling as well as the information control. This combination of flexibility and overall control makes possible the production of a wide range of products in small numbers (Luggen, 1991).

The concept of FMSs was introduced by David Williamson in London, England in the early 1960's. At the beginning, it was a computerized machining system called "System 24" that would run for 24 hours per day (16 unmanned on night shift) under the control of a computer. The idea of using computer-controlled machines to perform a variety of different operations was the beginning of FMSs.

The objective of FMS is to achieve the efficiency of automated high-volume mass production while retaining the flexibility of low volume job shop production (Groover and Zimmers, 1984). With the inherent flexibility of NC machines, an ideal FMS can handle a wide variety of dissimilar parts, producing them one at a time, in any order, as needed.

There exist different types of FMSs based on the different criteria such as the machine tools, the type of material handling system used, the process flexibility, the operational mode and the scheduling environment (Jaikumar, 1986). However, two main types of FMSs based on the classification made by Rachamadugu and Stecke (1987) are dedicated flexible manufacturing system (DFMS) and random/general flexible manufacturing system (GFMS). The DFMS is the FMS with small product variety, moderate to large volumes for each part type (i.e., 2,000-200,000/year) and stable demand and product mix. In contrast, the GFMS can produce a very large number of different part types with only small to moderate volume for each part type (i.e., less than 2,000/year) and a changing product mix.

The primary benefits attained by implementing FMSs are the short-term responsiveness to the day-to-day problems on the shop floor such as engineering changes, processing changes and machine unavailability by using NC machines, reduction of inventory by reducing lot sizes, savings on direct labor by removing operators from the machine site, increase in machine utilization by eliminating the setup time, and improvement on operational control by reducing the number of uncontrolled variables. In addition, the long-term advantages include the accommodation through quicker and easier alteration of the system to support the changing product volumes, different part mixes and new product additions (Maleki, 1991 and Luggen, 1991).

Despite the benefits of using FMSs, the operation of FMSs has proven to be difficult in practice. One aspect of FMSs' operations that has been particularly difficult is scheduling. The complications of scheduling in FMSs result from the flexibility, complexity and need for system integration (Hutchison et al., 1991). Scheduling in FMSs differs from that in a conventional job shop because each operation of a job may be performed by any one of several machines. Therefore, the decisions concerning the jobs require not only a sequencing decision but also a routing decision (Chang et al., 1989).

FMSs can be scheduled by a real-time or an off-line scheme. Real-time schemes schedule operation by operation, one at a time in real time or event by event basis, while off-line schemes schedule many operations at one time for the entire shop prior to actual production. Many research investigations have been performed to develop scheduling schemes for FMSs. The reported studies include Stecke and Solberg, 1981, Kimemia and Gershwin, 1983, Sarin and Dar-El, 1984, Shanker and Tzen, 1985, Chang et al., 1985, Denzler and Boe, 1987, Hutchison, 1988, Chang et al., 1989, and Hutchison et al., 1991. Some of these studies are reviewed in detail in the next chapter.

The problem of scheduling of parts in FMSs has been addressed in a random job-shop environment by Hutchison et. al. (1991). The authors compared two off-line schemes with a real-time scheme. The first off-line scheme establishes an overall optimal solution while the second off-line scheme decomposed the problem into a loading subproblem and a resulting scheduling subproblem and finds the optimal solution to both subproblems. The real-time scheme uses the Shortest Processing Time (SPT) dispatching rule with look-ahead control policy to schedule the parts. Two off-line schemes are formulated as a mixed integer, zero-one programming model and solved by using the branch-and-bound technique. The results indicate that both off-line scheduling schemes are significantly better than the real-time scheduling scheme with respect to makespan.

Previous studies in FMS scheduling for the most part have assumed that each part has a unique process plan (i.e., the sequence of operations on machines). However, in an actual manufacturing environment each part can have more than one process plan. To the best of our knowledge, none of the previous studies in FMS scheduling has addressed the issue of having more than one process plan for each part. Nevertheless, Logendran et al. (1994) studied the design problem of a cellular manufacturing system when each part can have more than one process plan. The problem considered by Logendran et al. (1994) is to determine the number of machines of each type and a unique process plan for each part in a manufacturing cell. The authors successfully developed heuristics to solve the problem based on the concept known as tabu search, which is a higher-level heuristic for solving combinatorial optimization problems.

Even though the problem considered by Logendran et al. (1994) is not for a flexible manufacturing system, the approach used to determine a unique process plan for each part is similar to that of the scheduling problem in an FMS because there is a need to determine a unique process plan prior to actually scheduling parts in an FMS.

Accordingly, this research aims at scheduling of parts in an FMS in the presence of alternative process plans. The scheduling environment in this research is static, meaning that the machines in the shop are idle when parts arrive. The machines are assumed to have enough capacity to produce parts. The objective of this research focuses on finding an optimal/near-optimal schedule which gives the minimum makespan. A makespan is the length of time required to complete all parts. Therefore, the problem is to schedule parts in order that they are completed in the shortest or minimum length of time. Other performance criteria such as an average flow time (Chang et al., 1985), weighted machine utilization (Sarin and Dar-El, 1984), and adjusted production rate (Hutchison and Khumawala, 1990), can also be used in the context of scheduling FMSs.

2. LITERATURE REVIEW

In general, FMSs' operational decisions consist of pre-release decisions and post-release decisions. Pre-release decisions, also called the FMS planning problem, consider the pre-arrangement of parts and tools before the FMS begins to process. Post-release decisions, also called the FMS scheduling problem, deal with sequencing and routing parts when the system is in operation (Hwan and Shogan, 1989 and Mukhopadhyay et al., 1992). By the same token, Kusiak (1985) also classified the FMS problems into two groups: (1) design problems and (2) operational problems. The design problems include the selection of part families to be manufactured, the selection of an FMS production system, the selection of a material-handling system, the selection of fixtures and pallets, the selection of an integrated computer system and the layout of systems. The operational problems deal with: (1) planning problem, (2) grouping problem, (3) machine loading (capacity balancing) problem, and (4) scheduling problem. The complexity of these problems depends on whether the FMS is of a dedicated type or a random/general type.

Three different scheduling decisions occur in operating an FMS on the shop-floor level (Denzler and Boe, 1987). These decisions include: (1) Part loading, (2) Part loading timing, and (3) Part routing/sequencing. The part loading decision concerns choosing the part type to enter the FMS when it is time for a new part to enter the system, considering such factors as part characteristics and machine workload. The part loading timing decision concerns when to enter a new part into the system, considering such factors as system congestion and part fixture availability. The part routing or the dispatching decision concerns routing parts through the FMS at the time of actual production, such as sequencing parts at the individual machines in the FMS.

The scheduling aspect of FMSs is still an emerging area of research. There are several features that make scheduling of FMSs difficult. First, it is possible to process many different types of parts on the system at any given time provided the machines are appropriately tooled. Second, changeovers between different types of parts are negligible, so parts were not necessarily processed in batches and each unit can be scheduled separately. Finally, an operation can be performed by more than one machine, so routing

decisions are also required (Kim and Yano, 1994). The routing flexibility is the important characteristic that distinguishes the FMS scheduling from a classic general job-shop problem. According to Rachamadugu and Stecke (1987), the FMS scheduling procedures can also be classified into four important dimensions: (1) operational mode, (2) FMS type, (3) scheduling environment and (4) responsiveness. The classification of FMS scheduling is shown in Table 2.1.

Most research in the past have concentrated on the relative performance of the simple dispatching rules. Stecke and Solberg (1981) reported a simulation study for a dedicated type FMS, comparing simple, commonly used 16 dispatching rules for FMS scheduling. The SPT/TOT dispatching rule (shortest processing time for the operation divided by total processing time for the part) performed the best over all levels of the alternate routing flexibility. Similarly, Shanker and Tzen (1985) studied the loading and dispatching problem in a random type FMS. The research compared two loading algorithms in conjunction with four different dispatching rules: FIFO (first-in-first-out), SPT (shortest processing time), LPT (longest processing time) and MOPR (most operations remaining). The study concluded that the SPT rule performed the best on average.

Researchers have also investigated off-line scheduling schemes. Chang and Sullivan (1984a,b) developed a two-phase method to simultaneously determine job routing and scheduling. In the first phase, a set of candidate schedules is generated separately for each job. Then, in the second phase, the best feasible combination of schedules from the set is found by solving the 0-1 integer programming problem. The performance of using the two-phase method is shown to be superior than the simple dispatching rules commonly used in practice. However, the application of the two-phase method is limited to relatively small FMSs because the second phase requires the solution of a large integer programming model.

Hutchison and Khumawala (1990) investigated a near-optimal scheduling scheme for a random, job-shop FMS within a dynamic environment. The study is divided into two phases: the first phase is to compare two off-line scheduling schemes

Table 2.1 The Classification of FMS Scheduling.

Dimension	Levels
Operational mode	1. Dedicated: small set of part types with moderate volumes (i.e., 2,000-200,000 /year) 2. Random: large set of part types with low volumes (i.e., less than 2,000 /year)
FMS type	1. Flexible Transfer Line: flow shop type of FMS 2. General Flexible Machining Systems (GFMS): job-shop type of FMS 3. Flexible Assembly Systems: FMS that assembles components and subassemblies
Scheduling environment	1. Static: order status changes periodically (e.g. day to day) 2. Dynamic: order status changes continually
Responsiveness	1. Real-time: schedule in an operation by operation or event by event basis 2. Off-line: schedule a complete set of parts at one time off-line, then the FMS manufactures these parts according to this schedule

(source: Rachamadugu and Stecke, 1987)

and the second phase is to compare the off-line scheme with seven real-time scheduling schemes. In the first phase, the first off-line scheme optimally solves the entire scheduling problem while the second off-line scheme decomposed the problem into a loading and a scheduling problem. In the decomposed schemes, the loading subproblem's solution determined the specific alternative machine options for each operation. The resulting scheduling subproblem has only one operation to machine assignment and is similar to the classic job-shop scheduling. The results of the first phase indicated that the decomposed

scheme is comparable to the optimal scheme. However, the decomposed scheme requires substantially lower computational effort than the optimal one.

In the second phase, the decomposed scheme was compared against seven real-time type schemes including SPT (shortest processing time), LPT (longest processing time), MWKR (most work remaining), LWKR (least work remaining), MOPR (most operations remaining), LOPR (least operations remaining) and SPT/TOT (shortest processing time divided by total processing time for the part). The best of the real-time scheme was found to be the SPT dispatching rule coupled with look-ahead control policy. However, the off-line decomposed scheme resulted in significantly better system performance than any of the real-time schemes. Additionally, the results indicated that at higher levels of routing flexibility (the probability that an operation has an alternative machine that can process it), the off-line scheme is significantly better than all of the real-time schemes.

Hutchison et al. (1991) also extended a similar study for the static scheduling environment. The scheduling approaches for random job-shop flexible manufacturing systems in the static environment were investigated to examine the effect of routing flexibility and scheduling schemes. The performance of two off-line schemes and one real-time scheme (i.e., SPT coupled with look-ahead control policy) were compared using the makespan (the total completion time) as a criterion. The results suggested that both off-line schemes are found to be much better than the real-time scheme. In addition, the relative performance of the real-time scheme deteriorates as routing flexibility increases.

Logendran et al. (1994) conducted a study of the cell formation in a cellular manufacturing system when alternative process plans are present. The study focused on the problem of determining the number of machines of each type and a unique process plan for each part prior to actually assigning parts and machines to each manufacturing cell. The authors formulated the problem as a general/binary integer linear programming and proved that the complexity of the problem is NP-hard in the strong sense. Subsequently, tabu search-based heuristics were developed to solve the problem instead of using the implicit enumerative method (i.e., the branch-and-bound technique) because it would require exceedingly large computational time even for moderately-sized problems.

Two tabu search-based heuristic algorithms have been developed and extended into two different methods. All four of them were tested in three different problem structures. The comparison of both heuristics with each method was performed based on a randomized block design. The results of the study indicated that the second algorithm was superior to the first algorithm on both methods. The authors also concluded that there is a real need for efficient higher-level heuristics to solve large-sized practical problems, and one cannot rely on an implicit enumerative method, namely the branch-and-bound technique for solving such problems.

Skorin-Kapov and Vakharia (1993) studied the scheduling of a flow-line manufacturing cell using a tabu search-based approach. The authors proposed the tabu search approach for sequencing part families and jobs within each family in a manufacturing cell. Part families consist of a set of similar jobs in terms of processing requirements. The authors developed a two-level tabu search-based approach that searched for the optimal/near-optimal sequence of part families at the first level and for the optimal/near-optimal sequence of jobs within each family in the second level. The comparison of the proposed heuristic with a simulated annealing-based heuristic (SAH) was also performed. Furthermore, the paper also cited the performance of six different versions of the proposed tabu search-based heuristic to examine the impact of tabu-list sizes and the use of long-term memory. The study concluded that in most cases the tabu search-based procedure outperformed the SAH procedure in terms of relative makespan and also required less computation time. The comparison of six different tabu search-based heuristics indicated that the use of variable tabu-list sizes and long-term memory based on maximal frequency is preferred at the expense of increased computation time.

3. PROBLEM STATEMENT

A flexible manufacturing system (FMS) provides the flexibility required for small batch manufacturing, at levels of productivity normally achieved with large volume manufacturing. One aspect of the FMS operational decisions that greatly influences FMS performance is the scheduling decisions. The scheduling decisions involve three different decisions which are the part loading decision, the loading time decision and the part routing decision (Suri and Whitney, 1984). It is desirable to make all scheduling decisions simultaneously so that the best possible schedule can be found.

The scheduling decisions can be made either off-line or real-time. Off-line scheduling decision is made for all operations at one time for the entire shop prior to the production while real-time scheduling decision is made for one operation at a time in real time of production. A study by Hutchison et al. (1991) investigated the scheduling approaches for random job-shop flexible manufacturing systems. The results of the study suggested that the off-line scheduling scheme results in significantly better system performance than the real-time scheduling scheme. However, the investigation assumed that each part has a unique process plan. It is important to recognize that, in a practical manufacturing environment, each part can have two or more alternative process plans and each operation of a part can be performed on alternative machines. An example on gear manufacture illustrating the use of alternative process plans in a cellular manufacturing system presented by Rajamani et al. (1990) further illustrates this point.

"If the initial raw material used in the manufacture of a gear is in the form of bar stock, the following eight steps are required to transform the raw material into a finished gear.

Processing steps (PS):

PS 1: Facing

PS 2: Turning

PS 3: Parting off

PS 4: Facing

PS 5: Centring

PS 6: Drilling

PS 7: Slotting

PS 8: Gear teeth cutting

A different set of processing steps can be identified if the raw material is in a different form, say blanks either cast or forged. Once the processing steps have been identified, the process planner determines the possible sequences of processing before grouping the processing steps into operations. The eight processing steps in the gear manufacture can be grouped into different sets as follows:

	Plan 1	Plan 2
Operation 1	PS 1, 2, 3	PS 1, 2, 3
Operation 2	PS 4, 5,	PS 4, 5, 6
Operation 3	PS 7	PS 7, 8
Operation 4	PS 8	

It is possible to alter such grouping to suit the manufacturing system requirements. For example, in gear manufacture the first six processing steps can be combined to perform them in one setup, say, on a turret lathe. Further, processing step PS 6 can be separated and performed on a drilling machine. Also, each operation in the plans can be performed on a number of compatible machines. For example, the gear-teeth-cutting operation can be performed on either a milling or gear hobbing machine if plan 1 is used. If plan 2 is used where the gear-teeth-cutting and slotting operations have been combined, it can only be performed on a milling machine."

It is, therefore, more realistic to include the alternative process plans in the problem environment. In this research an attempt has been made to investigate the scheduling of parts in FMSs when alternative process plans are present.

The issue of alternative process plans has been addressed by other researchers recently in the context of cellular manufacturing systems (Kusiak, 1987, Choobineh, 1988, and Logendran et al., 1994). Among these, Logendran et al. (1994) studied the design of cellular manufacturing systems using the higher-level heuristics, based on the concept known as tabu search. The design problem in the study was formulated as the

general/binary integer linear programming problem and then successfully solved by the tabu search-based heuristic. In the same context, Skorin-Kapov and Vakharia (1993) studied the scheduling of a flow-line manufacturing cell using a tabu search approach. Although in Skorin-Kapov and Vakharias'(1993) study alternative process plans were not allowed, the results of the study concluded that a tabu search-based heuristic outperformed an alternate heuristic based on simulated annealing (Vakharia and Chang, 1990) by generating better solutions with less computational effort.

The tabu search-based heuristics have been widely used on a number of classical and practical combinatorial problems for obtaining optimal/near-optimal solutions (Glover, 1990b). Tabu search strategies have already been successfully applied to different machine scheduling problems (Eck, 1989, Glover and Laguna, 1989, Taillard, 1989, Widmer, 1991, Barnes and Laguna, 1993, Reeves, 1993, and Taillard, 1993).

Glover (1989, 1990a,b), in series of articles, successfully describes the use of tabu search for solving hard combinatorial optimization problems. This method frequently has obtained optimal and near-optimal solutions with less computational effort than previously obtained with alternative strategies, particularly in arena of production scheduling. The application of tabu search to the scheduling problems included a job shop scheduling (Eck, 1989 and Taillard, 1989), a single-machine scheduling (Laguna et al., 1993 and Laguna and Glover, 1993), a flow shop sequencing problem (Widmer and Hertz, 1989 and Taillard, 1990) and a scheduling problem in a flow-line manufacturing cell (Skorin-Kapov and Vakharia, 1993).

The objectives of this research are as follows:

(i) to develop a mathematical model which focuses on minimizing the total completion time of all part types included in the FMS, when each part can have more than one process plan and each operation of a part can be performed on alternative machines, and

(ii) to develop an efficient scheduling methodology that would solve the model presented in (i).

The mathematical model for this problem is developed as a mixed-(binary) integer linear programming problem. The number of parts, number of alternative process plans

for each part, number of operations required of each part per process plan, the number of alternative machines, and the processing time required to perform a particular operation of a part on a machine per process plan are all known quantities.

The objective function focuses on minimizing the total completion time of all parts, commonly referred to as the makespan. The constraints force that only one process plan be selected for each part; that all operations of a part as per the chosen process plan be performed on one of the available machines; that the precedence relationships of operations are maintained; that no two operations are processed on a machine simultaneously; that the completion time of the first operation of a part on one of the available machines must be equal to or greater than its processing time; that the completion time for an operation on the machine that is not used must equal to zero; and that the makespan is the largest of all operation completion times.

The computational complexity of the problem considered here has been proven NP-hard in the strong sense (refer to subsection 4.6). As such, an implicit enumerative method such as the branch-and-bound technique can be used to solve only small-sized problems. Such an algorithm would be too time consuming even for moderately-sized problems due to the combinatorial nature of the problem. A higher-level heuristic algorithm based on a concept known as tabu search is, therefore, developed to efficiently solve large-sized practical problems.

The model development, notations used in the model and the heuristic algorithm are described in the following sections. The application of the heuristic algorithm to an example problem is presented to illustrate the functionality and efficacy of the heuristic algorithm and its potential in producing good solutions when applied to the practical problems.

4. MODEL DEVELOPMENT

4.1 Background

The model developed in this research is adapted from the mathematical models proposed by Hutchison et al. (1991) and Logendran et al. (1994). Hutchison et al. (1991) formulated the FMS scheduling problem when an operation required of a part can be performed on more than one machine, as a mixed-(binary) integer programming model using minimization of makespan as the objective function. However, the authors did not consider the possibility of having an alternative process plans for each part. In contrast, Logendran et al. (1994) included alternative process plans as well as alternative machine options into the design problem of a cellular manufacturing system. In this research, we investigate the scheduling of parts in an FMS in the presence of alternative process plans and alternative machine options. Therefore, the concept of modeling alternative process plans is somewhat similar to the study of Logendran et al. (1994) and the constraints for the scheduling problem is somewhat similar to the study of Hutchison et al. (1991).

The model assumptions are stated in the next section. This is followed by sections that describe the notations used in the development of the model and the description of the constraints in the model. Finally, a mathematical model is presented and the computational complexity of the problem is described.

4.2 Assumptions

(1) All of the parts and machines are simultaneously available at time t = 0.

(2) Each part can have alternative process plans and each operation of a part can be processed on alternative machines.

(3) Processing time required to complete an operation of an entire part order is known. These times are fixed and independent of the order in which the parts

are processed. Setup time is assumed to be included in the processing time.

(4) There is only one machine of each type.

(5) Splitting of any operation required for a part is not allowed.

(6) No machine can process more than one operation at a time.

(7) An operation of a part, once started on a machine, is continued until it is completed.

(8) Transportation time required to move a part type between machines is negligible.

