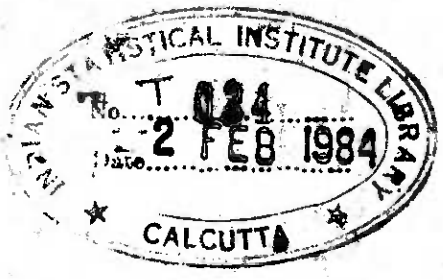
(81)

FLOW-SHOP SCHEDULING PROBLEMS

N.R. ACHUTHAN

Indian Statistical Institute

Calcutta



A thesis submitted to the Indian Statistical Institute

in partial fulfilment of the requirements for

the degree of Doctor of Philosophy

March, 1980.

A C K N O W L E D G E M E N T

I am indebted to Dr. A.C. Mukhopadhyay, my thesis supervisor, who promptly and cheerfully went through the manuscript and offered many valuable suggestions.

I express my thankfulness to Mr. D.T. Ghosh, Dr. J. Grabowski and Dr. J.B. Sidney for permitting me to include our joint work in this thesis.

I thank, my colleague and friend Anadi (Dr. T.S. Arthanari), who introduced me to this branch of Operations Research, for permitting me to include our joint work in this thesis.

I thank Dr. A.H.G. Rinnooy Kan of Erasmus University, Rotterdam and Dr. K.R. Baker of Duke University, Durham for their critical comments and suggestions on my work.

Mr. Jyotirmay Sarma typed this thesis very elegantly and quickly. Mr. Apurba Guha duplicated this thesis very meticulously. I thank both of them.

I thank my colleagues at the SQC and OR division for their share of association with this thesis.

N.R. Achuthan

C O N T E N T S

I N T R O D U C T I O N

Researchers' attention was drawn to the study of scheduling problems through mathematical modelling, probably for the first time, when Johnson (1954) published his famous paper in naval research logistic quarterly. Thereafter, many authors have continued to contribute to the growth of the theory of scheduling. A vast collection of papers related to this area is regularly published by journals like mamagement science, operations research, operational research quarterly, naval research logistic quarterly, opsearch etc.. A good grasp of the literature in this area can be had from the books by Conway et al.(1967), Baker (1974) and Rinnooy Kan (1976).

This thesis restricts its attention to the well known deterministic flow-shop scheduling problems which include the two machine flow-shop problem handled by Johnson (1954). The main feature of the flow-shop scheduling problem is : "There are m machines and n jobs. Each job consists of m tasks such that i^{th} task can be performed only on the i^{th} machine named M_i. Further, i^{th} task of a job can start on machine M_i only after the completion of $(i-1)^{th}$ task of that job on machine M_{i-1}, $2 \leq i \leq m$."

In the following we briefly outline the contents of this thesis chapterwise.

Chapter I deals with the basic definitions and notation used in this thesis. Under sections 1.0 and 1.1 we provide the definitions of the scheduling problem and the common objective functions used. In sections 1.2 and 1.3 we state clearly the flow-shop assumptions and define most of the symbols and concepts used in the later chapters of this thesis. In section 1.4 we introduce the lower bound classification and notation due to Lageweg et al. (1978). Under section 1.5 we provide a lucid introduction to NP-completeness following the presentation of Horowitz et al. (1978).

Chapter II deals with the $(n/3/F/F_{max})$ problem which is proved to be NP-complete by Garey et al. (1976). Under section 2.1 (based on Achuthan (1978)) we discuss the known special cases of this problem with polynomial time complex algorithms and subsequently discuss four new special cases and give polynomial time complex algorithms for them. To describe the new special cases we define q to be the smallest integer such that the sum of any q processing times on machine M_i is greater than or equal to sum of any q processing times on machine M_k. This condition is denoted by $M_i \overset{q}{\geq} M_k$. The four new special cases of this section are : (1) $M_2 \overset{q}{\geq} M_1$ (2) $M_2 \overset{q}{\geq} M_3$ (3) $M_2 \overset{q}{\geq} M_1$ and $M_2 \overset{q'}{\geq} M_3$ and (4) the processing times of all the jobs on second machine are equal and either $M_1 \overset{2}{\geq} M_2$ or $M_3 \overset{2}{\geq} M_2$ holds. In section 2.2 we discuss the heuristics proposed in the literature for