4.3 Notations

i = 1, 2, 3, ..., m machines

j = 1, 2, 3, ..., n parts

p = 1, 2, 3, ..., P_j process plans for part j

k = 1, 2, 3, ..., K(j,p) operations for (j,p) combination

$f_{ik}(j,p)$ = completion time of operation k on machine i for (j,p) combination

$f_{i(k+1)}(j,p)$ = completion time of operation k+1 on machine i for (j,p) combination

$pt_{ik}(j,p)$ = processing time to perform operation k on machine i for (j,p) combination

Tmax = the largest completion time for the last operation of all the jobs

Z_{jp} = $\begin{cases} 1 & \text{if part j is produced using plan p} \\ 0 & \text{otherwise} \end{cases}$

$Y_{ik}(j,p)$ = $\begin{cases} 1 & \text{if operation k for (j, p) combination is performed on machine i} \\ 0 & \text{otherwise} \end{cases}$

$$V_{ik(j,p)il(k,q)} = \begin{cases} 1 \text{ if operation k for (j, p) combination on machine i} \\ \quad \text{precedes operation l for (k, q) combination on machine i} \\ 0 \text{ otherwise} \end{cases}$$

$$a_{ik} = \begin{cases} 1 \text{ if machine i can perform operation k} \\ 0 \text{ otherwise} \end{cases}$$

$$b_k(j,p) = \begin{cases} 1 \text{ if operation k has to be performed for (j, p) combination} \\ 0 \text{ otherwise} \end{cases}$$

H = an arbitrarily large number

4.4 Mathematical Model

Min Tmax

subject to

$$\sum_{p=1}^{P_j} Z_{jp} = 1 \qquad \text{for } j = 1, 2, 3, \ldots, n \qquad (1)$$

$$\sum_{i=1}^{m} a_{ik} Y_{ik}(j,p) = b_k(j,p) Z_{jp} \qquad \text{for } j = 1, 2, 3, \ldots, n \qquad (2)$$

$$k = 1, 2, 3, \ldots, K(j,p)$$

$$p = 1, 2, 3, \ldots, P_j$$

$$f_{iK}(j,p) \leq \text{Tmax} \qquad \text{for } j = 1, 2, 3, \ldots n \qquad (3)$$

$$p = 1, 2, 3, \ldots P_j$$

$$i = 1, 2, 3, \ldots m$$

$$f_{i(1)}(j,p) \geq pt_{i(1)}(j,p)\, Y_{i(1)}(j,p) \qquad \text{for } j = 1, 2, 3, \ldots n \qquad (4)$$

$$p = 1, 2, 3, \ldots P_j$$

$$i = 1, 2, 3, \ldots m$$

$$f_{ik}(j,p) \leq HY_{ik}(j,p) \qquad \text{for } j = 1, 2, 3, \ldots n;\ p = 1, 2, 3, \ldots P_j;\ k = 1, 2, 3, \ldots K(j,p);\ i = 1, 2, 3, \ldots m \tag{5}$$

$$f_{i(k+1)}(j,p) - f_{hk}(j,p) + H[1 - a_{i(k+1)}Y_{i(k+1)}(j,p)] \geq pt_{i(k+1)}(j,p) \tag{6}$$

$$\text{for } j = 1, 2, 3, \ldots n;\ p = 1, 2, 3, \ldots P_j;\ k = 1, 2, 3, \ldots K(j,p)-1;\ i, h = 1, 2, 3, \ldots m$$

$$\begin{aligned} f_{ik}(r,p') - f_{il}(s,p'') + HV_{ik}(r,p')_{il}(s,p'') &\geq pt_{ik}(r,p')\,Y_{ik}(r,p') \\ f_{il}(s,p'') - f_{ik}(r,p') + H[1 - V_{ik}(r,p')_{il}(s,p'')] &\geq pt_{il}(s,p'')\,Y_{il}(s,p'') \end{aligned} \tag{7}$$

$$\text{for } r, s = 1, 2, 3, \ldots n\ (r \neq s);\ p' = 1, 2, 3, \ldots P_r;\ p'' = 1, 2, 3, \ldots P_s;\ k = 1, 2, 3, \ldots K(r,p');\ l = 1, 2, 3, \ldots K(s,p'');\ i = 1, 2, 3, \ldots m$$

In the above model, Tmax, Z_{jp}, $Y_{ik}(j,p)$, $V_{ik}(j,p)_{il}(k,q)$ and $f_{ik}(j,p)$ are decision variables.

4.5 Model Description

The problem is formulated as a mixed-(binary) integer linear programming model. The formulation shown below has an objective function which focuses on minimizing the makespan. The makespan is also the maximum completion time of all machining operations, assuming that all jobs (parts) are present at time zero. The constraints of the model can be described as follows.

Constraint (1) assures that only one alternative process plan can be used for each part.

Constraint (2) ensures that all of the operations of a part according to the process plan selected must be performed on one of the available machines.

Constraint (3) establishes Tmax as the largest of all operation completion times.

Constraint (4) assures that the completion time of the first operation required of a part on a machine must be equal to or greater than its processing time.

Constraint (5) ensures that if a machine option is not used to perform an operation required of a part, its completion time must be equal to zero.

Constraint (6) assures that the precedence relationships, reflecting the order of operations are maintained (i.e., operation k must be precede the operation k+1 of part j). The completion time of operation k+1 of part j must be greater than the completion time of operation k of part j by at least the processing time of operation k+1. This constraint guarantees that operation k must be finished before performing the subsequent operation k+1.

Constraint (7) guarantees that no two operations are processed on a machine simultaneously. If the same machine is used, the completion time of operation k of part r according to process plan p′ must be greater than the completion time of operation l of all of the remaining parts according to process plan p″ by at least the processing time of operation k, if operation k is processed on the machine before operation l.

4.6 Computational Complexity of the Research Problem

The mathematical model developed above is a mixed-(binary) integer linear programming model. Most mixed-(binary) integer linear programming problems falls into the class of NP-complete problems (Garey and Johnson, 1979) and our problem is no exception. The NP-complete problems indicate that it is unlikely to find a polynomially bounded optimizing algorithm for the problems because it requires a solution time that is bounded by a non-polynomial (i.e., exponential) function of the problem size. The computational complexity of the problem is investigated and proven to be NP-hard in the strong sense by considering a special case of the research problem. If the special case of

the research problem is strongly NP-hard, then the research problem described by the above model is NP-hard in the strong sense as well.

Consider the special case problem where each of n parts have a unique process plan. Each part has exactly three operations, processed in any order, with nonzero processing times on each of the three machines, M1, M2 and M3. This special case is clearly a 3-machine job-shop problem with an objective of minimizing the makespan.
It should be pointed out that a 2-machine job-shop problem with an objective of minimizing the makespan can be solved in polynomial time (Jackson, 1956). Within the context of job-shop scheduling, a problem with one machine is trivial.

Garey et al. (1976) have proven that the decision problem, obtained by reformulating the optimization problem corresponding to the makespan minimization of a 3-machine *flowshop* problem, is strongly NP-hard by showing a 3-partition problem is polynomially reducible to the reformulated decision problem. Our special case of the research problem is indeed a generalization of the corresponding 3-machine flowshop problem. As such, the special case considered above is also strongly NP-hard, and so is the proposed original research problem.

The number of potential feasible solutions could be astronomically large, even for a small problem. An implicit search algorithm such as the branch-and-bound technique can be used to solve a problem with only a small number of parts and process plans. However, for a problem with moderate number of parts and process plans, such an algorithm would turn out to be too time consuming and would require excessive computational effort. Thus, in the next chapter a higher-level heuristic, based on a concept known as tabu search, is proposed to efficiently solve large problems.

5. HEURISTIC ALGORITHM

5.1 Introduction

The tabu search method is a higher-level heuristic algorithm for solving combinatorial optimization problems, which is designed to guide other methods to overcome the limitations of local optimality. The method can be superimposed on any procedure whose process can be characterized as performing a sequence of moves to transform one solution into another provided an evaluation function exists for measuring the attractiveness of these moves. Tabu search introduced by Fred Glover (Glover, 1986) has been shown to be a remarkably effective approach in a wide variety of classical and practical optimization problems ranging from scheduling to telecommunications such as traveling salesman problem, employee scheduling, sequencing and production scheduling, character recognition and neutral networks.

The strategies of tabu search method employs conditions for constraining and freeing the search process by the use of tabu restrictions and aspiration criteria. Tabu restrictions are restrictions placed on certain branches of the search which may be unproductive due to predefined rules. A tabu restricted search can be followed only if these aspiration criteria are satisfied. The primary objective of the tabu restrictions is to permit the method to go beyond points of local optimality while still making high quality moves at each step. These restrictions in conjunction with the aspiration criteria help the search progress forward. Tabu search also has the ability to intensify the search (via the use of short-term memory) and diversify the search (via the use of long-term memory) into new regions using memory functions of varying time spans.

The remaining sections of this chapter are organized as follows. The mechanism of the tabu search method is described in the next section. The steps associated with the heuristic algorithm for the research problem are then presented. Finally, an application of the tabu search-based heuristic to an example problem is illustrated.

5.2 Mechanisms

The underlying principles of tabu search has been documented in Glover (1989, 1990a,b) and Laguna et al. (1991). The mechanisms used in the algorithm can be summarized as follows.

The form of a simple heuristic that uses moves to go from one solution to another until no improving moves are available is the well known hill-climbing heuristic. The hill-climbing heuristic can readily be embedded within the tabu search strategy. In general, a hill-climbing procedure progresses from an initial feasible solution along the path that changes the objective function value in a uniformly descending direction (for minimization) or ascending direction (for maximization) until no further improvement of the objective function value is made possible through the available moves. At the stopping point, the solution obtained is a local optimum, which is often not a global optimum for combinatorial problems. In this context, tabu search provides a guiding framework to escape the trap of local optimality. Basically, the procedure is similar to the hill-climbing strategy in that the starting initial solution would be perturbed to find the candidate moves that need to be considered to move the starting initial solution to the new better solution. At each step, the best member of the candidate moves (called the best move) would be selected. However, when a local optimum is found, the tabu search would select the best move from a list of candidate moves in a way that guides the search out of the local optimum. The process of tabu search to surpass local optimality is to introduce the mechanism that would make certain moves forbidden (tabu), intensify the search by using short-term memory and diversify the search by using the long-term memory.

The mechanism to forbid certain moves is designed to prevent the reversal and the repetition of the moves. In order to prevent cycling when local optimality is reached, tabu search introduces the condition or restriction called tabu restriction to forbid certain moves that would lead back to some earlier solutions. The tabu restriction is intended to permit the search process to go beyond the local optimal point while still making an improving move. The tabu restriction does not operate separately but is counterbalanced by the aspiration criterion. The aspiration criterion is simply the condition that overrides

the tabu restriction. In other words, the aspiration criterion would allow the forbidden or tabu move to be performed in the search process when the aspiration criterion was satisfied.

The moves that are restricted by the tabu restriction will be stored in the tabu list in the order in which they are made. The tabu list contains the tabu moves that would bring us to revisit the earlier solutions. Therefore, the tabu list is like the list or the record of the recent moves that have been made. The short-term memory is used to determine how long a tabu restriction will be enforced by determining the size of the tabu list. The tabu-list size is the parameter that is used to determine how many recent moves to be restricted. By adjusting the tabu-list size, intensification and diversification of the search can be made possible by narrowing down the possible moves (non-tabu moves) via increasing the tabu-list size and by expanding the possible moves (non-tabu moves) via decreasing the tabu-list size, respectively.

The long-term memory is designed to help the diversification process for the entire search procedure. The long-term memory enhances the potential of identifying new starting points in the regions that were not (or infrequently) previously searched after the search has been unable to find a better solution from the current starting point. By keeping track of the most frequently searched regions, the new initial starting points can be selected in a way that drives the search to investigate the unexplored regions.

The mathematical description of this concept as described in Widmer and Hertz (1989) and Taillard (1990) can be documented as follows. Suppose that the problem is

Minimize $c(x)$

Subject to $x \in X$

where $c(x)$ is any function of a discrete variable x and X is the set of feasible solutions. The basic step of the tabu search procedure starts with any feasible solution $x \in X$ and then moving to the best neighbor x' by function m ($m \in M(x)$). That is, from a current solution x, a function m transforms x into x' which is the new feasible solution ($x' = m(x)$). This transformation is called a move, and the neighborhood of x is defined as $\{x': x' = m(x); x, x' \in X, m \in M(x)\}$ which is the set of all possible moves applicable to current solution x that can transform it into new solutions.

In order to move wisely from the current solution to the next neighboring solution, the best neighbor in the neighborhood of the current solution is selected, where "best" is determined as that neighbor which optimizes the objective function (in this case, minimize c(x) over M(x)). The interesting feature of tabu search is the construction of a list T which contains the element t associated with m and x. The element t is the set of forbidden (tabu) moves which are not allowed at the present iteration. All the tabu moves are stored in a tabu list T. The reason behind the tabu list T is to exclude moves which would bring us back where we were at some previous iteration and keep the search trapped in local optimum.

For example, if a move from solution x to solution x′ is considered, the tabu list T would contain the transformation (move) to x′ that will take it back to x. Therefore, when further moves from x′ is considered that transformation (move) is now forbidden (tabu) from being applied to x′. A move remains a tabu move only for a certain number of iterations. The size of tabu list (tabu-list size) is the parameter that has to be predetermined. The tabu list is updated circularly according to the tabu-list size parameter. If a tabu move is to be added to the tabu list that has already attained its limit, the last old move (oldest element) would be dropped from the list before adding the new tabu move.

Up to this point, the stopping criteria should also be defined to stop the search. A general stopping criteria may be the maximum number of iterations. In other words, after a certain number of steps have been performed, the entire procedure would be stopped. Another general condition would be to stop the procedure if a prescribed number of iterations without improving the best solution has been performed. In this case, if there is no improvement in the objective function value after a specific number of iterations has been performed, the entire search would be terminated.

5.3 Steps Associated with the Heuristic Algorithm

The problem considered here is the scheduling of parts in a flexible manufacturing system which utilizes alternative process plans with alternative machines. In this context, each part can have more than one process plan (alternative process plans) and, within a process plan, each operation of a part can be performed on more than one machine (alternative machine options). The tabu search heuristic is used for both levels of the problem; the alternative process plans level (outside search), to find the optimal/near-optimal process plan and the alternative machines level (inside search), to find the optimal/near-optimal machine option within the optimal/near-optimal process plan.

The general idea of the application of tabu search-based heuristics to the problem is that the final solution is composed of the solution corresponding to the optimal/near-optimal process plan together with its solution corresponding to the optimal/near-optimal machine options that give the minimum makespan. The tabu search method is applied for the outside search to move from a solution corresponding to one process plan to another and for the inside search to move from a solution corresponding to one machine option to another. The relationship between the outside and inside search is that, once the outside search is performed to get the process planning configuration, the search process is switched to the inside. The inside search is performed to search for the solution corresponding to the optimal/near-optimal machine option as well as the resulting makespan of that process planning configuration from the outside search. When the solution corresponding to the optimal/near-optimal machine option is found, the search process is switched back to the outside search to find a new and better process planning solution. As a result, the search process will switch back and forth between the outside search and the inside search as the search progresses.

A feasible schedule Ω consists of a sequence of process plans called Ωp and a sequence of machine options within the selected process plan Ωp called Ωm. Two different neighborhoods for such a feasible solution are defined as follows.

- $Np(\Omega p) = \{\Omega' p$: $\Omega' p$ is a sequence of process plans obtained from Ωp by perturbing on the process plan for each part, yet one part at a time. The perturbation on a

process plan is simply to change the current process plan to another alternative process plan for each part.}

- $Nm(\Omega m) = \{\Omega' m$: $\Omega' m$ is a sequence of alternative machine options (corresponding to a sequence of process plans Ωp) obtained from Ωm by perturbing on the machine options for an operation of each part, one operation at a time. Similar to the perturbation on a process plan, the perturbation on a machine option is to change the current machine option to an alternative machine option for each operation of every part.}

The steps associated with the heuristic algorithm are presented next.

Step 1: Randomly generate the initial configuration/solution point for alternative process plans. The configuration for outside or the alternative process plans level search can be written in the form $\{pp_1, pp_2, ..., pp_N\}$, where pp_i denotes the selected process plan for part i and N is the total number of parts.

Step 2: Using the initial process planning configuration or Ωp_0 as a seed, completely evaluate its neighborhoods ($Np(\Omega p_0)$) by perturbing on a process plan for each part, one part at a time. In other words, when a process plan of one part is being perturbed, the other parts remain at their old process plans. The perturbation on a process plan is performed on all possible alternative process plans for each part. The result of the perturbation is a set of different process planning configurations which are considered the neighborhoods of the initial process planning configuration. Before moving the initial process planning configuration to one of its neighborhood configurations documented above, invoke the inside or the machine options level search for each and every configuration to find its optimal/near-optimal machine options and the resulting makespan. These will be used as suitable criteria for performing future moves.

Step 3: Initiate the inside search by randomly generating the configuration/solution point, representing the machine option for performing each operation of every part. The configuration for inside search can be written in the form $\{m_{11}, m_{21}, ..., m_{N1} \mid m_{12}, m_{22}, ...,$

m_{N2} | ... ... | m_{1K}, m_{2K}, ...,m_{NK}}, where m_{ij} represents the selected machine option performing operation j of part i. N and K is the total number of parts and the maximum number of all possible operations that would need to be performed for parts, respectively. In the event that an operation is not required to be performed for a part, "0" (zero) is used, and no machine option is selected. Each element in a machine option configuration corresponds to the machine number on which the operation of each part will be performed. The "|" sign divides the operation level.

Step 4: Similar to the outside search, using the initial machine option configuration or Ωm_0 as a seed, completely evaluate its neighborhoods $Nm(\Omega m_0)$ by perturbing on a machine option of each operation for each part, but one operation at a time. In other words, when a machine option of an operation for one part is being perturbed, the operations of other parts remain at their old machine options. The perturbation on a machine option is performed according to the order of the element in the machine option configuration. That is, m_{11} would be perturbed first, followed by m_{21}, m_{31} and so on. Perform the perturbation on a machine option to cover all possible alternative machine options for each operation of each part. The results of the perturbation is a set of different machine option configurations which is considered the neighborhoods of the initial machine option configuration.

Step 5: Evaluate the makespan for each machine option configuration in the neighborhoods using the Shortest Processing Time (SPT) dispatching rule. The reason for using SPT is that SPT had been proven to perform very well with regard to minimization of makespan in job-shop scheduling problems when no alternative process plans and alternative machine options are present.

Perform tabu search for the alternative machines level to find the optimal/near-optimal machine option by moving from the initial machine option configuration to the "best" candidate in its neighborhoods. The so-called m_move is the move that transforms a sequence of machine options (one machine option configuration) into another sequence of machine options (another machine option configuration) in the neighborhoods. The

value of a move is the difference between makespans after and before the move, and therefore an improving move has a negative value (Value of move = Makespan after moving - Makespan before moving).

Step 6: At each iteration, completely evaluate the neighborhoods of the current machine option solution, and perform the move with the smallest value of move. The following parameters of the tabu search are updated for the inside search:

(1) Inside-tabu list (m_tabu list): Each time a m_move is performed, store the operation of the part along with the machine option that was moved in the inside-tabu list (m_tabu list). Operations (along with their machine options) appeared in the m_tabu list indicate that these operations have been moved before at some previous iterations. These tabu operations are not allowed to move at the present iteration unless an aspiration criterion which allows the tabu status to be overridden is satisfied.

An operation remains a tabu (forbidden) operation only during a certain number of iterations determined by the m_tabu list size. The m_tabu list is updated circularly according to m_tabu list size, that is, if the m_tabu list was stored up to its size, before the next element is stored, the oldest one must be removed. Two types of tabu-list size are investigated in this research; the fixed tabu-list size and the variable tabu-list sizes.

- For the inside search, determine the size of the fixed tabu list by the following formula.

The fixed size of m_tabu list = $\lceil (N*K)/2 \rceil$

- For the inside search, determine the sizes of the variable tabu list by the following formulae.

The initial size of m_tabu list = $\lceil (N*K)/2 \rceil$

The decreased size of m_tabu list = $\lceil (N*K)/3 \rceil$

The increased size of m_tabu list = $\lceil (N*K)/1.5 \rceil$

where N is the total number of parts and K is the maximum number of all possible operations that would need to be performed for parts.

The sizes of the inside-tabu list (m_tabu list) is dependent on the number of parts, the number of operations and the number of alternative machines. The total number of possible moves of a solution for the inside search would increase as one of the above

numbers increases. In this research, only one alternative machine option is allowed for each operation. Therefore, the sizes of the tabu list is only dependent upon the number of parts and the number of operations.

Based on preliminary investigation of test problems, it was found appropriate to use a reduction factor of 2 in the denominator of the formula for the fixed size of m_tabu list. Correspondingly, the reduction factors for the decreased and the increased size of m_tabu list in the variable tabu list were found to be 3 and 1.5, respectively.

To allow a tabu move to be performed, the aspiration criterion/level for inside search, namely inside_AL, is created and initially set equal to the makespan of the initial machine option configuration. The inside_AL is updated if the makespan evaluated for the current machine option solution is found to be better than the makespan of the best machine option solution found so far.

(2) Inside candidate list (ICL) and inside index list (IIL): Create two lists, namely the inside candidate list and the inside index list. The IIL contains the local optima evaluated as the inside search progresses, while the ICL consists of potential machine option configurations chosen to perform future perturbations.

Admit the initial machine option configuration (Ωm_0) into both ICL and IIL. It is admitted into the IIL because it is considered the first local optimum. Evaluate the makespan of the initial configuration as MS_0. Using the minimum makespan as the objective function, perform the m_move to move the initial configuration (Ωm_0) to the next configuration (Ωm_1) in the neighborhoods. If the makespan of the next configuration (MS_1) is lower than the makespan of the initial configuration (MS_0), then the next configuration (MS_1) would receive a star (*), indicating that it has potential of becoming the next local optimum.

The next configuration (Ωm_1) is considered for perturbation next. If the new configuration point (Ωm_2) has a makespan $MS_2 \geq MS_1$, then the configuration corresponding to MS_1 would receive two star (**) and would be admitted to the IIL as it is the next local optimum. If, on the other hand, $MS_2 < MS_1$, then the configuration corresponding to MS_2 would receive a star. For the inside search, the final solution/configuration, indicating which machine option should be used for each operation

of every part, is selected as the one with the smallest makespan from among the entries into the IIL.

(3) Number of iterations without improvement for inside search: Every time a m_move is performed, increase the number of iterations (m_iter) by one. The number of iterations indicates the total number of moves for the inside search. Create and update the number of iterations without improvement for the inside search the same way as m_iter, except that increase it by one only if there is no improvement in the objective function (makespan) value after moving from one machine option configuration to another. An improvement means the value of move having a negative value. Thus, no improvement means that the makespan after the move is equal to or greater than the makespan before the move.

(4) Inside long-term memory (M_LTM): The inside long-term memory (M_LTM) is the frequency matrix that keeps track of the tenure of machine options. That is, the long-term memory will keep track of the number of times that each operation of each part has been performed on each machine. The inside long-term memory is used as a tool to create a restart in order to diversify the inside search. The inside long-term memory based restart would be created using the frequency entries in the M_LTM matrix. The M_LTM matrix is updated continuously as the search progresses. Every time a m_move is performed to move a current machine option configuration to a new machine option configuration, increase by one the entries of the M_LTM matrix corresponding to the new machine option configuration. By keeping track of the frequency of machine option being used, the M_LTM matrix provides the information about which specific machine option is most or least frequently used by each operation of each part.

Step 7: To terminate the inside search, the number of iterations without improvement is used as a stopping criterion. The number of iterations without improvement is dependent on the problem size involving the total number of parts, the total number of operations and the total number of alternative machine types for each operation. The larger the size of the problem, the larger is the value of the number of iterations without improvement. For

the inside search, the number of iterations without improvement is assumed inversely proportional to the total number of process plans.

- For the fixed tabu list, the stopping criterion is evaluated as:

The number of iterations without improvement for the inside search (IIT)

$$= \text{int}\left(\frac{\left(\sum_{i=1}^{N}\sum_{j=1}^{P_j} O_{ij}\right) * M * N}{\sum_{j=1}^{N} P_j * \text{reduction factor}} \right)$$

where N = total number of parts.

P_j = total number of processing plans for part j.

O_{ij} = total number of operations for part i according to plan j.

M = total number of machine types that can process each operation (M = 2 in this research).

and the reduction factor is assumed equal to 4 for the inside search.

- For the variable tabu list, the stopping criteria are evaluated as:

(i) If there is no improvement in the last [int (IIT/3)] iterations with the initial m_tabu list size, decrease the m_tabu list size to the decreased size evaluated in step 6.

(ii) If there is no improvement in the last [int (IIT/3)] iterations with the decreased m_tabu list size, increase the m_tabu list size to the increased size evaluated in step 6.

(iii) If there is no improvement in the last [int (IIT/3)] iterations with the increased m_tabu list size, stop performing the move and start to diversify the inside search.

Step 8: To diversify the inside search, the inside long-term memory based restart is generated by using M_LTM frequency matrix. The restarts generate new initial configurations which is intended to search in the regions that were not previously investigated using the old initial configuration as the starting point. Two types of long-term memory are investigated in this research: the long-term memory based on minimal frequencies (LTM_MIN) and the long-term memory based on maximal frequencies (LTM_MAX).

- For the LTM_MAX, generate the restarts by taking the maximal entry in the matrix, and fixing the operation which has the maximal entry to the machine option corresponding to that entry throughout the subsequent search.
- For the LTM_MIN, generate the restarts by taking the minimal entry in the matrix, and fixing the operation which has the minimal entry to the machine option corresponding to that entry throughout the subsequent search.

Once the restart configuration is obtained, reinitialize the inside-tabu list (m_tabu list) and repeat steps 4, 5, 6 and 7 using this restart configuration as a new starting point until the required number of restarts for the inside search has been performed. The required number of restarts for the inside search is assumed equal to 4 in this research. Four starting solutions had been previously used by Laguna et al. (1993) in a single-machine scheduling problem.

Using the approach based on minimal frequencies will create new initial configurations in the search regions not investigated so far. On the other hand, if the approach based on maximal frequencies is used, new initial configurations will be created in the regions considered "good" during the previous search.

Step 9: When the required number of restarts for the inside search has been reached, the optimal/near-optimal machine option configuration would be determined as the one with the smallest makespan from all four restarts. Now, the direction of the search would be switched from inside to outside search.

Perform steps 3 through step 8 for each of the outside configurations in the neighborhoods ($Np(\Omega p_0)$) to find its optimal/near-optimal machine option and the resulting makespan. Every time an inside search is invoked for a new process planning configuration in the neighborhoods ($Np(\Omega p_0)$), all parameters for the inside search including m_tabu list, m_iter, ICL, IIL, inside_AL, and M_LTM must be reinitialized.

Similar to the inside search, perform tabu search for the alternative process plans level (outside search) to find the optimal/near-optimal process plan by moving from the initial process planning configuration to the "best" candidate in its neighborhoods. The p_move identified is the move that transforms a sequence of process plans (one process

planning configuration) into another sequence of process plans (another process planning configuration) in the neighborhoods. By using the makespan resulting from going inside each process planning configuration in the neighborhoods as a criterion, the p_move is actually performed in the same manner as the m_move. The value of move and the aspiration criterion would also operate in the same way as those for the inside search.

Step 10: This step is similar to step 6 performed for inside search. At each iteration, completely evaluate the neighborhoods of the current process planning solution, and perform the move with the smallest value of move. The following parameters of the tabu search are updated for the outside search:

(1) Outside-tabu list (p_tabu list): Each time a p_move is performed, store the process plan along with the part that was moved in the outside-tabu list (p_tabu list). The p_tabu list is also updated circularly as the m_tabu list in the inside search. Two types of p_tabu list size are investigated here as well.

- For the outside search, determine the size of the fixed tabu list by the following formula.

$$\text{The fixed size of p\_tabu list} = \left\lceil [\sum_{\text{all } j} (P_j - 1)] / 2 \right\rceil$$

- For the outside search, determine the sizes of the variable tabu list by the following formulae.

$$\text{The initial size of p\_tabu list} = \left\lceil [\sum_{\text{all } j} (P_j - 1)] / 2 \right\rceil$$

$$\text{The decreased size of p\_tabu list} = \left\lceil [\sum_{\text{all } j} (P_j - 1)] / 3 \right\rceil$$

$$\text{The increased size of p\_tabu list} = \sum_{\text{all } j} (P_j - 1)$$

where P_j is the total number of process plans for part j.

The sizes of the outside-tabu list (p_tabu list) are dependent on the number of alternative process plans for each part. For each part, the maximum number of possible moves of a solution for the outside search is equal to the sum of the alternative process plans minus one (one here corresponds to the currently selected process plan).

Similar to the inside search, the aspiration criterion/level for outside, namely outside_AL, to override a tabu status and allow a tabu move to be performed is also created and, initially, set equal to the makespan of the initial process planning configuration. As for the inside search, the outside_AL is updated if the makespan evaluated for the current process planning solution is found to be better than the best process planning solution found so far.

(2) Outside candidate list (OCL) and outside index list (OIL): As for the inside search, create an outside candidate list (OCL) and an outside index list (OIL). OIL contains the local optima evaluated as the outside search progresses, while OCL consists of potential process planning configurations selected to perform future perturbations. The OCL and OIL are analogous to the ICL and IIL, respectively. Thus, the approach used for admitting the configuration to the OCL and OIL would be the same as those for the ICL and IIL. As for the inside search, the final configuration/solution, indicating which process plan should be used for each part, is selected as the one with the smallest makespan from among the entries in the OIL.

(3) Number of iterations without improvement for outside search: Every time a p_move is performed, increase the number of iterations (p_iter) by one, as in m_move and m_iter. The number of iterations without improvement for the outside search is also created and updated similar to those for the inside search. Thus, the number of iterations without improvement for the outside search is increased by one if no improvement is observed after moving from one process planning configuration to another.

(4) Outside long-term memory (P_LTM): The outside long-term memory (P_LTM), comparable to M_LTM, is the frequency matrix that keeps track of the tenure of process plans. In the same fashion as the M_LTM, the P_LTM matrix is updated continuously as the search progresses. Every time a p_move is performed to move a current process planning configuration to a new process planning configuration, increase by one the entry of the P_LTM matrix corresponding to the new process planning configuration. By keeping track of the frequency of process plan being used, the P_LTM matrix provides the information about which specific process plan is most or least frequently used by each part. The frequency entries in the P_LTM matrix will also be

employed to create the restarts for the outside search, similar to the manner the M_LTM matrix is used in the inside search.

Step 11: Similar to step 7 for the inside search, the number of iterations without improvement would be used as a stopping criterion to terminate the outside search. The number of iterations without improvement should be increased as the size of the problem becomes larger. The number of iterations without improvement for outside search is assumed inversely proportional to the total number of machines.

- For the fixed tabu list, the stopping criterion is evaluated as:

The number of iterations without improvement for the outside search (OIT)

$$= \text{int}\left(\frac{\sum_{\text{all } j} P_j * K}{\text{reduction factor} * m} \right)$$

where P_j = total number of process plans for part j.

K = maximum number of all possible operations that would need to be performed for parts.

m = total number of machines.

and the reduction factor is assumed equal to 2 for the outside search.

- For the variable tabu list, the stopping criteria are evaluated as:

(i) If there is no improvement in the last [int (OIT/3)] iterations with the initial p_tabu list size, decrease the p_tabu list size to the decreased size evaluated in step 10.

(ii) If there is no improvement in the last [int (OIT/3)] iterations with the decreased p_tabu list size, increase the p_tabu list size to the increased size evaluated in step 10.

(iii) If there is no improvement in the last [int (OIT/3)] iterations with the increased p_tabu list size, stop performing the move and start to diversify the outside search.

However, preliminary investigations with medium and large problem structures indicated that using only the number of iterations without improvement as the only stopping criterion did not terminate the outside search within a reasonable computation time because the number of iterations without improvement was justifiably large for medium and large problem structures. Thus, maximum number of entries into the outside

index list (OIL) was introduced as a second stopping criterion. The termination of the search is dictated by whichever stopping criterion is activated first. For a given problem structure, to determine an appropriate value for the maximum number of entries into the OIL, an experiment would need to be performed with different values to find the most appropriate number that is either not too small which will terminate the search too soon or not too large which will terminate the search too late.

Step 12: To diversify the outside search, the outside long-term memory based restart is generated by using P_LTM frequency matrix. Two types of long-term memory are investigated as in the inside search.

- For the LTM_MAX, generate the restarts by taking the maximal entry in the matrix, and fixing the part which has the maximal entry to the process plan corresponding to that entry throughout the subsequent search.
- For the LTM_MIN, generate the restarts by taking the minimal entry in the matrix, and fixing the part which has the minimal entry to the process plan corresponding to that entry throughout the subsequent search.

Once the restart configuration is obtained, reinitialize the outside-tabu list (p_tabu list) and repeat the outside search using this restart configuration as a new starting point until the required number of restarts for the outside search has been performed. The required number of restarts for the outside search is assumed equal to 2 in this research.

When the required number of restarts for the outside search has been reached, the entire search would be terminated. The optimal/near-optimal process planning configuration would be the one with the smallest makespan from both restarts. The optimal/near-optimal process plan along with its optimal/near-optimal machine option configuration will be combined to give the final (optimal/near-optimal) schedule with the minimum makespan.

5.4 Application of the Heuristic to Example Problem

To clearly understand the above steps, consider an example problem which involves four parts and three machines. The same example was previously considered by Rajamani et al. (1990) and later by Logendran et al. (1994) in the context of manufacturing cell design. The data pertaining to the example are presented in Table 5.1-5.3. Each part has alternative process plans. For instance, consider part 1 (P1) in Table 5.1 which has two different process plans. The first process plan requires that operations 1 and 2 must be performed, while the second process plan requires that operations 2 and 3 be performed. The data presented in Table 5.2 further indicate that there is alternative machine options for each operation. Operation 1 can be performed on machine 1 (M1) or machine 3 (M3), operation 2 can be performed on either M2 or M3, and operation 3 on either M1 or M2. Thus, there are four different ways that P1 can be processed by the first process plan. That is, operation 1 and 2 can be performed on M1 and M2, or M1 and M3, or M3 and M2, or M3 and M3, respectively. Similarly, there are four different ways that P1 can be processed by the second process plan.

Table 5.1 Data indicating alternative process plans for the example problem.

Operation	j = 1		j = 2		j = 3			j = 4	
	p = 1	p = 2	p = 1	p = 2	p = 1	p = 2	p = 3	p = 1	p = 2
k =1	1		1		1	.	1	1	1
k =2	1	1	1	1	1	1	1	1	1
k =3		1	1	1		1	1	1	

Table 5.2 Data indicating alternative machine options for the example problem.

	Whether k can be Performed on Following Machines		
	i = 1	i = 2	i = 3
k = 1	1		1
k = 2		1	1
k = 3	1	1	

Table 5.3 Processing times for the example problem.

	j = 1		j = 2		j = 3			j = 4	
	p = 1	p = 2	p = 1	p = 2	p = 1	p = 2	p = 3	p = 1	p = 2
k = 1, i = 1	5		3		2		8	1	9
k = 1, i = 3	7		4		2		9	2	8
k = 2, i = 2	3	9	7	3	3	1	5	2	9
k = 2, i = 3	4	7	7	2	4	2	3	2	10
k = 3, i = 1		8	10	6		11	7	3	
k = 3, i = 2		7	8	6		8	9	2	

Step 1: For this example, the initial feasible process planning configuration can be generated by selecting the first process plan for all parts. The resulting initial process planning configuration corresponding to the first process plan for each part is Ωp_0 = {1,1,1,1}. This configuration is used as a seed for perturbation in the next step.

Step 2: Using Ωp_0 = {1,1,1,1} as a seed, evaluate its neighborhoods by perturbing on a process plan for each part, yet one part at a time.

The neighborhoods of Ωp_0 are:

$Np(\Omega p_0)$ ={2,1,1,1}, {1,2,1,1}, {1,1,2,1}, {1,1,3,1}, and {1,1,1,2}.

Step 3: For the initial process planning configuration {1,1,1,1}, the inside search is invoked by generating the inside initial feasible machine option configuration. Although this can be randomly generated for each operation of each part, the first machine option for each operation is selected for ease of understanding in this example. In other words, the first operation is performed on machine 1 (M1), the second operation is performed on machine 2 (M2) and finally, the third operation is performed on machine 1 (M1).

The resulting initial machine option configuration is:

Ωm_0 for {1,1,1,1} = {1,1,1,1 | 2,2,2,2 | 0,1,0,1}.

Step 4: Using the initial feasible machine option configuration from step 3 as a seed, completely generate its neighborhoods by perturbing on a machine option of each operation for each part, yet one operation at a time.

The neighborhoods of Ωm_0 are:

$Nm(\Omega m_0)$ =

{3,1,1,1 | 2,2,2,2 | 0,1,0,1}, {1,3,1,1 | 2,2,2,2 | 0,1,0,1}, {1,1,3,1 | 2,2,2,2 | 0,1,0,1},
{1,1,1,3 | 2,2,2,2 | 0,1,0,1}, {1,1,1,1 | 3,2,2,2 | 0,1,0,1}, {1,1,1,1 | 2,3,2,2 | 0,1,0,1},
{1,1,1,1 | 2,2,3,2 | 0,1,0,1}, {1,1,1,1 | 2,2,2,3 | 0,1,0,1}, {1,1,1,1 | 2,2,2,2 | 0,2,0,1},
and {1,1,1,1 | 2,2,2,2 | 0,1,0,2}.

Step 5: Evaluate the makespan for each machine option configuration in the neighborhoods using the Shortest Processing Time (SPT) dispatching rule. The results obtained for the makespan are:

For {1,1,1,1 | 2,2,2,2 | 0,1,0,1}, the makespan is equal to 34 time units.

For {3,1,1,1 | 2,2,2,2 | 0,1,0,1}, the makespan is equal to 27 time units.

For {1,3,1,1 | 2,2,2,2 | 0,1,0,1}, the makespan is equal to 31 time units.

For {1,1,3,1 | 2,2,2,2 | 0,1,0,1}, the makespan is equal to 32 time units.

For {1,1,1,3 | 2,2,2,2 | 0,1,0,1}, the makespan is equal to 35 time units.

For {1,1,1,1 | 3,2,2,2 | 0,1,0,1}, the makespan is equal to 24 time units.

For {1,1,1,1 | 2,3,2,2 | 0,1,0,1}, the makespan is equal to 24 time units.

For {1,1,1,1 | 2,2,3,2 | 0,1,0,1}, the makespan is equal to 34 time units.

For {1,1,1,1 | 2,2,2,3 | 0,1,0,1}, the makespan is equal to 34 time units.

For {1,1,1,1 | 2,2,2,2 | 0,2,0,1}, the makespan is equal to 32 time units, and

For {1,1,1,1 | 2,2,2,2 | 0,1,0,2}, the makespan is equal to 31 time units.

Step_6: After the makespan is evaluated for all configurations in the neighborhoods, perform the inside move (m_move). The m_move transforms a sequence of machine options into another sequence of machine options in Nm(Ωm). The value of move is the difference between makespans before and after performing the move (Value of move = MS after performing the move - MS before performing the move).

For the example, the makespan for the initial feasible machine option configuration is 34 time units. Select {1,1,1,1 | 3,2,2,2 | 0,1,0,1} as the next configuration because it has the smallest value of move (-10) among configurations in the neighborhoods. However, {1,1,1,1 | 2,3,2,2 | 0,1,0,1} configuration has the same value of move as {1,1,1,1 | 3,2,2,2 | 0,1,0,1}. The first-best strategy is used to break ties.

After performing an inside move, update the following parameters for the inside tabu search.

(1) Inside-tabu list (m_tabu list)

In the example, a m_move is performed to move the initial feasible machine option configuration {1,1,1,1 | 2,2,2,2 | 0,1,0,1} to the next machine option configuration which is {1,1,1,1 | 3,2,2,2 | 0,1,0,1}. The machine option as well as the operation number and the part number that was moved would be stored as the first element in the inside-tabu list (m_tabu list) .

- m_tabu list = {m_{12} (2)}

The interpretation of the element in the m_tabu list is that operation 2 of part 1 was performed on machine 2 (M2) in the most recent iteration and it has been moved to the other alternative machine option (M3 in this case). The "(2)" that comes after m_{12} represents machine 2 (M2) which is the previous machine option used by operation 2 of part 1 before performing the move.

The inside aspiration level (inside_AL) is also updated each time a makespan evaluated for the current solution is better than the best makespan found so far. In this example, the resulting makespan of the new solution (24) is better than the makespan of the initial feasible solution (34). Thus, the inside_AL is updated as per the better makespan of the new solution.

- inside_AL = 24

In addition, the tabu-list size for this example is determined as follows:

- The fixed size of m_tabu list = $\lceil (4*3)/2 \rceil = 6$
- The variable sizes of m_tabu list:

The initial size of m_tabu list = fixed size = $\lceil (4*3)/2 \rceil = 6$

The decreased size of m_tabu list = $\lceil (4*3)/3 \rceil = 4$

The increased size of m_tabu list = $\lceil (4*3)/1.5 \rceil = 8$

(2) Inside candidate list (ICL) and inside index list (IIL)

The initial feasible machine option configuration is admitted into both ICL and IIL as described earlier. The next configuration obtained in this example is also admitted into ICL as it is chosen to perform future perturbations. Since, the next configuration has a better makespan (24) than that of the initial configuration (34), it is also given a star as it has the potential of becoming the next local optimal.

- ICL = { {1,1,1,1 | 2,2,2,2 | 0,1,0,1}
  {1,1,1,1 | 3,2,2,2 | 0,1,0,1}* }.
- IIL = { {1,1,1,1 | 2,2,2,2 | 0,1,0,1} }.

(3) Number of iterations without improvement for inside search

Every time a m_move is performed, the number of iterations (m_iter) is increased by one. If there is no improvement in makespan relative to the makespan of the most recent configuration, the number of iterations without improvement is increased by one. However, if in any iteration there is an improvement in makespan, the number of iterations without improvement will be reinitialized to zero.

For this example, there is evidently an improvement in makespan (from 34 to 24). Therefore, the number of iterations without improvement is not increased and it still has the value of zero.

- m_iter = 1.

- m_iter_without_improvement = 0.

(4) Inside long-term memory (M_LTM)

The inside long-term memory (M_LTM) is the frequency matrix that keeps track of the tenure of a machine option for each operation of every part throughout the inside search. Every time a new machine option solution is constructed, the entries in the frequency matrix corresponding to the operations and their respective machine options in the configuration are increased by one.

Initially, all entries in the M_LTM frequency matrix is initialized to zero. After the move from the initial machine option configuration {1,1,1,1 | 2,2,2,2 | 0,1,0,1} to the next machine option configuration {1,1,1,1 | 3,2,2,2 | 0,1,0,1} is performed, the M_LTM would be updated as shown in Table 5.4.

Table 5.4 Updated M_LTM frequency matrix for the machine option configuration {1,1,1,1 | 3,2,2,2 | 0,1,0,1}.

Operation-Part	Machine 1 (M1)	Machine 2 (M2)	Machine 3 (M3)
O1-P1	1	0	0
O1-P2	1	0	0
O1-P3	1	0	0
O1-P4	1	0	0
O2-P1	0	0	1
O2-P2	0	1	0
O2-P3	0	1	0
O2-P4	0	1	0
O3-P1	0	0	0
O3-P2	1	0	0
O3-P3	0	0	0
O3-P4	1	0	0

Step 7: To terminate the inside search, the number of iterations without improvement for this example is determined as follows.

- For the fixed tabu list, the stopping criterion is evaluated as:

The number of iterations without improvement for inside search (IIT)

= int [(42*4)/(9*4)] = 4.

- For the variable tabu list, the stopping criteria are evaluated as:

(i) If there is no improvement in the last [int (IIT/3)] iterations (i.e., 1 iteration) with the initial m_tabu list size, decrease the m_tabu list size to 4 as evaluated in step 6.

(ii) If there is no improvement in the last 1 iteration with the decreased m_tabu list size, increase the m_tabu list size to 8 as evaluated in step 6.

(iii) If there is no improvement in the last 1 iteration with the increased m_tabu list size, stop performing the move and start to diversify the inside search.

The results of the inside search with fixed tabu-list size for $\Omega p_0 = \{1,1,1,1\}$ using $\Omega m_0 = \{1,1,1,1 \mid 2,2,2,2 \mid 0,1,0,1\}$ as an initial machine option configuration are shown in Table 5.5.

Table 5.5 Results obtained for the inside search of $\Omega p_0 = \{1,1,1,1\}$, starting with $\Omega m_0 = \{1,1,1,1 \mid 2,2,2,2 \mid 0,1,0,1\}$ as an initial machine option configuration.

# m_iter	Entries into ICL	Makespan (MS)	Entries into IIL
0	{1,1,1,1 | 2,2,2,2 | 0,1,0,1}**	34	{1,1,1,1 | 2,2,2,2 | 0,1,0,1}
1	{1,1,1,1 | 3,2,2,2 | 0,1,0,1}*	24	
2	{1,1,1,1 | 3,2,2,2 | 0,2,0,1}**	21	{1,1,1,1 | 3,2,2,2 | 0,2,0,1}
3	{3,1,1,1 | 3,2,2,2 | 0,2,0,1}	21	
4	{3,3,1,1 | 3,2,2,2 | 0,2,0,1}	21	
5	{3,3,1,1 | 3,2,3,2 | 0,2,0,1}**	19	{3,3,1,1 | 3,2,3,2 | 0,2,0,1}
6	{3,3,3,1 | 3,2,3,2 | 0,2,0,1}	21	
7	{3,3,3,3 | 3,2,3,2 | 0,2,0,1}	23	
8	{3,3,3,3 | 3,2,3,2 | 0,2,0,2}	23	
9	{3,3,3,3 | 3,2,3,3 | 0,2,0,2}	25	

The inside search, starting with {1,1,1,1 | 2,2,2,2 | 0,1,0,1} is terminated after 9 moves have been made since the number of iterations without improvement for the fixed tabu-list size (4 in this example) has been reached. The best solution for the inside search starting with {1,1,1,1 | 2,2,2,2 | 0,1,0,1} is the configuration in the IIL given by {3,3,1,1 | 3,2,3,2 | 0,2,0,1} with the lowest makespan of 19 time units.

Step 8: The inside search is now diversified using a new restart, identified based on the inside long-term memory frequency matrix (M_LTM). The corresponding M_LTM frequency matrix for the inside search after the number of iterations without improvement has been reached in step 7 is shown in Table 5.6.

Table 5.6 The M_LTM frequency matrix for the inside search of $\Omega p_0 = \{1,1,1,1\}$, starting with $\Omega m_0 = \{1,1,1,1 \mid 2,2,2,2 \mid 0,1,0,1\}$ as an initial machine option configuration.

Operation-Part	Machine 1 (M1)	Machine 2 (M2)	Machine 3 (M3)
O1-P1	2	0	7
O1-P2	3	0	6
O1-P3	5	0	4
O1-P4	6	0	3
O2-P1	0	0	9
O2-P2	0	9	0
O2-P3	0	4	5
O2-P4	0	8	1
O3-P1	0	0	0
O3-P2	1	8	0
O3-P3	0	0	0
O3-P4	7	2	0

Using the long-term memory based on the maximal frequency (LTM_MAX), the first new restart is invoked by taking the maximal entry in the M_LTM matrix and fixing the operation corresponding to the maximal entry to the machine option of that entry throughout the subsequent search.

In this example, both operation 2 of part 1 and operation 2 of part 2 have the same maximal entry of 9. The row-wise first-best strategy is used to break ties. Thus, the maximal entry of 9 corresponding to performing the operation 2 of part 1 on machine 3 is used to generate the first new restart. The first new restart based on maximal frequency is {1,1,1,1 | 3,2,2,2 | 0,1,0,1}. The results obtained from the subsequent search starting with the first long-term memory restart and the resulting M_LTM are shown in Table 5.7 and Table 5.8, respectively. As operation 2 of part 1 uses machine option 3 throughout the subsequent search, it is underlined to imply that the machine option 3 for operation 2 of part 1 is now fixed.

The results for the inside search starting with the second, third and fourth long-term memory restarts and their corresponding M_LTM frequency matrixes are shown in Table 5.9-5.13, respectively.

Table 5.7 Results obtained for the inside search of $\Omega p_0 = \{1,1,1,1\}$, starting with $\Omega m_0 = \{1,1,1,1 \mid 3,2,2,2 \mid 0,1,0,1\}$ as the first restart configuration.

# m_iter	Entries into ICL	Makespan (MS)	Entries into IIL
0	{1,1,1,1 | 3,2,2,2 | 0,1,0,1}**	24	{1,1,1,1 | 3,2,2,2 | 0,1,0,1}
1	{3,1,1,1 | 3,2,2,2 | 0,1,0,1}**	23	{3,1,1,1 | 3,2,2,2 | 0,1,0,1}
2	{3,3,1,1 | 3,2,2,2 | 0,1,0,1}	23	
3	{3,3,1,1 | 3,2,3,2 | 0,1,0,1}**	21	{3,3,1,1 | 3,2,3,2 | 0,1,0,1}
4	{1,3,1,1 | 3,2,3,2 | 0,1,0,1}	21	
5	{1,3,1,1 | 3,2,3,2 | 0,2,0,1}**	19	{1,3,1,1 | 3,2,3,2 | 0,2,0,1}
6	{1,3,3,1 | 3,2,3,2 | 0,2,0,1}	21	
7	{1,3,3,3 | 3,2,3,2 | 0,2,0,1}	23	
8	{1,3,3,3 | 3,2,3,2 | 0,2,0,2}	23	
9	{1,1,3,3 | 3,2,3,2 | 0,2,0,2}	23	

Table 5.8 The M_LTM frequency matrix for the inside search of $\Omega p_0 = \{1,1,1,1\}$, starting with $\Omega m_0 = \{1,1,1,1 \mid \underline{3},2,2,2 \mid 0,1,0,1\}$ as the first restart configuration.

Operation-Part.	Machine 1 (M1)	Machine 2 (M2)	Machine 3 (M3)
O1-P1	8	0	10
O1-P2	5	0	13
O1-P3	10	0	8
O1-P4	12	0	6
O2-P1	0	0	9
O2-P2	0	18	0
O2-P3	0	6	12
O2-P4	0	17	1
O3-P1	0	0	0
O3-P2	5	13	0
O3-P3	0	0	0
O3-P4	14	4	0

Table 5.9 Results obtained for the inside search of $\Omega p_0 = \{1,1,1,1\}$, starting from $\Omega m_0 = \{1,1,1,1 \mid 2,\underline{2},2,2 \mid 0,1,0,1\}$ as the second restart configuration.

# m_iter	Entries into ICL	Makespan (MS)	Entries into IIL
0	{1,1,1,1 | 2,2,2,2 | 0,1,0,1}**	34	{1,1,1,1 | 2,2,2,2 | 0,1,0,1}
1	{3,1,1,1 | 2,2,2,2 | 0,1,0,1}*	27	
2	{3,3,1,1 | 2,2,2,2 | 0,1,0,1}**	23	{3,3,1,1 | 2,2,2,2 | 0,1,0,1}
3	{3,3,3,1 | 2,2,2,2 | 0,1,0,1}	23	
4	{3,3,3,1 | 3,2,2,2 | 0,1,0,1}	23	
5	{3,3,3,1 | 3,2,2,2 | 0,2,0,1}**	21	{3,3,3,1 | 3,2,2,2 | 0,2,0,1}
6	{1,3,3,1 | 3,2,2,2 | 0,2,0,1}	21	
7	{1,3,3,1 | 3,2,2,2 | 0,2,0,2}	23	
8	{3,3,3,1 | 3,2,2,2 | 0,2,0,2}	23	
9	{3,1,3,1 | 3,2,2,2 | 0,2,0,2}	23	

Table 5.10 The M_LTM frequency matrix for the inside search of $\Omega p_0 = \{1,1,1,1\}$, starting with $\Omega m_0 = \{1,1,1,1 \mid 2,2,2,2 \mid 0,1,0,1\}$ as the second restart configuration.

Operation-Part.	Machine 1 (M1)	Machine 2 (M2)	Machine 3 (M3)
O1-P1	10	0	17
O1-P2	7	0	20
O1-P3	12	0	15
O1-P4	21	0	6
O2-P1	0	3	15
O2-P2	0	18	0
O2-P3	0	15	12
O2-P4	0	26	1
O3-P1	0	0	0
O3-P2	9	18	0
O3-P3	0	0	0
O3-P4	20	7	0

Table 5.11 Results obtained for the inside search of $\Omega p_0 = \{1,1,1,1\}$, starting with $\Omega m_0 = \{1,1,1,1 \mid 2,2,2,\underline{2} \mid 0,1,0,1\}$ as the third restart configuration.

# m_iter	Entries into ICL	Makespan (MS)	Entries into IIL
0	{1,1,1,1 | 2,2,2,2 | 0,1,0,1}**	34	{1,1,1,1 | 2,2,2,2 | 0,1,0,1}
1	{1,1,1,1 | 2,3,2,2 | 0,1,0,1}*	24	
2	{1,3,1,1 | 2,3,2,2 | 0,1,0,1}**	21	{1,3,1,1 | 2,3,2,2 | 0,1,0,1}
3	{3,3,1,1 | 2,3,2,2 | 0,1,0,1}	21	
4	{3,3,1,1 | 2,3,2,2 | 0,1,0,2}	21	
5	{1,3,1,1 | 2,3,2,2 | 0,1,0,2}	21	
6	{1,3,1,1 | 2,3,2,2 | 0,2,0,2}**	19	{1,3,1,1 | 2,3,2,2 | 0,2,0,2}
7	{1,3,3,1 | 2,3,2,2 | 0,2,0,2}	21	
8	{1,3,3,3 | 2,3,2,2 | 0,2,0,2}	23	
9	{1,1,3,3 | 2,3,2,2 | 0,2,0,2}**	22	{1,1,3,3 | 2,3,2,2 | 0,2,0,2}
10	{1,1,3,3 | 2,3,3,2 | 0,2,0,2}	23	
11	{1,1,3,3 | 2,2,3,2 | 0,2,0,2}	26	
12	{3,1,3,3 | 2,2,3,2 | 0,2,0,2}	33	
13	{3,1,3,3 | 3,2,3,2 | 0,2,0,2}*	23	
14	{3,1,1,3 | 3,2,3,2 | 0,2,0,2}**	21	{3,1,1,3 | 3,2,3,2 | 0,2,0,2}
15	{3,1,1,1 | 3,2,3,2 | 0,2,0,2}	21	
16	{3,1,1,1 | 3,2,3,2 | 0,2,0,1}	21	
17	{3,1,1,1 | 3,2,3,2 | 0,1,0,1}	23	
18	{3,1,1,1 | 3,3,3,2 | 0,1,0,1}	35	

Table 5.12 The M_LTM frequency matrix for the inside search of $\Omega p_0 = \{1,1,1,1\}$, starting with $\Omega m_0 = \{1,1,1,1 \mid 2,2,2,\underline{2} \mid 0,1,0,1\}$ as the third restart configuration.

Operation-Part.	Machine 1 (M1)	Machine 2 (M2)	Machine 3 (M3)
O1-P1	19	0	26
O1-P2	18	0	27
O1-P3	23	0	22
O1-P4	32	0	13
O2-P1	0	15	21
O2-P2	0	25	11
O2-P3	0	24	21
O2-P4	0	26	1
O3-P1	0	0	0
O3-P2	16	29	0
O3-P3	0	0	0
O3-P4	26	19	0

Table 5.13 Results obtained for the inside search of $\Omega p_0 = \{1,1,1,1\}$, starting with $\Omega m_0 = \{1,1,1,\underline{1} \mid 2,2,2,2 \mid 0,1,0,1\}$ as the fourth restart configuration.

# m_iter	Entries into ICL	Makespan (MS)	Entries into IIL
0	{1,1,1,1 | 2,2,2,2 | 0,1,0,1}**	34	{1,1,1,1 | 2,2,2,2 | 0,1,0,1}
1	{1,3,1,1 | 2,2,2,2 | 0,1,0,1}*	31	
2	{1,3,1,1 | 3,2,2,2 | 0,1,0,1}*	23	
3	{1,3,1,1 | 3,2,2,2 | 0,2,0,1}**	21	{1,3,1,1 | 3,2,2,2 | 0,2,0,1}
4	{1,3,1,1 | 3,2,2,3 | 0,2,0,1}	22	
5	{3,3,1,1 | 3,2,2,3 | 0,2,0,1}	22	
6	{3,3,1,1 | 3,2,3,3 | 0,2,0,1}	22	
7	{1,3,1,1 | 3,2,3,3 | 0,2,0,1}	22	

When the required number of restarts for the inside search has been reached, the optimal/near-optimal machine option configuration would be selected as the one with the smallest makespan among the best solutions obtained with each restart. Results presented in Table 5.14 show the best solutions obtained with each restart in this example.

Table 5.14 Summary of results obtained for the inside search of $\Omega p_0 = \{1,1,1,1\}$ with four long-term memory restarts.

Number of Restart	The Best Solution in the IIL	Resulting Makespan
Initial feasible configuration	{3,3,1,1 | 3,2,3,2 | 0,2,0,1}	19
First long-term memory restart	{1,3,1,1 | 3,2,3,2 | 0,2,0,1}	19
Second long-term memory restart	{3,3,3,1 | 3,2,2,2 | 0,2,0,1}	21
Third long-term memory restart	{1,3,1,1 | 2,3,2,2 | 0,2,0,2}	19
Fourth long-term memory restart	{1,3,1,1 | 3,2,2,2 | 0,2,0,1}	21

The optimal/near-optimal solution in this example is obtained from three different configurations giving the same makespan of 19 time units. They are {3,3,1,1 | 3,2,3,2 | 0,2,0,1}, {1,3,1,1 | 3,2,3,2 | 0,2,0,1}, and {1,3,1,1 | 2,3,2,2 | 0,2,0,2}. The long-term memory based on maximal frequency in this case did not improve the best solution obtained from the initial feasible solution. However, it did find another solution with the same makespan. Thus, the optimal/near-optimal machine option solution for $\Omega p_0 = \{1,1,1,1\}$ is the first configuration (i.e., {3,3,1,1 | 3,2,3,2 | 0,2,0,1}) with a makespan of 19 time units.

Step 9: Repeat steps 3 through 8 for each configuration in the outside neighborhoods obtained in step 2 (i.e., $Np(\Omega p_0)$ = {2,1,1,1}, {1,2,1,1}, {1,1,2,1}, {1,1,3,1}, and {1,1,1,2}).

The results for the inside search for each process planning configuration in the neighborhoods of {1,1,1,1} is shown in Table 5.15.

Table 5.15 Results obtained for the inside search of each process planning configuration in $Np(\Omega p_0)$.

The Process Planning Configurations in the Neighborhood of {1,1,1,1}	The Optimal/Near-Optimal Machine Option Configurations for the Inside Search	Resulting Makespan
{2,1,1,1}	{0,1,1,1 | 3,2,2,2 | 1,2,0,1}	21
{1,2,1,1}	{3,0,1,1 | 2,3,2,2 | 0,1,0,1}	12
{1,1,2,1}	{3,3,0,1 | 3,2,2,2 | 0,2,1,1}	19
{1,1,3,1}	{3,1,1,1 | 3,2,3,2 | 0,2,1,2}	22
{1,1,1,2}	{3,3,1,1 | 3,2,2,2 | 0,1,0,0}	22

Step 10: Similar to step 6 of the inside search, the move for outside search (p_move) is now performed. A p_move transforms a sequence of process plans into another sequence of process plans in $Np(\Omega p)$. The value of move is the same of that for the inside search (Value of move = MS after performing the move - MS before performing the move).

In this example, the makespan for the initial feasible process planning configuration is 19 time units. The next configuration selected is {1,2,1,1} as it has the smallest value of move (-7) among the configurations in the neighborhoods.

Similar to the inside search, the following parameters for the outside tabu search are updated after performing an outside move.

(1) Outside-tabu list (p_tabu list)

In this example, a p_move is performed to move the initial feasible process planning configuration {1,1,1,1} to the next configuration {1,2,1,1}. The part number as well as the process plan that was moved would be stored as the first element in the outside-tabu list (p_tabu list).

- p_tabu list = {pp_2 (1)}

The interpretation of the p_tabu list is that process plan 1 was selected for part 2 in the most recent iteration and it has been moved to the alternative process plan 2. The "(1)" that comes after pp_2 represents process plan 1 which is the process plan used by part 2 prior to performing the move.

As for the inside search, the outside aspiration level (outside_AL) is updated if the makespan evaluated for the current solution is found to be better than the best solution found so far. This being the case, the outside_AL is updated as per the better makespan of the new solution.

- outside_AL = 12

In addition, the tabu-list size for this example can be determined as follows.

The fixed size of p_tabu list = $\lceil (5/2) \rceil = 3$

The variable sizes of p_tabu list:

The initial size of p_tabu list = fixed size = $\lceil (5/2) \rceil = 3$

The decreased size of p_tabu list = $\lceil (5/3) \rceil = 2$

The increased size of p_tabu list = 5-1 = 4

(2) Outside candidate list (OCL) and outside index list (OIL)

Similar to the inside search, the initial feasible process planning configuration is admitted into both OCL and OIL. The next configuration obtained in this example is also admitted into OCL as it is chosen to perform future perturbations. The next configuration has a better makespan (12) than that of the initial configuration (19). Thus, it is also given a star as it has the potential of becoming the next local optimal.

- OCL = { {1,1,1,1}

{1,2,1,1}* }.

- OIL = { {1,1,1,1} }.

(3) Number of iterations without improvement for outside search

Similar to the inside search, every time a p_move is performed, the number of iterations (p_iter) is increased by one. If there is no improvement in the makespan relative to the makespan of the most recent configuration, the number of iterations without improvement is increased by one. However, if in any iteration there is an improvement in

makespan, the number of iterations without improvement will be reinitialized to zero. For this example, there is an improvement in makespan (from 19 to 12), therefore, the number of iterations without improvement is not increased and it still has the value of zero.

- p_iter = 1.
- p_iter_without_improvement = 0.

(4) Outside long-term memory (P_LTM)

The outside long-term memory (P_LTM) is comparable to the inside long-term memory (M_LTM). The P_LTM. frequency matrix keeps track of the tenure of process plan for each part throughout the outside search. Every time a new process planning configuration is constructed, the entries in the frequency matrix corresponding to the parts and their respective process plans in the configuration are increased by one.

Initially, all entries in the P_LTM frequency matrix is initialized to zero. After the move from the initial process planning configuration {1,1,1,1} to the next process planning configuration {1,2,1,1} is performed, the P_LTM would be updated as shown in Table 5.16.

Table 5.16 Updated P_LTM frequency matrix for the process planning configuration {1,2,1,1}.

P_LTM	Process Plan 1	Process Plan 2	Process Plan 3
Part 1	1	0	-
Part 2	0	1	-
Part 3	1	0	0
Part 4	1	0	-

Step 11: To terminate the outside search, the number of iterations without improvement for this example can be determined as follows:

- For the fixed tabu list, the stopping criterion is evaluated as:

The number of iterations without improvement for outside search (OIT)

= int [(9*3)/(2*3)] = 4

- For the variable tabu list, the stopping criteria are evaluated as:

(i) If there is no improvement in the last [int (OIT/3)] iteration (i.e., 1 iteration) with the initial p_tabu list size, decrease the p_tabu list size to 2 as evaluated in step 10.

(ii) If there is no improvement in the last 1 iteration with the decreased p_tabu list size, increase the p_tabu list size to 4 as evaluated in step 10.

(iii) If there is no improvement in the last 1 iteration with the increased p_tabu list size, then stop performing the move and start to diversify the outside search.

The results of the outside search of the fixed tabu-list size using $\Omega p_0 = \{1,1,1,1\}$ as an initial process planning configuration are shown in Table 5.17.

Table 5.17 Results obtained for the outside search starting with $\Omega p_0 = \{1,1,1,1\}$ as an initial process planning configuration.

# p_iter	Entries into OCL	Makespan (MS)	Entries into OIL
0	{1,1,1,1}**	19	{1,1,1,1}
1	{1,2,1,1}**	12	{1,2,1,1}
2	{1,2,2,1}	13	
3	{2,2,2,1}	17	
4	{2,2,2,2}	20	
5	{2,1,2,2}	25	

From Table 5.17, the outside search starting with {1,1,1,1} is terminated after 5 moves have been performed since the number of iterations without improvement for the fixed tabu-list size (4 in this example) has been reached. The best solution for the search starting with {1,1,1,1} is the configuration in the OIL with the smallest makespan of 12 time units given by {1,2,1,1}.

Step 12: The outside search is now diversified using a new restart, identified based on the outside long-term memory frequency matrix (P_LTM).

The resulting P_LTM frequency matrix for the outside search at the end of step 11 is shown in Table 5.18.

Table 5.18 The P_LTM frequency matrix for the outside search starting with $\Omega p_0 = \{1,1,1,1\}$ as an initial process planning configuration.

P_LTM	Process Plan 1	Process Plan 2	Process Plan 3
Part 1	2	3	-
Part 2	1	4	-
Part 3	1	4	0
Part 4	3	2	-

Using the long-term memory based on the maximal frequency (LTM_MAX), the first new restart is invoked by taking the maximal entry in the P_LTM matrix. The part corresponding to the maximal entry is then fixed to its respective process plan during the subsequent search.

The maximal entry in this example is 4 which corresponds to using process plan 2 for part 2. Thus, the first new restart based on maximal frequency is {1,2,1,1}. The results obtained for the subsequent search starting with the first long-term memory restart and the corresponding P_LTM are shown in Table 5.19 and Table 5.20, respectively. As part 2 is processed according to process plan 2 throughout the subsequent search, it is now underlined to imply that process plan 2 of part 2 is now fixed. Notice that, although the number of iterations without improvement is set equal to 4, the search is terminated at the end of the sixth iteration which is only one iteration after the fifth iteration without any improvement. This is due to one of two reasons. First, every new process planning configuration is checked against those so far considered to avoid duplication. Second, a

process plan corresponding to a part could be on the p_tabu list, thus preventing from considering a process planning configuration. The tabu status given for a process plan corresponding to a part would be overridden if the new process planning configuration including the tabu process plan for the part resulted in a better makespan than the current best makespan.

Table 5.19 Results obtained for the outside search starting with $\Omega p_0 = \{1,\underline{2},1,1\}$ as the first restart configuration.

# p_iter	Entries into OCL	Makespan (MS)	Entries into OIL
0	{1,2,1,1}**	12	{1,2,1,1}
1	{2,2,1,1}	15	
2	{2,2,3,1}	21	
3	{1,2,3,1}**	19	{1,2,1,1}
4	{1,2,3,2}	24	
5	{1,2,1,2}**	18	{1,2,1,2}
6	{1,2,2,2}	18	

Table 5.20 The P_LTM frequency matrix for the outside search starting with $\Omega p_0 = \{1,\underline{2},1,1\}$ as the first restart configuration.

P_LTM	Process Plan 1	Process Plan 2	Process Plan 3
Part 1	6	5	-
Part 2	1	4	-
Part 3	3	5	3
Part 4	6	5	-

The results for the inside search starting with second long-term memory restart and the summarized results with two restarts are shown in Table 5.21 and 5.22, respectively.

Table 5.21 Results obtained for the outside search starting with $\Omega p_0 = \{\underline{1},1,1,1\}$ as the second restart configuration.

# p_iter	Entries into OCL	Makespan (MS)	Entries into OIL
0	{1,1,1,1}**	19	{1,1,1,1}
1	{1,1,2,1}	19	
2	{1,1,3,1}	22	
3	{1,1,3,2}	26	
4	{1,1,1,2}**	22	{1,1,1,2}
5	{1,1,2,2}	22	

Table 5.22 Results obtained for the outside search with two long-term memory restarts.

Number of Restart	The Best Solution in the OIL	Resulting Makespan
Initial feasible configuration	{1,2,1,1}	12
First long-term memory restart	{1,2,1,1}	12
Second long-term memory restart	{1,1,2,1}	19

The entire search is finally terminated as the required number of restarts for the outside search has been reached. In summary, the optimal/near-optimal solution given by {1,2,1,1} and {3,0,1,1 | 2,3,2,2 | 0,1,0,1} with a minimum makespan of 12 time units is obtained with all six test heuristics (TSH 1-TSH 6) with CPU time of 2.400, 4.670, 5.050, 2.300, 2.590, 2.640 seconds, respectively. The above solution presented on a time-scaled Gantt chart is shown in Figure 5.1.

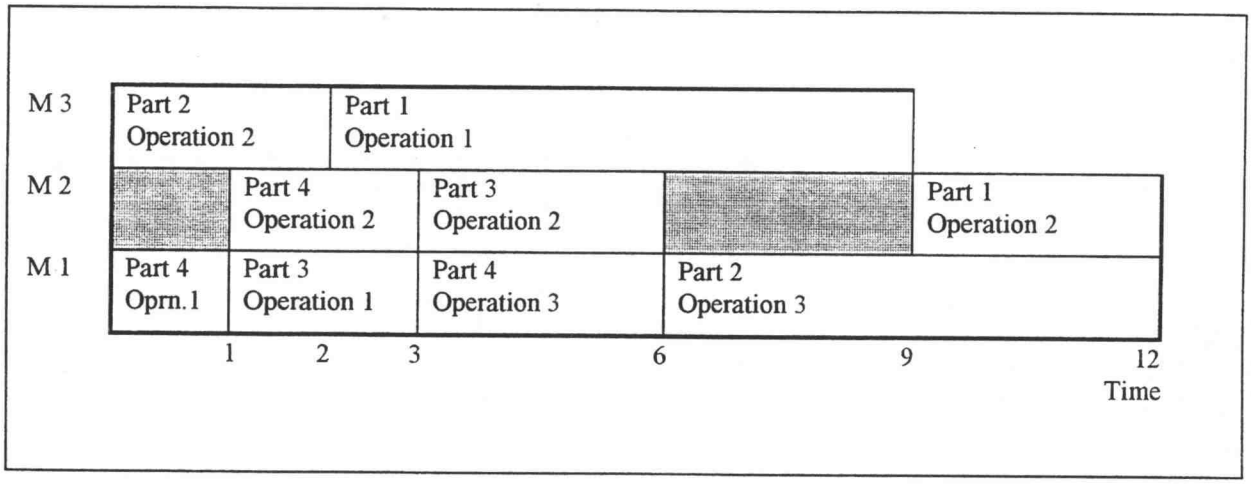


Figure 5.1 Gantt chart for the example problem.

6. EXPERIMENTAL DESIGN

6.1 Determination of the Optimal Makespan

The optimal/near-optimal makespan evaluated for the example problem, consisting 4 parts, 3 machines and 3 operations with each of the six heuristics is 12 time units. Yet, how good or bad the solution is, as compared to the global optimal solution cannot be assessed unless the optimal solution for the same problem is determined. An attempt has been made to determine the optimal solution for the example problem and compare it with the solution obtained from the heuristics. The model when formulated had 304 variables and 590 constraints. Of the 304 variables, 268 variables are binary integer variables. However, the maximum number of constraints that can be handled by SuperLINDO (1989) computer software is only 250 constraints. As the example problem exceeds the limit set on the number of constraints, a smaller problem, consisting of 3 parts, 3 machines and 3 operations, is constructed in order that the number of constraints and variables introduced in the formulation of that problem will be within the limits set in the SuperLINDO (1989) computer software.

The small problem is constructed as follows. The number of process plans per part and the number of operations required to be performed on each part is predetermined. Either two or three operations are allowed for each process plan in the small problem. To determine which operations should be performed for a process plan with two operations, two different uniformly distributed random numbers in [0,1] are generated and used. If the random number generated is between 0.0-0.329999 the first operation is selected, if it is between 0.33-0.659999 the second operation is selected, and finally, if it is between 0.66-1.0 the third operation is selected. If two random numbers generated in succession fell into the same range, the second random number is discarded and a new random number is generated. The reason is that a meaningful problem should not have the same operation performed twice. Additionally, if two process plans required the same set of operations be performed on the part, the random number corresponding to the last operation of the second process plan is discarded and a new random number is generated.

The processing time per operation is randomly generated from an uniform distribution described between 10 and 20 time units.

The data pertaining to the small problem is presented in Table 6.1-6.3. The mathematical model formulated is presented in Appendix A. The formulation for this small problem consists of 243 constraints and 133 variables. Of the 133 variables, 106 variables are binary integer variables that can take on the value of either 0 or 1. The model is solved for the optimal solution using the branch-and-bound enumeration technique incorporated in SuperLINDO (1989) software.

Table 6.1 Data indicating alternative process plans for the small problem.

Operation	j = 1		j = 2		j = 3	
	p = 1	p = 2	p = 1	p = 2	p = 1	p = 2
k =1	1		1		1	1
k =2	1	1	1	1		1
k =3		1	1	1	1	

Table 6.2 Data indicating alternative machine options for the small problem.

	Whether k can be Performed on Following Machines		
	i = 1	i = 2	i = 3
k = 1	1	1	
k = 2	1		1
k = 3		1	1

Table 6.3 Processing times for the small problem.

	j = 1		j = 2		j = 3	
	p = 1	p = 2	p = 1	p = 2	p = 1	p = 2
k = 1, i = 1	10		19		11	11
k = 1, i = 2	16		18		11	10
k = 2, i = 1	12	16	11	13		10
k = 2, i = 3	18	15	19	10		14
k = 3, i = 2		13	17	11	20	
k = 3, i = 3		19	15	14	14	

An integer solution of 56 was found after 488 iterations. The number of iterations in SuperLINDO (1989) is equal to the number of steps required to solve the subproblems obtained from performing the LP-relaxation of the original problem (i.e., the same problem except that a set of integer restrictions is disregarded). In SuperLINDO (1989) computer software, the simplex method is used to solve each of the LP-relaxation subproblems. Thus, the number of iterations in this computer software is also equal to the number of pivots used in the simplex method. The number of pivots in the simplex method represents the number of moves required to move a current feasible solution to a better adjacent basic feasible solution in solving the LP problem. For further details in the branch-and-bound technique and the simplex method, the reader is advised to refer to the text by Hillier and Lieberman (1990).

No better solution was identified even when the default limit of 52,378 was reached for the number of pivots. The user is then asked to specify the additional number of pivots. Although an additional of 1,000,000 pivots were specified, SuperLINDO (1989) could not identify a better solution within a reasonable time. Interestingly, the run time on a Pentium 75 MHz machine with 16MB RAM for performing the additional 1,000,000 pivots is well over 2 hours.

The problem, although small, involves 106 binary integer variables. It means the number of explicit solutions that can be enumerated is $2^{106} = 8.11*10^{31}$. This partly

explains the difficulty encountered by SuperLINDO (1989) in identifying an optimal solution, although it uses the branch-and-bound technique which is an implicit enumeration algorithm for solving combinatorial optimization problems.

The solution obtained from the heuristics (TSH 1-TSH 6) consists of the process planning configuration and the machine option configuration. The unique process plan for each part is {1,2,2}, meaning that process plan 1 is selected for part 1 and process plan 2 is selected for parts 2 and 3. The operation assignment is {1,0,2 | 1,3,3 | 0,2,0}, meaning that operations 1 and 2 of part 1 are performed on machine 1 according to process plan 1, operations 2 and 3 of part 2 are performed on machines 3 and 2, respectively, according to process plan 2, and finally, operations 1 and 2 of part 3 are performed on machines 2 and 3, respectively, according to process plan 2. The minimum makespan determined from all six heuristics is 24 time units. The CPU time for determining the optimal/near-optimal solution for the small problem with each of the six heuristics is 1.180, 1.260, 1.270, 1.170, 1.230, and 1.280 seconds, respectively. The Gantt chart for the above solution is presented in Figure 6.1.

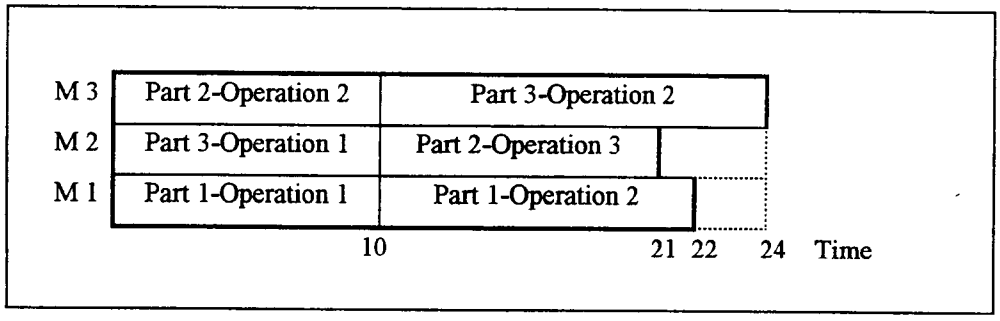


Figure 6.1 Gantt chart for the small problem.

This best makespan from the heuristics was also used as an upper bound for aiding SuperLINDO (1989) to examine other solutions. When SuperLINDO (1989) is aided by an upper bound, it is supposed to identify solutions better than the upper bound and disregard all those that are inferior. Ironically, when an upper bound of 24 was specified for this small problem, SuperLINDO (1989) could not identify a better makespan than 24.

The same situation prevailed even after an additional 1,000,000 pivots were specified with SuperLINDO (1989). The fact that SuperLINDO (1989) has failed to identify a better makespan than that was determined by all six heuristics, much less an optimal makespan for the small problem, further reinforces the fact that there is clearly a need for efficient heuristics that can identify near-optimal solutions for complex problems on scheduling of FMSs.

The small problem is structured in a manner that the processing time required of each operation of a part lies anywhere between 10 and 20 time units. The minimum number of operations required to completely process any of the three parts is 2. The optimal makespan is the minimum time required to complete the operations required of all three parts. Evidently, the optimal makespan must also be greater than or equal to the minimum of the minimum time required to completely process each of the three parts independently. From an observation of the processing times given in Table 6.3, it is easy to see that the minimum of 20 time units is obtained when operations 1 and 2 of part 3 are performed on machine 2 and 1, respectively, according to process plan 2. That is, the lower bound for the optimal makespan of the small problem is 20 time units. Thus, the best solution (24 time units) has a deviation of 20% ((24-20)/20*100) from the known lower bound. The best integer solution obtained from the SuperLINDO (1989) (56 time units) is much higher than the best solution obtained from the heuristics (24 time units). This clearly demonstrates that the implicit enumeration method (i.e., branch-and-bound) used in SuperLINDO (1989) has indeed failed to identify even a "good" (if not optimal) solution for the small problem considered in this research.

6.2 Comparison of Tabu search-based Heuristics

The comparison of six different tabu search-based heuristics is performed to examine the effect of (i) using fixed versus variable tabu-list sizes, and (ii) using long-term memory versus not using long-term memory. The characteristics of six different tabu search-based heuristics can be described as:

TSH 1: The tabu-search based heuristic with fixed tabu-list size and no long-term memory

TSH 2: The tabu-search based heuristic with fixed tabu-list size and LTM_MIN

TSH 3: The tabu-search based heuristic with fixed tabu-list size and LTM_MAX

TSH 4: The tabu-search based heuristic with variable tabu-list sizes and no long-term memory

TSH 5: The tabu-search based heuristic with variable tabu-list sizes and LTM_MIN

TSH 6: The tabu-search based heuristic with variable tabu-list sizes and LTM_MAX

A single-factor experiment is performed to compare the performance of the six different tabu search-based heuristics. The factor in this case is characterized by each of the six heuristics and measured by the best makespan evaluated. The experiment with a single factor of interest can be performed either as a completely randomized design or a randomized complete block design. For example, to perform a completely randomized design at six different levels of the factor (six different heuristics) and 10 replicates (sample size of 10), 60 different test problems of the appropriate structure would have to be generated and each heuristic tested with 10 different problems. As each heuristic is tested with different test problems, if a lesser makespan is evaluated for a heuristic, it cannot be entirely attributed to the performance of that heuristic as the decision may have been partially effected by the difference between the structure of the test problems.

Instead, the experiment is conducted as a randomized complete block design using the test problem as a block. The reason for conducting a randomized complete block design is that it eliminates the influence of differences in structure of the test problems which can contribute to identifying a difference in the performance of the heuristics. Should a difference in performance of the heuristics is identified when the experiment is conducted as a randomized complete block design, then that difference can be wholly

attributed to the difference in performance of each heuristic and not the difference between test problems. A block (sample) size of 10, representing 10 different test problems, is used to test each of the six heuristics. For further details on completely randomized and randomized block designs, the reader is advised to refer to the text by Montgomery (1991).

Four different problem structures are tested. The problem structure is defined by the number of parts, the number of alternative process plans for each part, the number of operations required of each part, and the number of alternative machine options for each operation. The first problem structure is similar to the example problem considered in the heuristic algorithm section, which consisted of four parts (P), three machines (M) and three operations (O), denoted by 4P*3M*3O. The second problem structure is denoted by 5P*4M*4O, while the third and fourth structures are denoted by 10P*5M*5O and 14P*7M*70, respectively. The data structure for the second, third and fourth problem structures are shown in Table B.1-B.3 (Appendix B), respectively. Additionally, the parameters used in the tabu search-based heuristics for each problem structure are given in Table B.4 (Appendix B).

The number of operations for each part is varied from one to three operations per part. Integer values representing the processing time required of each part order is generated randomly from a uniform distribution between 10 and 20 time units. Each operation has only one alternative machine option which means it can be processed on a different machine. If an operation has an alternative machine, it is assumed that the processing time on the alternative machine may be shorter, equal or longer than the processing time on the original machine. This assumption also allows sequential operations to be processed on the same machine.

For each problem structure, 10 different blocks (test problems) are generated and the minimum makespan for each block with each of the six heuristics is determined. An analysis of variance is then performed to determine if the average makespan obtained for those 10 problems is significantly different among the six heuristics. In this experiment, the significance level α, also referred to as type I error, is assumed equal to 5% ($\alpha = 0.05$). If a difference in the average makespans is found, a Least Significance Difference (LSD) test

is then performed to identify which heuristics contributed to the difference. Although Duncan's Multiple Range, Newman Keul's and Tukey's tests may be used for this purpose, LSD is selected as it is readily available in STATGRAPHICS (1988), a computerized statistical programming package developed by the Statistical Graphics Corporation. The normality assumption for performing an analysis of variance is also checked using the normal probability plot of the residuals for each problem structure. There is no severe indication of nonnormality shown in all problem structures. The normal probability plots for each problem structure are in Figure C.1-C.4 (Appendix C). The algorithmic steps were coded using Microsoft QuickC and run on a Pentium 75 MHz machine with 16 MB RAM.

7. RESULTS AND DISCUSSIONS

The experimental results for each test problem obtained with each heuristic along with the CPU time are presented in Table D.1-D.4 (Appendix D), for the 4P*3M*3O, 5P*4M*4O, 10P*5M*5O and 14P*7M*7O problem structures, respectively. The analysis of variance along with the LSD analysis for each problem structure is also presented in Table E.1-E.4 (Appendix E).

The summary of results for the average makespan along with the LSD analysis for each problem structure are shown in Table 7.1. It is also helpful to summarize the results obtained from the LSD analysis in terms of the homogeneous groups for each problem structure as shown in Table 7.2-7.5. The "X" sign shown in the tables denotes the heuristics that do not differ significantly based on the LSD analysis.

From Table 7.2, for the 4P*3M*3O problem structure, TSH 2, 3, 5 and 6 have determined the same minimum average makespan of 43.6 time units while both TSH 1 and 4 have determined the minimum average makespan of 44 time units. Although there is no significant difference between TSH 2, 3, 5, and 6 at $\alpha = 0.05$, they have evaluated a lesser average makespan than TSH 1 and 4.

For the 5P*4M*4O problem structure, TSH 3 and 6 have determined the minimum average makespan of 43.7 time units, TSH 2 and 5 have determined the minimum average makespan of 44 time units, and TSH 1 and 4 have determined the minimum average makespan of 44.3 and 44.8 time units, respectively. For the same problem structure, at $\alpha = 0.05$, TSH 4 is found to be significantly inferior to TSH 2, 3, 5 and 6 as shown in Table 7.3. Additionally, it can be noted that TSH 1 and TSH 4 have determined a larger average makespan than the other heuristics which is consistent with the results obtained for the previous problem structure.

From Table 7.4, for the 10P*5M*5O problem structure, TSH 3 has evaluated a minimum average makespan of 56.2 time units while TSH 5, 2 and 6 have evaluated a larger average makespan of 56.3, 56.5, 56.9 time units, respectively. Both TSH 1 and 4 have evaluated a minimum average makespan of 57 time units. The LSD test shows that

Table 7.1 Summary of results obtained for the comparison of TSH 1-TSH 6.

Average. Makespan (MS) with Number of Blocks = 10	Problem Structures			
	4P*3M*3O	5P*4M*4O	10P*5M*5O	14P*7M*7O
TSH 1	44.0	44.3	57	69.6
TSH 2	43.6	44.0	56.5	68.7
TSH 3	43.6	43.7	56.2	68.5
TSH 4	44.0	44.8	57	69.9
TSH 5	43.6	44.0	56.3	68.6
TSH 6	43.6	43.7	56.9	68.4
Is MS Significantly Different Between Heuristics at $\alpha = 0.05$?	No	Yes	Yes	Yes
TSH 1 & TSH 2	-	No	No	Yes
TSH 1 & TSH 3	-	No	Yes	Yes
TSH 1 & TSH 4	-	No	No	No
TSH 1 & TSH 5	-	No	Yes	Yes
TSH 1 & TSH 6	-	No	No	Yes
TSH 2 & TSH 3	-	No	No	No
TSH 2 & TSH 4	-	Yes	No	Yes
TSH 2 & TSH 5	-	No	No	No
TSH 2 & TSH 6	-	No	No	No
TSH 3 & TSH 4	-	Yes	Yes	Yes
TSH 3 & TSH 5	-	No	No	No
TSH 3 & TSH 6	-	No	Yes	No
TSH 4 & TSH 5	-	Yes	Yes	Yes
TSH 4 & TSH 6	-	Yes	No	Yes
TSH 5 & TSH 6	-	No	Yes	No

Table 7.2 Results obtained for the LSD analysis of 4P*3M*3O problem structure.

Heuristics	Average Makespan	Homogeneous Groups
TSH 2	43.6	X
TSH 3	43.6	X
TSH 5	43.6	X
TSH 6	43.6	X
TSH 1	44.0	X
TSH 4	44.0	X

Table 7.3 Results obtained for the LSD analysis of 5P*4M*4O problem structure.

Heuristics	Average Makespan	Homogeneous Groups	
TSH 3	43.7	X	
TSH 6	43.7	X	
TSH 2	44.0	X	
TSH 5	44.0	X	
TSH 1	44.3	X	X
TSH 4	44.8		X

Table 7.4 Results obtained for the LSD analysis of 10P*5M*5O problem structure.

Heuristics	Average Makespan	Homogeneous Groups	
TSH 3	56.2	X	
TSH 5	56.3	X	
TSH 2	56.5	X	X
TSH 6	56.9		X
TSH 1	57.0		X
TSH 4	57.0		X

Table 7.5 Results obtained for the LSD analysis of 14P*7M*7O problem structure.

Heuristics	Average Makespan	Homogeneous Groups	
TSH 6	68.4	X	
TSH 3	68.5	X	
TSH 5	68.6	X	
TSH 2	68.7	X	
TSH 1	69.6		X
TSH 4	69.9		X

at $\alpha = 0.05$ the performance of TSH 1, 4 and 6 is a significantly inferior to the rest of the heuristics. TSH 1 and 4 have consistently determined a significantly larger makespan than TSH 3 and 5 as in the previous problem structure. Interestingly, the minimum average makespan determined by TSH 3 is significantly better than TSH 6 in this problem structure.

For the 14P*7M*7O problem structure in Table 7.5, TSH 6 has evaluated the best average minimum makespan of 68.4 time units. However, no significant difference between TSH 6 and TSH 3, 5 and 2 is found. Both TSH 1 and TSH 4 have again

determined a significantly larger makespan of 69.6 and 69.9 time units, respectively, than the rest of the heuristics.

(i) The use of long-term memory in tabu-search based heuristics.

Based on the results obtained, TSH 2, 3, 5 and 6 which use the long-term memory have consistently determined a better minimum average makespan than TSH 1 and 4 which did not use the long-term memory. This is true on all problem structures. Notice that for the smallest problem structure (4P*3M*3O), although the difference is not significant in a statistical sense, TSH 2, 3, 5 and 6 have determined a lesser minimum average makespan than TSH 1 and 4. When the size of the problems becomes larger, like in the 5P*4M*4O problem structure, the difference in performance is more pronounced as indicated by the significant difference between TSH 4 and TSH 2, 3, 5 and 6. As the problem size further increases like in 10P*5M*5O and 14P*7M*7O, the performance of TSH 2, 3, 5 and 6 which reflect the use of long-term memory is shown to be significantly superior to both TSH 1 and 4 which do not use long-term memory. Thus, it can be concluded that the improvement in solution quality obtained by the use of long-term memory is more pronounced as the size of the problems becomes larger.

For the comparison of the use of long-term memory based on maximal frequency (LTM_MAX) with the use of long-term memory based on minimal frequency (LTM_MIN), the heuristics using LTM_MAX (TSH 3 and 6) and the heuristics using LTM_MIN (TSH 2 and 5) have determined the same average makespan for the 4P*3M*3O problem structure. For the 5P*4M*4O and 14P*7M*7O, the heuristics using LTM_MAX have determined a lesser average makespan than the heuristics using LTM_MIN. For the 10P*5M*5O, the heuristic using LTM_MAX with the fixed tabu-list size (TSH 3) has consistently determined a lesser average makespan than the heuristics using LTM_MIN (TSH 2 and 5). However, the heuristic using LTM_MAX with the variable tabu-list sizes (TSH 6) has determined a larger average makespan than the heuristic using LTM_MIN (TSH 2 and 5). Thus, in most cases, the use of long-term memory based on maximal frequency (LTM_MAX) is more efficient than the use of long-term memory based on minimal frequency (LTM_MIN).

Among the heuristics using LTM_MAX, the heuristic with LTM_MAX and fixed tabu-list size (TSH 3) has determined a lesser average makespan than the one with variable tabu-list sizes (TSH 6). Similarly, the heuristic with LTM_MIN and fixed tabu-list size (TSH 5) has outperformed the heuristic with LTM_MIN and variable tabu-list sizes (TSH 2). Even though the use of LTM_MIN is improved by incorporating fixed tabu-list size, its performance as a whole is outperformed by the use of LTM_MAX.

(ii) The use of fixed versus variable tabu-list sizes.

The TSH 1, TSH 2, and TSH 3 employed the fixed tabu-list size while TSH 4, TSH 5, and TSH 6 employed the variable tabu-list sizes. As the performance of TSH 3 and TSH 6 is closely comparable, it is hard to recommend the use of fixed tabu-list size over the variable tabu-list sizes or vice versa. However, in the 10P*5M*5O problem structure, TSH 3 has outperformed TSH 6. For the problem structures tested in this research, the use of fixed tabu-list size is generally preferred over the variable tabu-list sizes.

In conclusion, it can be stated that in every problem structure, TSH 1 characterized by a tabu search-based heuristic with fixed tabu-list size and no long-term memory and TSH 4 characterized by a tabu search-based heuristic with variable tabu-list sizes and no long-term memory are clearly inferior to the rest of the heuristics (TSH 2, 3, 5 and 6). Furthermore, TSH 3 representing the tabu search-based heuristic with LTM_MAX has consistently outperformed the other heuristics in terms of average minimum makespan. TSH 3 is, therefore, recommended for solving the scheduling problem considered in this research.

8. CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

The problem of scheduling parts in Flexible Manufacturing Systems (FMSs) is investigated when alternative process plans as well as alternative machine options are present. The problem is formulated as a mixed-(binary) integer linear programming model and is proven to be NP-hard in the strong sense. This rules out the possibility of employing an implicit enumeration-based algorithm such as the branch-and-bound technique because such algorithms would turn out to be too time consuming even for a moderately-sized problems. Tabu search-based heuristics are proposed to efficiently solve large-sized problems within a reasonable computational time. This fact has been further demonstrated by attempting to solve a small problem (3-part, 3-machine, and 3-operation problem) using the implicit enumeration technique (i.e., branch-and-bound) incorporated in SuperLINDO (1989) computer software. The solution obtained from SuperLINDO (1989) (56 time units) is much higher than both the best solution obtained from the heuristics (24 time units) and the lower bound determined for that problem (20 time units). This clearly shows that there is indeed a need for efficient heuristics to solve complex problems in scheduling of FMSs.

Six different tabu search-based heuristics are tested on four different problem structures. An extensive statistical analysis based on a randomized-block design has been performed to compare the performances of six different heuristics (TSH 1-TSH 6) using makespan as the criteria. The tabu search-based heuristic with fixed tabu-list size and long-term memory based on maximal frequency (TSH 3) shows a superior performance over the other heuristics on almost every problem structure attempted. In general, the use of long-term memory based on maximal frequency and fixed tabu-list size is preferred to solve problems in scheduling of parts in FMSs.

The above research can be extended to investigate the scheduling of FMSs in a dynamic scheduling environment. In a dynamic scheduling environment, according to Rachamadugu and Stecke (1987), the order status of parts change continually. That is, parts are introduced into the FMS continuously as orders arrive. New order of parts that come in can have alternative process plans and they need to compete with the orders of

parts that are already scheduled. The original schedule would need to be rescheduled which can impact the performance of the heuristic algorithms when used in a real-time mode.

Another area of extension of this research is to include system disruptions such as machine breakdowns or tool breakages into the scheduling environment. When machine breakdowns or tool breakages are allowed, rescheduling is required due to delays caused to the schedule. The heuristic algorithms applied in a real-time scheduling environment should be modified to include the effects of these delays.

BIBLIOGRAPHY

Barnes, J.W., and Laguna, M., 1993, Solving the multiple-machine weighted flow time problem using tabu search. *IIE Transactions*, 25:121-128.

Chang, Y.L. and Sullivan, R.S., 1984a, Real-time scheduling of flexible manufacturing systems-A conceptual and mathematical foundation. Working paper 83/84-4-36, University of Texas at Austin, TX, USA., referenced by Chang et al., 1989.

Chang, Y.L. and Sullivan, R.S., 1984b, Schedule generation in a dynamic job shop with parallel workstation. Working paper 83/84-4-24, University of Texas at Austin, TX, USA., referenced by Chang et al., 1989.

Chang, Y.L., Sullivan, R., Bagchi, U., and Wilson, J., 1985, Experimental investigation of real-time scheduling in flexible manufacturing systems. *Annals of Operation Research*, 3:355-378, reference by Hutchison et al., .1991)

Chang, Y., Matsuo, H., Sullivan, R.S.,1989, A bottleneck-based beam search for scheduling in a flexible manufacturing system. *International Journal of Production Research*, 27:1949-1961.

Choobineh, F., 1988, A framework for the design of cellular manufacturing systems, *International Journal of Production Research*, 26:1161-1172, referenced by Logendran et al., 1994.

De Meyer, A., Nakane, J., Miller, J.G., and Ferdows, K., 1989, Flexibility: the next competitive battle the manufacturing futures survey. *Strategic Management Journal*, 10:135-144.

Denzler, D.R., and Boe, W. J., 1987, Experimental investigation of flexible manufacturing system scheduling decision rules. *International Journal of Production Research*, 25:979-994.

Eck, B.T., 1989, Good solutions to job shop scheduling problems via tabu search, *Research report*, Columbia University, NY, USA., referenced by Glover, 1990.

Garey, M.R., Johnson, D.S., and Sethi, R., 1976, The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2), 117-129.

Garey, M.R., and Johnson, D.S., 1979, *Computers and Intractability : A Guide to the Theory of NP-Completeness* (New York: Freeman).

Glover, F., 1986, Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13:533-549.

Glover, F., 1989, Tabu search-Part I. *ORSA Journal on Computing*, 1:190-206.

Glover, F., and Laguna, M., 1989, Target analysis to improve a tabu search method for machine scheduling, *Technical report*, US West Advanced Technologies, Boulder, Colorado, USA., referenced by Glover, 1990.

Glover, F., 1990a, Tabu search-Part II. *ORSA Journal on Computing*, 1:4-32.

Glover, F., 1990b, Tabu search: A tutorial. *Interfaces*, 20:74-94.

Groover, M.P., and Zimmers, E.W.Jr., 1984, CAD/CAM: *Computer Aided Design and Manufacturing* (New Jersey: Prentice Hall), referenced by Mukhopadhyay et al., 1992.

Hillier and, F.S., and Lieberman,G.J., 1990, *Introductions to Operations Research*, 5th ed. (New York: McGraw-Hill)

Hutchison, J., 1988, Scheduling random job shop flexible manufacturing systems. *Doctoral dissertation*, University of Houston, TX, USA., referenced by Hutchison et al., 1991.

Hutchison, J., and Khumawala, B., 1990, Scheduling random flexible manufacturing systems with dynamic environments. *Journal of Operations Management*, 9:335-351.

Hutchison, J., Leong, K., Synder, D., and Ward., P., 1991, Scheduling approaches for random job shop flexible manufacturing systems. *International Journal of Production Research*, 29:1053-1067.

Hwan, S.S., and Shogan, A.W., 1989, Modelling and solving an FMS part problem, *International Journal of Production Research*, 27:1349-1366.

Jackson, J.R., 1956, An extension of Johnson's results on job lot scheduling. *Naval Research Logistics Quarterly*, 3:201-203.

Jaikumar, R., 1986, Post industrial manufacturing. *Harvard Business Review*, 64:69-76.

Kim, Y.D., and Yano, C.A., 1994, A due date-based approach to part type selection in flexible manufacturing systems. *International journal of Production Research*, 32:1027-1043.

Kimemia, J., and Gershwin, S.B.,1983, An algorithm for the computer control of a flexible manufacturing system. *IIE Transactions*, 15:353-362, referenced by Chang et al., 1989.

Kusiak A., 1985, Flexible manufacturing systems: a structural approach. *International Journal of Production Research*, 23:1057-1073.

Laguna, M., Barnes, J.W., and Glover, F., 1991, Tabu search methods for a single machine scheduling problem. *Journal of Intelligent Manufacturing*, 2:63-74.

Laguna, M., Barnes, J.W., and Glover, F., 1993, Intelligent scheduling with tabu search : An application to jobs with linear delay penalties and sequence-dependent setup costs and times. *Journal of Applied Intelligence*, 3:159-172.

Laguna, M., and Glover, F., 1993, Integrating target analysis and tabu search for improved scheduling systems. *Expert Systems With Applications*, 6:287-297.

Logendran, R., Ramakrishna, P. and Sriskandarajah C., 1994, Tabu search-based heuristics for cellular manufacturing systems in the presence of alternative process plans. *International Journal of Production Research*, 32:273-297.

Luggen, W. W., 1991, *Flexible Manufacturing cells and systems* (New Jersey: Prentice Hall).

Maleki, R.A., 1991, *Flexible Manufacturing Systems: The Technology and Management*, (New Jersey: Prentice Hall).

Montgomery, D.C., 1991, *Design and Analysis of Experiments* (New York: Wiley).

Mukhopadhyay, S.K., Midha, S., and Krishna, V.M., 1992, A heuristic procedure for loading problems in flexible manufacturing systems, *International Journal of Production Research*, 9:2213-2228.

Rachamadugu, R., and Stecke, K., 1987, Classification and review of FMS scheduling procedures. Working paper #481R (University of Michigan, MI, USA), referenced by Hutchison et al., 1991)

Rajamani, D., Singh, N., and Aneja, Y.P., 1990, Integrated design of cellular manufacturing systems in the presence of alternative process plans. *International Journal of Production Research*, 28:1541-1554.

Reeves, C.R., 1993, Improving the efficiency of tabu search for machine sequencing problems. *Journal of the Operational Research Society*, 44:375-382.

Sarin, S., and Dar-El., E., 1984, Approaches to the scheduling problem in flexible manufacturing system. *The Industrial Engineering Conference Proceedings*, Institute of Industrial Engineers., referenced by Hutchison et al., 1991.

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

Skorin Kapov, J., 1990, Tabu search applied to the quadratic assignment problem. *ORSA Journal on Computing*, 2:33-45.

Skorin Kapov, J. and Vakharia, A.J.,1993, Scheduling a flow-line manufacturing cell: a tabu search approach. *International Journal of Production Research*, 31:1721-1734.

STATGRAPHICS, 1988 (Rockville, MD: Statistical Graphics Corporation, Inc.).

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

Suri, R., and Whitney, C.K., 1984, Decision support requirements in flexible manufacturing. *Journal of Manufacturing Systems*, 3:61-69.

SuperLINDO, 1989 (Chicago, IL: Lindo System, Inc.)

Taillard, E., 1989, Parallel tabu search techniques for the job shop scheduling problem, *Report ORWP 89/11*, Swiss Federal Institute of Technology of Lausanne, CH-1015, Switzerland, referenced by Skorin-Kapov and Vakharia, 1993.

Taillard, E., 1990, Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research*, 47:65-74.

Taillard, E., 1993, Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64:278-285.

Vakharia, A.J. and Chang, Y.L., 1990, A simulated annealing approach to scheduling a manufacturing cell, *Naval Research Logistics*, 37:559-577.

Widmer, M. and Hertz, A., 1989, A new heuristic method for the flow shop sequencing problem. *European Journal of Operational Research*, 41:186-193.

Widmer, M., 1991, Job shop scheduling with tool constraints: a tabu search approach. *Journal of the Operational Research Society*,42:75-82.

APPENDICES

APPENDIX A.

Mathematical Formulation for the Small Problem in SuperLINDO

Mathematical Formulation for the Small Problem in SuperLINDO

MIN TMAX

SUBJECT TO

2) Z11 + Z12 = 1
3) Z21 + Z22 = 1
4) Z31 + Z32 = 1
5) Y11,11 + Y21,11 - Z11 = 0
6) Y12,11 + Y32,11 - Z11 = 0
7) Y12,12 + Y32,12 - Z12 = 0
8) Y23,12 + Y33,12 - Z12 = 0
9) Y11,21 + Y21,21 - Z21 = 0
10) Y12,21 + Y32,21 - Z21 = 0
11) Y23,21 + Y33,21 - Z21 = 0
12) Y12,22 + Y32,22 - Z22 = 0
13) Y23,22 + Y33,22 - Z22 = 0
14) Y11,31 + Y21,31 - Z31 = 0
15) Y23,31 + Y33,31 - Z31 = 0
16) Y11,32 + Y21,32 - Z32 = 0
17) Y12,32 + Y32,32 - Z32 = 0
18) - TMAX + F12,11 <= 0
19) - TMAX + F32,11 <= 0
20) - TMAX + F23,12 <= 0
21) - TMAX + F33,12 <= 0
22) - TMAX + F23,21 <= 0
23) - TMAX + F33,21 <= 0
24) - TMAX + F23,22 <= 0
25) - TMAX + F33,22 <= 0
26) - TMAX + F23,31 <= 0
27) - TMAX + F33,31 <= 0
28) - TMAX + F12,32 <= 0
29) - TMAX + F32,32 <= 0
30) - 10 Y11,11 + F11,11 >= 0
31) - 16 Y21,11 + F21,11 >= 0
32) - 16 Y12,12 + F12,12 >= 0
33) - 15 Y32,12 + F32,12 >= 0
34) - 19 Y11,21 + F11,21 >= 0
35) - 18 Y21,21 + F21,21 >= 0
36) - 13 Y12,22 + F12,22 >= 0
37) - 10 Y32,22 + F32,22 >= 0
38) - 11 Y11,31 + F11,31 >= 0
39) - 11 Y21,31 + F21,31 >= 0
40) - 11 Y11,32 + F11,32 >= 0
41) - 10 Y21,32 + F21,32 >= 0
42) - 1000000 Y11,11 + F11,11 <= 0
43) - 1000000 Y21,11 + F21,11 <= 0
44) - 1000000 Y12,11 + F12,11 <= 0
45) - 1000000 Y32,11 + F32,11 <= 0
46) - 1000000 Y12,12 + F12,12 <= 0
47) - 1000000 Y32,12 + F32,12 <= 0
48) - 1000000 Y23,12 + F23,12 <= 0
49) - 1000000 Y33,12 + F33,12 <= 0

50) - 1000000 Y11,21 + F11,21 <= 0
51) - 1000000 Y21,21 + F21,21 <= 0
52) - 1000000 Y12,21 + F12,21 <= 0
53) - 1000000 Y32,21 + F32,21 <= 0
54) - 1000000 Y23,21 + F23,21 <= 0
55) - 1000000 Y33,21 + F33,21 <= 0
56) - 1000000 Y12,22 + F12,22 <= 0
57) - 1000000 Y32,22 + F32,22 <= 0
58) - 1000000 Y23,22 + F23,22 <= 0
59) - 1000000 Y33,22 + F33,22 <= 0
60) - 1000000 Y11,31 + F11,31 <= 0
61) - 1000000 Y21,31 + F21,31 <= 0
62) - 1000000 Y23,31 + F23,31 <= 0
63) - 1000000 Y33,31 + F33,31 <= 0
64) - 1000000 Y11,32 + F11,32 <= 0
65) - 1000000 Y21,32 + F21,32 <= 0
66) - 1000000 Y12,32 + F12,32 <= 0
67) - 1000000 Y32,32 + F32,32 <= 0
68) - 1000000 Y12,11 + F12,11 - F11,11 >= - 999988
69) - 1000000 Y12,11 + F12,11 - F21,11 >= - 999988
70) - 1000000 Y32,11 + F32,11 - F11,11 >= - 999982
71) - 1000000 Y32,11 + F32,11 - F21,11 >= - 999982
72) - 1000000 Y23,12 + F23,12 - F12,12 >= - 999987
73) - 1000000 Y23,12 + F23,12 - F32,12 >= - 999987
74) - 1000000 Y33,12 + F33,12 - F12,12 >= - 999981
75) - 1000000 Y33,12 + F33,12 - F32,12 >= - 999981
76) - 1000000 Y12,21 - F11,21 + F12,21 >= - 999989
77) - 1000000 Y12,21 - F21,21 + F12,21 >= - 999989
78) - 1000000 Y32,21 - F11,21 + F32,21 >= - 999981
79) - 1000000 Y32,21 - F21,21 + F32,21 >= - 999981
80) - 1000000 Y23,21 + F23,21 - F12,21 >= - 999983
81) - 1000000 Y23,21 + F23,21 - F32,21 >= - 999983
82) - 1000000 Y33,21 + F33,21 - F12,21 >= - 999985
83) - 1000000 Y33,21 + F33,21 - F32,21 >= - 999985
84) - 1000000 Y23,22 + F23,22 - F12,22 >= - 999989
85) - 1000000 Y23,22 + F23,22 - F32,22 >= - 999989
86) - 1000000 Y33,22 + F33,22 - F12,22 >= - 999986
87) - 1000000 Y33,22 + F33,22 - F32,22 >= - 999986
88) - 1000000 Y23,31 + F23,31 - F11,31 >= - 999980
89) - 1000000 Y23,31 + F23,31 - F21,31 >= - 999980
90) - 1000000 Y33,31 + F33,31 - F11,31 >= - 999986
91) - 1000000 Y33,31 + F33,31 - F21,31 >= - 999986
92) - 1000000 Y12,32 + F12,32 - F11,32 >= - 999990
93) - 1000000 Y12,32 + F12,32 - F21,32 >= - 999990
94) - 1000000 Y32,32 + F32,32 - F11,32 >= - 999986
95) - 1000000 Y32,32 + F32,32 - F21,32 >= - 999986
96) - 10 Y11,11 + 1000000 V11A11C + F11,11 - F11,21 >= 0
97) - 19 Y11,21 - 1000000 V11A11C - F11,11 + F11,21 >= - 1000000
98) - 10 Y11,11 + 1000000 V11A12C + F11,11 - F12,21 >= 0
99) - 11 Y12,21 - 1000000 V11A12C - F11,11 + F12,21 >= - 1000000
100) - 10 Y11,11 + 1000000 V11A12D + F11,11 - F12,22 >= 0
101) - 13 Y12,22 - 1000000 V11A12D - F11,11 + F12,22 >= - 1000000
102) - 10 Y11,11 + 1000000 V11A11E + F11,11 - F11,31 >= 0
103) - 11 Y11,31 - 1000000 V11A11E - F11,11 + F11,31 >= - 1000000

104) - 10 Y11,11 + 1000000 V11A11F + F11,11 - F11,32 >= 0
105) - 11 Y11,32 - 1000000 V11A11F - F11,11 + F11,32 >= - 1000000
106) - 10 Y11,11 + 1000000 V11A12F - F12,32 + F11,11 >= 0
107) - 10 Y12,32 - 1000000 V11A12F + F12,32 - F11,11 >= - 1000000
108) - 12 Y12,11 + 1000000 V12A11C + F12,11 - F11,21 >= 0
109) - 19 Y11,21 - 1000000 V12A11C - F12,11 + F11,21 >= - 1000000
110) - 12 Y12,11 + 1000000 V12A12C + F12,11 - F12,21 >= 0
111) - 11 Y12,21 - 1000000 V12A12C - F12,11 + F12,21 >= - 1000000
112) - 12 Y12,11 + 1000000 V12A12D + F12,11 - F12,22 >= 0
113) - 13 Y12,22 - 1000000 V12A12D - F12,11 + F12,22 >= - 1000000
114) - 12 Y12,11 + 1000000 V12A11E + F12,11 - F11,31 >= 0
115) - 11 Y11,31 - 1000000 V12A11E - F12,11 + F11,31 >= - 1000000
116) - 12 Y12,11 + 1000000 V12A11F + F12,11 - F11,32 >= 0
117) - 11 Y11,32 - 1000000 V12A11F - F12,11 + F11,32 >= - 1000000
118) - 12 Y12,11 + 1000000 V12A12F + F12,11 - F12,32 >= 0
119) - 10 Y12,32 - 1000000 V12A12F - F12,11 + F12,32 >= - 1000000
120) - 16 Y12,12 + 1000000 V12B11C + F12,12 - F11,21 >= 0
121) - 19 Y11,21 - 1000000 V12B11C - F12,12 + F11,21 >= - 1000000
122) - 16 Y12,12 + 1000000 V12B12C + F12,12 - F12,21 >= 0
123) - 11 Y12,21 - 1000000 V12B12C - F12,12 + F12,21 >= - 1000000
124) - 16 Y12,12 + 1000000 V12B12D + F12,12 - F12,22 >= 0
125) - 13 Y12,22 - 1000000 V12B12D - F12,12 + F12,22 >= - 1000000
126) - 16 Y12,12 + 1000000 V12B11E + F12,12 - F11,31 >= 0
127) - 11 Y11,31 - 1000000 V12B11E - F12,12 + F11,31 >= - 1000000
128) - 16 Y12,12 + 1000000 V12B11F + F12,12 - F11,32 >= 0
129) - 11 Y11,32 - 1000000 V12B11F - F12,12 + F11,32 >= - 1000000
130) - 16 Y12,12 + 1000000 V12B12F - F12,32 + F12,12 >= 0
131) - 10 Y12,32 - 1000000 V12B12F + F12,32 - F12,12 >= - 1000000
132) - 19 Y11,21 + 1000000 V11C11E + F11,21 - F11,31 >= 0
133) - 11 Y11,31 - 1000000 V11C11E - F11,21 + F11,31 >= - 1000000
134) - 19 Y11,21 + 1000000 V11C11F + F11,21 - F11,32 >= 0
135) - 11 Y11,32 - 1000000 V11C11F - F11,21 + F11,32 >= - 1000000
136) - 19 Y11,21 + 1000000 V11C12F - F12,32 + F11,21 >= 0
137) - 10 Y12,32 - 1000000 V11C12F + F12,32 - F11,21 >= - 1000000
138) - 11 Y12,21 + 1000000 V12C11E - F11,31 + F12,21 >= 0
139) - 11 Y11,31 - 1000000 V12C11E + F11,31 - F12,21 >= - 1000000
140) - 11 Y12,21 + 1000000 V12C11F - F11,32 + F12,21 >= 0
141) - 11 Y11,32 - 1000000 V12C11F + F11,32 - F12,21 >= - 1000000
142) - 11 Y12,21 + 1000000 V12C12F - F12,32 + F12,21 >= 0
143) - 10 Y12,32 - 1000000 V12C12F + F12,32 - F12,21 >= - 1000000
144) - 13 Y12,22 + 1000000 V12D11E + F12,22 - F11,31 >= 0
145) - 11 Y11,31 - 1000000 V12D11E - F12,22 + F11,31 >= - 1000000
146) - 13 Y12,22 + 1000000 V12D11F + F12,22 - F11,32 >= 0
147) - 11 Y11,32 - 1000000 V12D11F - F12,22 + F11,32 >= - 1000000
148) - 13 Y12,22 + 1000000 V12D12F - F12,32 + F12,22 >= 0
149) - 10 Y12,32 - 1000000 V12D12F + F12,32 - F12,22 >= - 1000000
150) - 16 Y21,11 + 1000000 V21A21C + F21,11 - F21,21 >= 0
151) - 18 Y21,21 - 1000000 V21A21C - F21,11 + F21,21 >= - 1000000
152) - 16 Y21,11 + 1000000 V21A23C - F23,21 + F21,11 >= 0
153) - 17 Y23,21 - 1000000 V21A23C + F23,21 - F21,11 >= - 1000000
154) - 16 Y21,11 + 1000000 V21A23D - F23,22 + F21,11 >= 0
155) - 11 Y23,22 - 1000000 V21A23D + F23,22 - F21,11 >= - 1000000
156) - 16 Y21,11 + 1000000 V21A21E + F21,11 - F21,31 >= 0
157) - 11 Y21,31 - 1000000 V21A21E - F21,11 + F21,31 >= - 1000000

158) - 16 Y21,11 + 1000000 V21A23E - F23,31 + F21,11 >= 0
159) - 20 Y23,31 - 1000000 V21A23E + F23,31 - F21,11 >= - 1000000
160) - 16 Y21,11 + 1000000 V21A21F + F21,11 - F21,32 >= 0
161) - 10 Y21,32 - 1000000 V21A21F - F21,11 + F21,32 >= - 1000000
162) - 13 Y23,12 + 1000000 V23B21C + F23,12 - F21,21 >= 0
163) - 18 Y21,21 - 1000000 V23B21C - F23,12 + F21,21 >= - 1000000
164) - 13 Y23,12 + 1000000 V23B23C + F23,12 - F23,21 >= 0
165) - 17 Y23,21 - 1000000 V23B23C - F23,12 + F23,21 >= - 1000000
166) - 13 Y23,12 + 1000000 V23B23D + F23,12 - F23,22 >= 0
167) - 11 Y23,22 - 1000000 V23B23D - F23,12 + F23,22 >= - 1000000
168) - 13 Y23,12 + 1000000 V23B21E + F23,12 - F21,31 >= 0
169) - 11 Y21,31 - 1000000 V23B21E - F23,12 + F21,31 >= - 1000000
170) - 13 Y23,12 + 1000000 V23B23E + F23,12 - F23,31 >= 0
171) - 20 Y23,31 - 1000000 V23B23E - F23,12 + F23,31 >= - 1000000
172) - 13 Y23,12 + 1000000 V23B21F + F23,12 - F21,32 >= 0
173) - 10 Y21,32 - 1000000 V23B21F - F23,12 + F21,32 >= - 1000000
174) - 18 Y21,21 + 1000000 V21C21E + F21,21 - F21,31 >= 0
175) - 11 Y21,31 - 1000000 V21C21E - F21,21 + F21,31 >= - 1000000
176) - 18 Y21,21 + 1000000 V21C23E - F23,31 + F21,21 >= 0
177) - 20 Y23,31 - 1000000 V21C23E + F23,31 - F21,21 >= - 1000000
178) - 18 Y21,21 + 1000000 V21C21F + F21,21 - F21,32 >= 0
179) - 10 Y21,32 - 1000000 V21C21F - F21,21 + F21,32 >= - 1000000
180) - 17 Y23,21 + 1000000 V23C21E + F23,21 - F21,31 >= 0
181) - 11 Y21,31 - 1000000 V23C21E - F23,21 + F21,31 >= - 1000000
182) - 17 Y23,21 + 1000000 V23C23E + F23,21 - F23,31 >= 0
183) - 20 Y23,31 - 1000000 V23C23E - F23,21 + F23,31 >= - 1000000
184) - 17 Y23,21 + 1000000 V23C21F + F23,21 - F21,32 >= 0
185) - 10 Y21,32 - 1000000 V23C21F - F23,21 + F21,32 >= - 1000000
186) - 11 Y23,22 + 1000000 V23D21E + F23,22 - F21,31 >= 0
187) - 11 Y21,31 - 1000000 V23D21E - F23,22 + F21,31 >= - 1000000
188) - 11 Y23,22 + 1000000 V23D23E + F23,22 - F23,31 >= 0
189) - 20 Y23,31 - 1000000 V23D23E - F23,22 + F23,31 >= - 1000000
190) - 11 Y23,22 + 1000000 V23D21F + F23,22 - F21,32 >= 0
191) - 10 Y21,32 - 1000000 V23D21F - F23,22 + F21,32 >= - 1000000
192) - 18 Y32,11 + 1000000 V32A32C + F32,11 - F32,21 >= 0
193) - 19 Y32,21 - 1000000 V32A32C - F32,11 + F32,21 >= - 1000000
194) - 18 Y32,11 + 1000000 V32A33C + F32,11 - F33,21 >= 0
195) - 15 Y33,21 - 1000000 V32A33C - F32,11 + F33,21 >= - 1000000
196) - 18 Y32,11 + 1000000 V32A32D + F32,11 - F32,22 >= 0
197) - 10 Y32,22 - 1000000 V32A32D - F32,11 + F32,22 >= - 1000000
198) - 18 Y32,11 + F32,11 - F33,22 + 1000000 V32A33D >= 0
199) - 14 Y33,22 - F32,11 + F33,22 - 1000000 V32A33D >= - 1000000
200) - 18 Y32,11 + 1000000 V32A33E + F32,11 - F33,31 >= 0
201) - 14 Y33,31 - 1000000 V32A33E - F32,11 + F33,31 >= - 1000000
202) - 18 Y32,11 + 1000000 V32A32F + F32,11 - F32,32 >= 0
203) - 14 Y32,32 - 1000000 V32A32F - F32,11 + F32,32 >= - 1000000
204) - 15 Y32,12 + 1000000 V32B32C + F32,12 - F32,21 >= 0
205) - 19 Y32,21 - 1000000 V32B32C - F32,12 + F32,21 >= - 1000000
206) - 15 Y32,12 + 1000000 V32B33C - F33,21 + F32,12 >= 0
207) - 15 Y33,21 - 1000000 V32B33C + F33,21 - F32,12 >= - 1000000
208) - 15 Y32,12 + 1000000 V32B32D + F32,12 - F32,22 >= 0
209) - 10 Y32,22 - 1000000 V32B32D - F32,12 + F32,22 >= - 1000000
210) - 15 Y32,12 + 1000000 V32B33D - F33,22 + F32,12 >= 0
211) - 14 Y33,22 - 1000000 V32B33D + F33,22 - F32,12 >= - 1000000

```
   212) - 15 Y32,12 + 1000000 V32B33E - F33,31 + F32,12 >=  0
   213) - 14 Y33,31 - 1000000 V32B33E + F33,31 - F32,12 >= - 1000000
   214) - 15 Y32,12 + 1000000 V32B32F - F32,32 + F32,12 >=  0
   215) - 14 Y32,32 - 1000000 V32B32F + F32,32 - F32,12 >= - 1000000
   216) - 19 Y33,12 + 1000000 V33B32C + F33,12 - F32,21 >=  0
   217) - 19 Y32,21 - 1000000 V33B32C - F33,12 + F32,21 >= - 1000000
   218) - 19 Y33,12 + 1000000 V33B33C + F33,12 - F33,21 >=  0
   219) - 15 Y33,21 - 1000000 V33B33C - F33,12 + F33,21 >= - 1000000
   220) - 19 Y33,12 + 1000000 V33B32D + F33,12 - F32,22 >=  0
   221) - 10 Y32,22 - 1000000 V33B32D - F33,12 + F32,22 >= - 1000000
   222) - 19 Y33,12 + 1000000 V33B33D + F33,12 - F33,22 >=  0
   223) - 14 Y33,22 - 1000000 V33B33D - F33,12 + F33,22 >= - 1000000
   224) - 19 Y33,12 + 1000000 V33B33E + F33,12 - F33,31 >=  0
   225) - 14 Y33,31 - 1000000 V33B33E - F33,12 + F33,31 >= - 1000000
   226) - 19 Y33,12 + 1000000 V33B32F + F33,12 - F32,32 >=  0
   227) - 14 Y32,32 - 1000000 V33B32F - F33,12 + F32,32 >= - 1000000
   228) - 19 Y32,21 + 1000000 V32C33E - F33,31 + F32,21 >=  0
   229) - 14 Y33,31 - 1000000 V32C33E + F33,31 - F32,21 >= - 1000000
   230) - 19 Y32,21 + 1000000 V32C32F - F32,32 + F32,21 >=  0
   231) - 14 Y32,32 - 1000000 V32C32F + F32,32 - F32,21 >= - 1000000
   232) - 15 Y33,21 + 1000000 V33C33E + F33,21 - F33,31 >=  0
   233) - 14 Y33,31 - 1000000 V33C33E - F33,21 + F33,31 >= - 1000000
   234) - 15 Y33,21 + 1000000 V33C32F + F33,21 - F32,32 >=  0
   235) - 14 Y32,32 - 1000000 V33C32F - F33,21 + F32,32 >= - 1000000
   236) - 10 Y32,22 + 1000000 V32D33E - F33,31 + F32,22 >=  0
   237) - 14 Y33,31 - 1000000 V32D33E + F33,31 - F32,22 >= - 1000000
   238) - 10 Y32,22 + 1000000 V32D32F - F32,32 + F32,22 >=  0
   239) - 14 Y32,32 - 1000000 V32D32F + F32,32 - F32,22 >= - 1000000
   240) - 14 Y33,22 + 1000000 V33D33E + F33,22 - F33,31 >=  0
   241) - 14 Y33,31 - 1000000 V33D33E - F33,22 + F33,31 >= - 1000000
   242) - 14 Y33,22 + 1000000 V33D32F + F33,22 - F32,32 >=  0
   243) - 14 Y32,32 - 1000000 V33D32F - F33,22 + F32,32 >= - 1000000
END
INTE   Z11
INTE   Z12
INTE   Z21
INTE   Z22
INTE   Z31
INTE   Z32
INTE  Y11,11
INTE  Y12,11
INTE  Y21,11
INTE  Y32,11
INTE  Y12,12
INTE  Y23,12
INTE  Y32,12
INTE  Y33,12
INTE  Y11,21
INTE  Y21,21
INTE  Y12,21
INTE  Y32,21
INTE  Y23,21
INTE  Y33,21
INTE  Y12,22
```

```
INTE  Y32,22
INTE  Y23,22
INTE  Y33,22
INTE  Y11,31
INTE  Y21,31
INTE  Y23,31
INTE  Y33,31
INTE  Y11,32
INTE  Y21,32
INTE  Y12,32
INTE  Y32,32
INTE V11A11C
INTE V11A12C
INTE V11A12D
INTE V11A11E
INTE V11A11F
INTE V11A12F
INTE V12A11C
INTE V12A12C
INTE V12A12D
INTE V12A11E
INTE V12A11F
INTE V12A12F
INTE V12B11C
INTE V12B12C
INTE V12B12D
INTE V12B11E
INTE V12B11F
INTE V12B12F
INTE V11C11E
INTE V11C11F
INTE V11C12F
INTE V12C11E
INTE V12C11F
INTE V12C12F
INTE V12D11E
INTE V12D11F
INTE V12D12F
INTE V21A21C
INTE V21A23C
INTE V21A23D
INTE V21A21E
INTE V21A23E
INTE V21A21F
INTE V23B21C
INTE V23B23C
INTE V23B23D
INTE V23B21E
INTE V23B23E
INTE V23B21F
INTE V21C21E
INTE V21C23E
INTE V21C21F
INTE V23C21E
```

INTE V23C23E
INTE V23C21F
INTE V23D21E
INTE V23D23E
INTE V23D21F
INTE V32A32C
INTE V32A33C
INTE V32A32D
INTE V32A33D
INTE V32A33E
INTE V32A32F
INTE V32B32C
INTE V32B33C
INTE V32B32D
INTE V32B33D
INTE V32B33E
INTE V32B32F
INTE V33B32C
INTE V33B33C
INTE V33B32D
INTE V33B33D
INTE V33B33E
INTE V33B32F
INTE V32C33E
INTE V32C32F
INTE V33C33E
INTE V33C32F
INTE V32D33E
INTE V32D32F
INTE V33D33E
INTE V33D32F

Note:

(1) A denotes as process plan 1 of part 1 or (1,1) combination.
B denotes as process plan 2 of part 1 or (1,2) combination.
C denotes as process plan 1 of part 2 or (2,1) combination.
D denotes as process plan 2 of part 2 or (2,2) combination.
E denotes as process plan 1 of part 3 or (3,1) combination and
F denotes as process plan 2 of part 3 or (3,2) combination.

(2) H is assumed equal to 1,000,000.

APPENDIX B.

Potential Data Set

Table B.1 A potential data set for the second problem structure-5P*4M*4O: operation k; part j; process plan p.

k	Alternative machines	j = 1			j = 2		j = 3		j = 4		j = 5	
		p = 1	p = 2	p = 3	p = 1	p = 2	p = 1	p = 2	p = 1	p = 2	p = 1	p = 2
1	1,2	1		1	1				1		1	
2	3,4	1	1			1	1	1	1			1
3	1,4			1				1		1		1
4	2,3		1		1	1	1			1	1	

Table B.2 A potential data set for the third problem structure-10P*5M*5O: operation k; part j; process plan p.

k	Alternative machines	j = 1			j = 2		j = 3		j = 4		j = 5	
		p = 1	p = 2	p = 3	p = 1	p = 2	p = 1	p = 2	p = 1	p = 2	p = 1	p = 2
1	1,2	1		1		1	1	1		1	1	
2	3,4		1			1	1			1		
3	1,5	1	1			1			1		1	1
4	2,3			1	1						1	1
5	4,5	1		1	1			1				

k	Alternative machines	j = 6			j = 7		j = 8		j = 9		j = 10		
		p = 1	p = 2	p = 3	p = 1	p = 2	p = 1	p = 2	p = 1	p = 2	p = 1	p = 2	p = 3
1	1,2	1					1					1	1
2	3,4	1		1				1	1			1	
3	1,5			1		1	1						
4	2,3		1		1					1	1		1
5	4,5		1		1	1			1	1	1	1	

Table B.3 A potential data set for the third problem structure-14P*7M*7O: operation k; part j; process plan p.

k	Alternative machines	j = 1		j = 2			j = 3		j = 4		j = 5		
		p = 1	p = 2	p = 1	p = 2	p = 3	p = 1	p = 2	p = 1	p = 2	p = 1	p = 2	p = 3
1	1,2		1	1						1	1		
2	3,4	1			1			1	1		1		1
3	5,6				1							1	
4	1,3		1			1	1			1			1
5	3,6			1								1	
6	5,7	1			1			1		1			
7	2,4	1				1			1		1		1

k	Alternative machines	j = 6		j = 7		j = 8		j = 9			j = 10	
		p = 1	p = 2	p = 1	p = 2	p = 1	p = 2	p = 1	p = 2	p = 3	p = 1	p = 2
1	1,2		1			1		1				1
2	3,4		1	1						1		
3	5,6	1			1		1		1		1	
4	1,3			1			1	1	1			1
5	3,6		1		1	1				1		
6	5,7	1				1				1		1
7	2,4				1		1	1			1	

k	Alternative machines	j = 11			j = 12			j = 13	j = 14	
		p = 1	p = 2	p = 3	p = 1	p = 2	p = 3	p = 1	p = 1	p = 2
1	1,2			1			1		1	
2	3,4		1	1		1	1			
3	5,6	1			1			1		1
4	1,3	1			1					
5	3,6		1			1			1	1
6	5,7			1			1			
7	2,4		1			1				

Table B 4 Parameters used in tabu search-based heuristics for each problem structure.

Parameters	4P*3M*3O		5P*4M*4O		10P*5M*5O		14P*7M*7O	
	Inside Search	Outside Search	Inside Search	Outside Search	Inside Search	Outside Search	Inside Search	Outside Search
Tabu List Size	Fixed: 6 Variable: -initial: 6 -decreased: 4 -increased: 8	Fixed: 3 Variable: -initial: 3 -decreased: 2 -increased: 4	Fixed: 10 Variable: -initial: 10 -decreased: 7 -increased: 14	Fixed: 3 Variable: -initial: 3 -decreased: 2 -increased: 5	Fixed: 25 Variable: -initial: 25 -decreased: 17 -increased: 34	Fixed: 7 Variable: -initial: 7 -decreased: 5 -increased: 12	Fixed: 49 Variable: -initial: 49 -decreased: 33 -increased: 66	Fixed: 9 Variable: -initial: 9 -decreased: 6 -increased: 17
Number of Iterations w/o Improvement	4	4	5	5	10	11	17	16
Number of Restarts	2	4	2	4	2	4	2	4

APPENDIX C.

Normal Probability Plots for Each problem Structure

Figure C.1 Normal probability plot for 4P*3M*3O problem structure

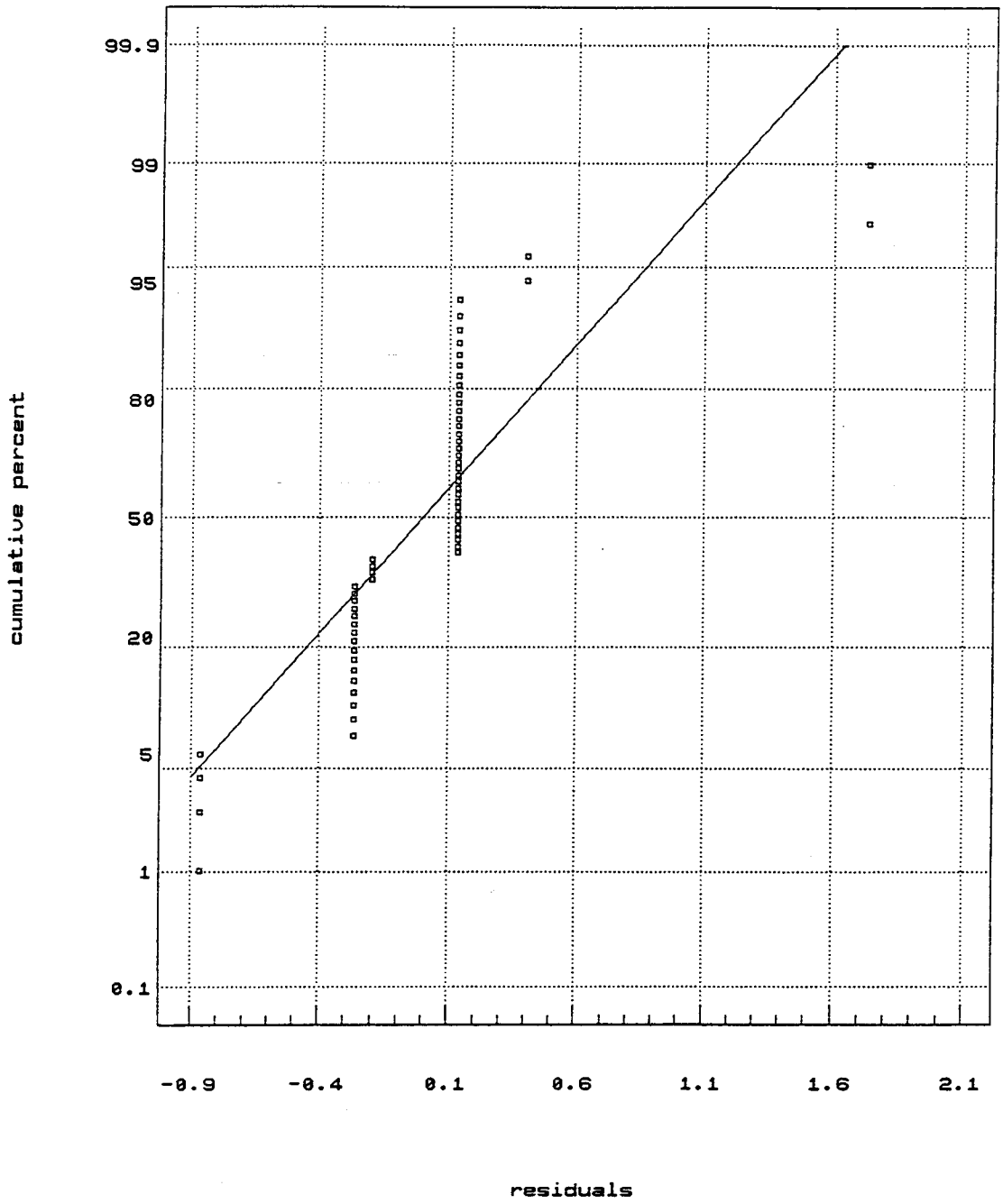


Figure C.2 Normal probability plot for 5P*4M*4O problem structure

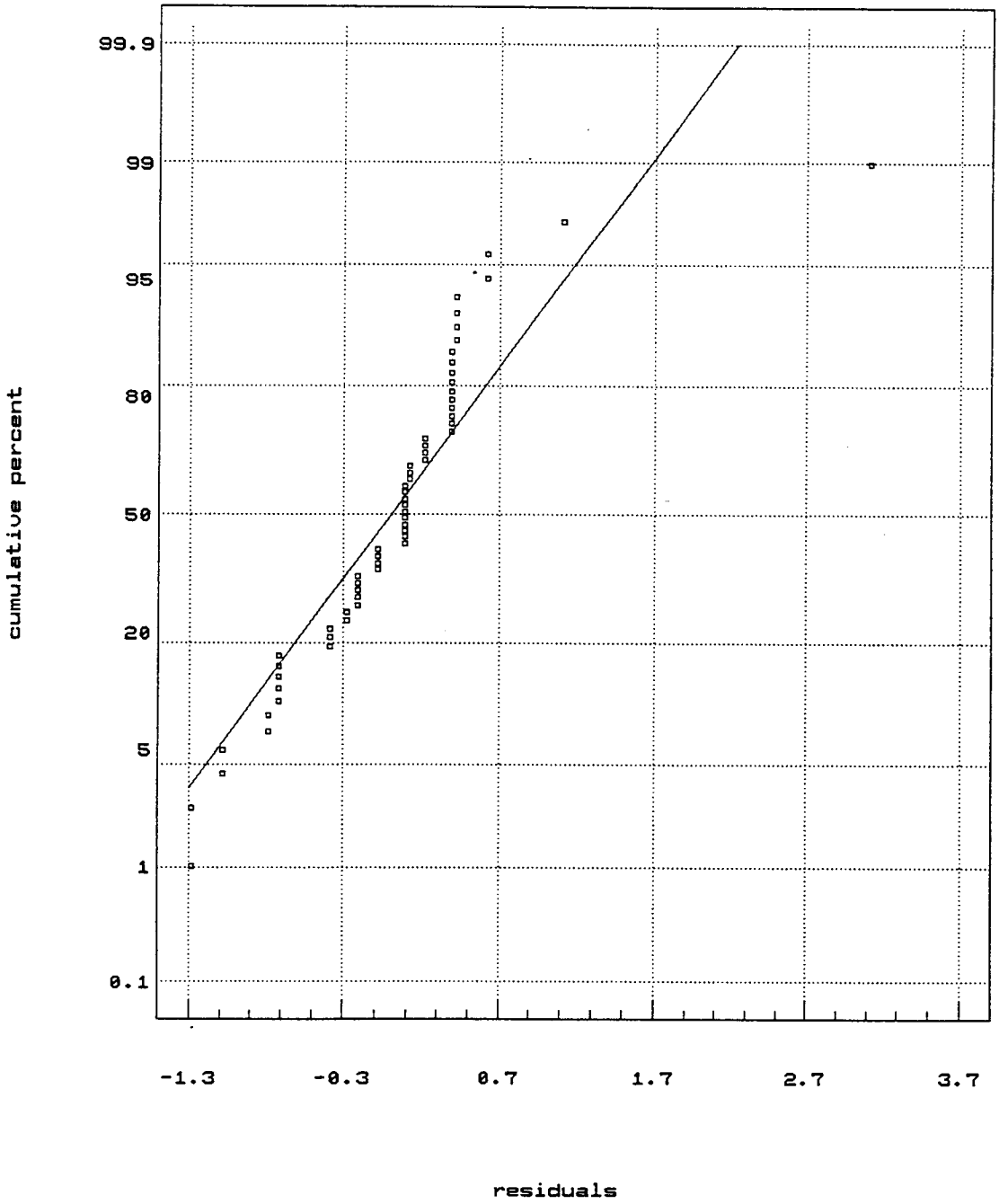


Figure C.3 Normal probability plot for 10P*5M*5O problem structure

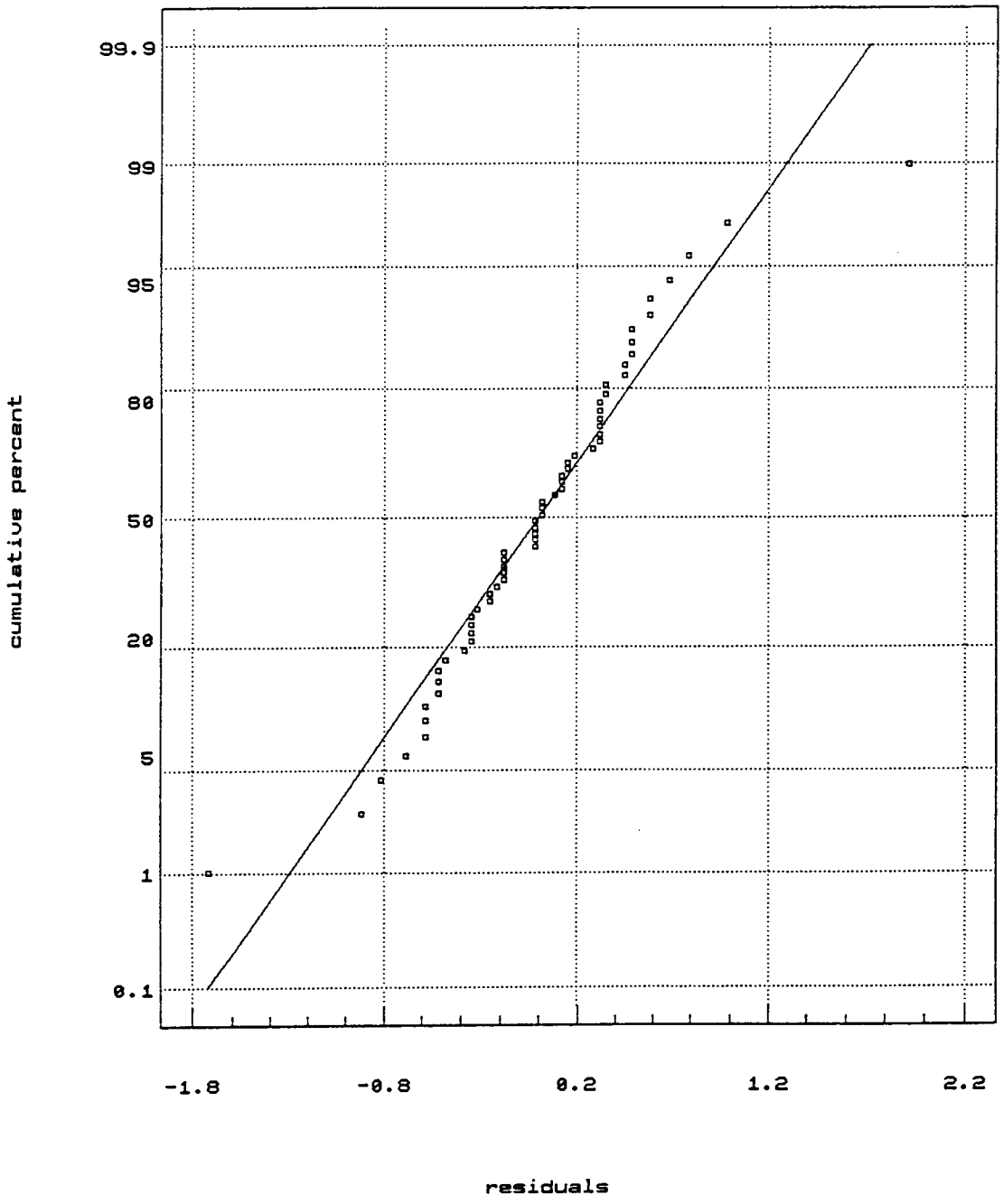
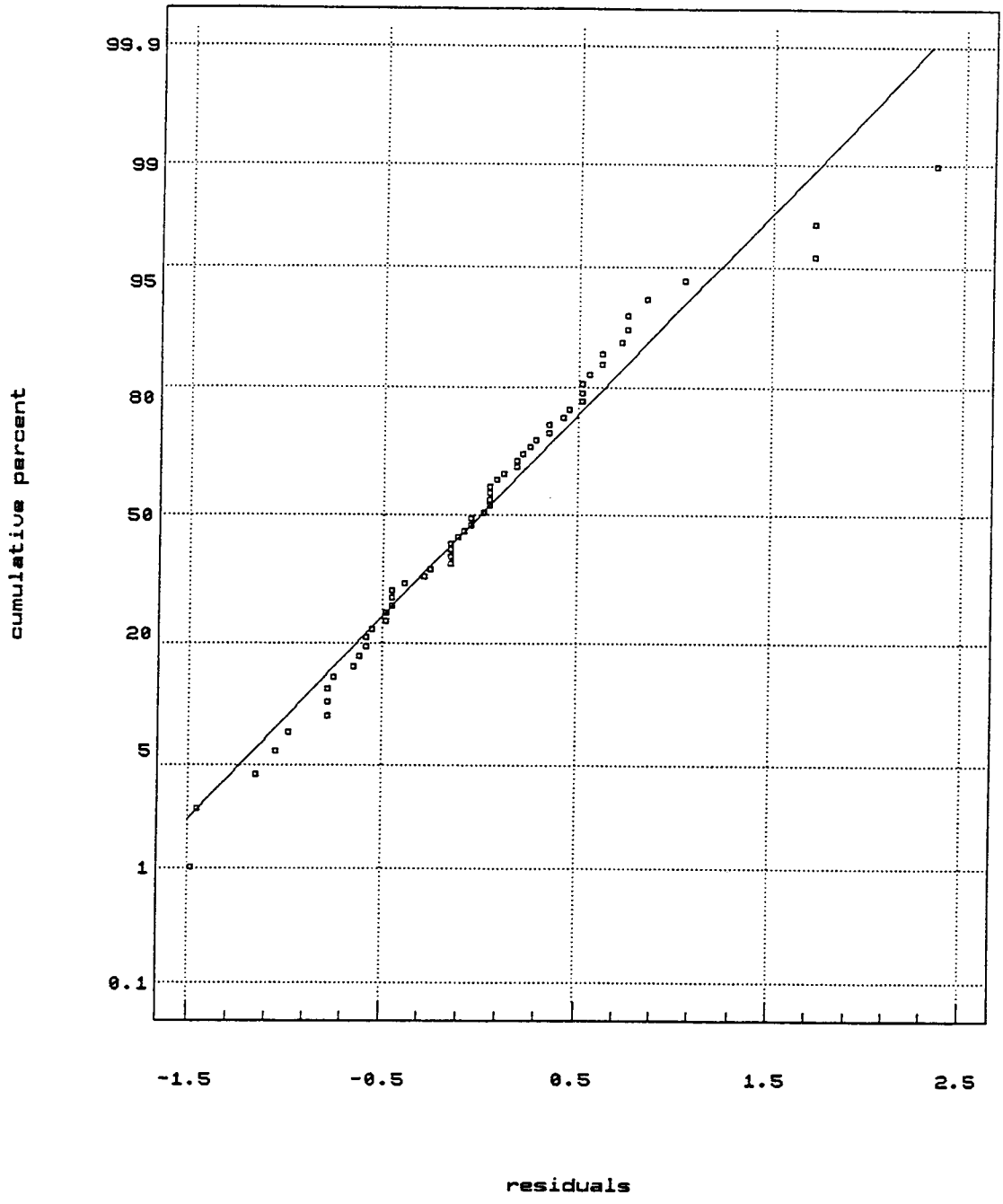


Figure C.4 Normal probability plot for 14P*7M*7O problem structure



APPENDIX D.

Results Obtained for Each Problem Structure

Table D.1 Results obtained for 4P*3M*3O problem structure.

4P*3M*3O

Problem Structure 1	TSH 1	TSH 2	TSH 3	TSH 4	TSH 5	TSH 6
Problem 1	49	49	49	49	49	49
Problem 2	45	45	45	45	45	45
Problem 3	45	45	45	45	45	45
Problem 4	36	36	36	36	36	36
Problem 5	45	42	42	45	42	42
Problem 6	44	44	44	44	44	44
Problem 7	41	40	40	41	40	40
Problem 8	44	44	44	44	44	44
Problem 9	44	44	44	44	44	44
Problem 10	47	47	47	47	47	47
Avg. MakeSpan	44	43.6	43.6	44	43.6	43.6

CPU time (in seconds)

Problem Structure 1	TSH 1	TSH 2	TSH 3	TSH 4	TSH 5	TSH 6
Problem 1	7.09	7.14	6.73	7.41	7.36	7.75
Problem 2	5	5.71	7.91	5.5	6.81	7.14
Problem 3	5.44	7.09	6.59	5.93	7.42	7.63
Problem 4	5.05	6.04	7.14	5.27	6.32	6.87
Problem 5	4.94	7.64	7.14	4.95	7.74	7.09
Problem 6	6.86	6.76	6.32	7.09	7.09	7.14
Problem 7	5.77	6.75	6.54	6.04	7.25	7.36
Problem 8	4.67	5.87	6.15	4.73	6.27	6.65
Problem 9	4.17	6.7	7.47	5.05	7.25	6.59
Problem 10	5.76	7.69	7.78	5.87	8.02	7.85
Avg. Time	5.475	6.739	6.977	5.784	7.153	7.207

Table D.2 Results obtained for 5P*4M*4O problem structure.

5P*4M*4O

Problem Structure 2	TSH 1	TSH 2	TSH 3	TSH 4	TSH 5	TSH 6
Problem 1	44	44	44	44	44	44
Problem 2	45	45	45	45	45	45
Problem 3	46	45	43	46	45	43
Problem 4	43	43	43	43	43	43
Problem 5	43	43	43	43	43	43
Problem 6	46	46	45	46	46	45
Problem 7	45	45	45	46	45	45
Problem 8	44	42	42	47	42	42
Problem 9	43	43	43	43	43	43
Problem 10	44	44	44	45	44	44
Avg. MakeSpan	44.3	44	43.7	44.8	44	43.7

CPU time (in seconds)

Problem Structure 2	TSH 1	TSH 2	TSH 3	TSH 4	TSH 5	TSH 6
Problem 1	9.56	14.67	12.58	15.26	15.27	16.65
Problem 2	9.72	13.78	12.69	13.79	13.84	13.95
Problem 3	10.06	13.13	13.57	13.67	13.07	14.28
Problem 4	10.41	14.88	14.77	14.21	14.94	15.32
Problem 5	10.73	12.96	12.74	14.61	13.41	13.57
Problem 6	9.77	13.46	14.17	14.23	13.84	14.05
Problem 7	12.47	16.6	13.57	17.08	17.14	16.59
Problem 8	9.83	13.35	12.57	10.82	13.4	13.35
Problem 9	10.62	14.5	13.13	14.99	14.78	14.94
Problem 10	9.5	14.72	13.02	15.98	15.44	15.49
Avg. Time	10.267	14.205	13.281	14.464	14.513	14.819

Table D.3 Results obtained for 10P*5M*5O problem structure.

10P*5M*5O

Problem Structure 3	TSH 1	TSH 2	TSH 3	TSH 4	TSH 5	TSH 6
Problem 1	56	55	55	56	55	55
Problem 2	54	53	53	54	53	53
Problem 3	58	58	55	58	56	58
Problem 4	53	53	53	53	53	53
Problem 5	54	54	54	54	54	54
Problem 6	58	58	57	58	57	58
Problem 7	62	61	61	62	61	64
Problem 8	59	58	59	59	59	58
Problem 9	59	59	59	59	59	60
Problem 10	57	56	56	57	56	56
Avg MakeSpan	57	56.5	56.2	57	56.3	56.9

CPU time (in mins)

Problem Structure 3	TSH 1	TSH 2	TSH 3	TSH 4	TSH 5	TSH 6
Problem 1	4.46	32.07	21.42	2.38	11.57	19.46
Problem 2	5.1	25.53	23.34	3	11.09	8.44
Problem 3	6	29.05	21.57	2.21	10.3	11.27
Problem 4	4.24	21.49	27.07	2.25	12	9.11
Problem 5	5.15	27.48	24.59	5.03	8.26	9.15
Problem 6	5.25	32.1	29.13	5.17	16.42	9.06
Problem 7	7.42	31.1	32.26	7.07	10.36	7.37
Problem 8	4.58	26.25	29.4	2.36	9.11	10.49
Problem 9	9.04	39.52	23.26	2.58	18.38	17.26
Problem 10	8.57	31.39	33.54	9.08	9.32	15.02
Avg. Time	5.981	29.598	26.558	4.113	11.681	11.663

Note: Maximum number of entries in the OIL of 10 is used as a second stopping criterion.

Table D.4 Results obtained for 14P*7M*7O problem structure.

14P*7M*7O

Problem Structure 4	TSH 1	TSH 2	TSH 3	TSH 4	TSH 5	TSH 6
Problem 1	68	67	68	68	68	68
Problem 2	69	68	68	69	68	66
Problem 3	68	68	68	68	68	67
Problem 4	67	67	67	69	68	69
Problem 5	71	71	71	74	71	70
Problem 6	71	71	70	71	70	70
Problem 7	70	69	69	70	69	68
Problem 8	71	71	70	71	69	71
Problem 9	73	71	70	73	71	71
Problem 10	68	64	64	66	64	64
Avg. MakeSpan	69.6	68.7	68.5	69.9	68.6	68.4

CPU time (in mins)

Problem Structure 4	TSH 1	TSH 2	TSH 3	TSH 4	TSH 5	TSH 6
Problem 1	67.52	183	188	21.57	102	121
Problem 2	68.52	225	249	27.09	139	122
Problem 3	59.26	201	188	23.05	82	113
Problem 4	83.48	218	214	27.31	142	128
Problem 5	76.06	245	249	27.03	137	119
Problem 6	73.22	206	228	17.46	110	106
Problem 7	67.07	195	167	23.09	103	119
Problem 8	86.57	206	238	26.59	133	125
Problem 9	87.32	217	270	23.57	115	123
Problem 10	66.23	217	192	18.54	110	97
Avg. Time	73.525	211.3	218.3	23.53	117.3	117.3

Note: Maximum number of entries in the OIL of 15 is used as a second stopping criterion.

APPENDIX E.

Analysis of Variance for Each Problem Structure

Table E.1 Results obtained from analysis of variance for 4P*3M*3O problem structure.

Source of Variation	Sum of Squares	Degree of Freedom	Mean Square	F-ratio
MAIN EFFECTS				
Treatments (TSH)	2.13333	5	0.426667	1.714
Blocks (Problems)	682.40000	9	75.822222	304.643
RESIDUAL (Error)	11.200000	45	0.2488889	
TOTAL (CORRECTED)	695.73333	59		

Contrast	Differences	LSD limits
TSH 1-TSH 2	0.40000	0.44947
TSH 1-TSH 3	0.40000	0.44947
TSH 1-TSH 4	0.00000	0.44947
TSH 1-TSH 5	0.40000	0.44947
TSH 1-TSH 6	0.40000	0.44947
TSH 2-TSH 3	0.00000	0.44947
TSH 2-TSH 4	-0.40000	0.44947
TSH 2-TSH 5	0.00000	0.44947
TSH 2-TSH 6	0.00000	0.44947
TSH 3-TSH 4	-0.40000	0.44947
TSH 3-TSH 5	0.00000	0.44947
TSH 3-TSH 6	0.00000	0.44947
TSH 4-TSH 5	0.40000	0.44947
TSH 4-TSH 6	0.40000	0.44947
TSH 5-TSH 6	0.00000	0.44947

Table E.2 Results obtained from analysis of variance for 5P*4M*4O problem structure.

Source of Variation	Sum of Squares	Degree of Freedom	Mean Square	F-ratio
MAIN EFFECTS				
Treatments (TSH)	8.683333	5	1.7366667	3.192
Blocks (Problems)	55.416667	9	6.1574074	11.317
RESIDUAL (Error)	24.483333	45	0.5440741	
TOTAL (CORRECTED)	88.583333	59		

Contrast	Differences	LSD limits
TSH 1-TSH 2	0.30000	0.66455
TSH 1-TSH 3	0.60000	0.66455
TSH 1-TSH 4	-0.50000	0.66455
TSH 1-TSH 5	0.30000	0.66455
TSH 1-TSH 6	0.60000	0.66455
TSH 2-TSH 3	0.30000	0.66455
TSH 2-TSH 4	-0.80000	0.66455*
TSH 2-TSH 5	0.00000	0.66455
TSH 2-TSH 6	0.30000	0.66455
TSH 3-TSH 4	-1.10000	0.66455*
TSH 3-TSH 5	-0.30000	0.66455
TSH 3-TSH 6	0.00000	0.66455
TSH 4-TSH 5	0.80000	0.66455*
TSH 4-TSH 6	1.10000	0.66455*
TSH 5-TSH 6	0.30000	0.66455

* denotes a statistically significant difference

Table E.3 Results obtained from analysis of variance for 10P*5M*5O problem structure.

Source of Variation	Sum of Squares	Degree of Freedom	Mean Square	F-ratio
MAIN EFFECTS				
Treatments (TSH)	6.55000	5	1.310000	3.548
Blocks (Problems)	430.48333	9	47.831481	129.534
RESIDUAL (Error)	16.616667	45	0.3692593	
TOTAL (CORRECTED)	453.65000	59		

Contrast	Differences	LSD limits
TSH 1-TSH 2	0.50000	0.54747
TSH 1-TSH 3	0.80000	0.54747*
TSH 1-TSH 4	0.00000	0.54747
TSH 1-TSH 5	0.70000	0.54747*
TSH 1-TSH 6	0.10000	0.54747
TSH 2-TSH 3	0.30000	0.54747
TSH 2-TSH 4	-0.50000	0.54747
TSH 2-TSH 5	0.20000	0.54747
TSH 2-TSH 6	-0.40000	0.54747
TSH 3-TSH 4	-0.80000	0.54747*
TSH 3-TSH 5	-0.10000	0.54747
TSH 3-TSH 6	-0.70000	0.54747*
TSH 4-TSH 5	0.70000	0.54747*
TSH 4-TSH 6	0.10000	0.54747
TSH 5-TSH 6	-0.60000	0.54747*

* denotes a statistically significant difference

Table E.4 Results obtained from analysis of variance for 14P*7M*7O problem structure.

Source of Variation	Sum of Squares	Degree of Freedom	Mean Square	F-ratio
MAIN EFFECTS				
Treatments (TSH)	20.15	5	4.030000	5.847
Blocks (Problems)	223.68333	9	24.853704	36.059
RESIDUAL (Error)	31.016667	45	0.6892593	
TOTAL (CORRECTED)	274.85000	59		

Contrast	Differences	LSD limits
TSH 1-TSH 2	0.90000	0.74798*
TSH 1-TSH 3	1.10000	0.74798*
TSH 1-TSH 4	-0.30000	0.74798
TSH 1-TSH 5	1.00000	0.74798*
TSH 1-TSH 6	1.20000	0.74798*
TSH 2-TSH 3	0.20000	0.74798
TSH 2-TSH 4	-1.20000	0.74798*
TSH 2-TSH 5	0.10000	0.74798
TSH 2-TSH 6	0.30000	0.74798
TSH 3-TSH 4	-1.40000	0.74798*
TSH 3-TSH 5	-0.10000	0.74798
TSH 3-TSH 6	0.10000	0.74798
TSH 4-TSH 5	1.30000	0.74798*
TSH 4-TSH 6	1.50000	0.74798*
TSH 5-TSH 6	0.20000	0.74798

* denotes a statistically significant difference