the $(n/3/F/F_{max})$ problem and conduct an experimental investigation with the aim of comparing the polynomial time complex heuristics for their performances. The experimental study includes six new basic heuristics and two new improvement procedures. On the basis of restrictions on processing times the problems are classified into three disjoint groups. In the case of two groups we provide a combination of heuristics with computational complexity $O(3n \log n + (12n - 5)n)$ and this suggested heuristic solves more than 75% of the problems optimally and around 96% of the problems within 5% error. For the third group we suggest the linear branch-and-bound heuristic incorporating a lower bound due to Lageweg et al. (1978) and dominance criteria of Szwarc (1971). This heuristic has computational complexity $O(n^4)$ and it solves about 63% of the problems optimally and 97% of the problems within 5% error. The material of this section 2.2 (based on Achuthan et al. (1979)) is a joint work with D.T. Ghosh.

Chapter III deals with the $(n/m/F/F_{max})$ problem. In section 3.1 we prove the following result : "If $\pi = (\pi(1), \ldots, \pi(n))$ is a Johnson's permutation for every two machine flow-shop problem constructed with machines M_u and M_v (in that order) $1 \leq u < v \leq m$, then π is an optimal solution for the $(n/m/F/F_{max})$ problem." We provide examples to show that the conditions for the result cannot be weakened. This result is a generalisation of the version proved for m=3 by Burns et al. (1976) who disproved Johnson's (1954) conjecture in this regard. Under

section 3.2 we handle a special case of the $(n/m/F/F_{max})$ problem, viz. There is a common Johnson's permutation for every two machine flow-shop problem constructed with the machines M_u and M_v (in that order) $1 \leq u < v \leq m$ but $(u,v) \neq (1,m)$. In this case we prove that there exists a permutation π satisfying the conditions of the result in section 3.1. Corresponding version of this special case for $m = 3$ was handled by Burns et al. (1975). The contents of sections 3.1 and 3.2 are published in Achuthan (1977). In section 3.3 we extend the probabilistic analysis due to Baker (1975) and Gupta (1975) to Szwarc's dominance criteria applied to the original and the reverse flow-shop problems simultaneously. Subsequently, we provide branch-and-bound procedure for the $(n/m/F/F_{max})$ problem wherein a node represents a two-sided partial permutation.

Chapter IV deals with the maximum penalty problem. Here the maximum penalty $f(\sigma)$ for an arbitrary schedule σ is defined by

$$f(\sigma) = \max \; g\Big(\max_{1 \leq j \leq n} E_j\Big), \; h\Big(\max_{1 \leq j \leq n} T_j\Big)$$

where $E_j = \max \; a_j - S_j, 0$; $T_j = \max \; C_j - b_j, 0$; S_j and C_j are start time and completion time of job j under schedule σ; a_j and b_j are given target start time and due date associated with job j and the functions g and h are monotonically non-decreasing continuous functions such that $g(0) = h(0) = 0$. This objective function has

been introduced by Sidney (1977). In section 4.1 we prove the reduction of the maximum penalty problem to a maximum tardiness problem under a very general set of conditions but using a vital translation assumption. In the subsection 4.1.1 we generalise a known procedure for solving the $(n/1/./T_{max})$ problem with preemption resume allowed to solve the maximum tardiness problem connected with n jobs and m machines with precedence relations among tasks of a particular job; but no precedence relations among the jobs and with some more important assumptions. In section 4.2 we introduce the maximum penalty flow-shop problem and prove interesting results which show that it is not enough to consider the permutation schedules alone while solving this problem. In this section we generalise some of the results of Sidney (1977). In subsequent sections we make the no passing assumption. In section 4.3 we prove a result for the maximum penalty flow-shop problem and this result is similar to the one in section 3.1 for the $(n/m/F/F_{max})$ problem. In section 4.4 we prove the NP-completeness of the maximum penalty flow-shop problem. In section 4.5 after developing the lower bounds we provide a branch-and-bound procedure to solve the maximum penalty flow-shop problem. The results of Chapter IV excluding subsection 4.1.1 are based on the joint work with J.B. Sidney and J. Grabowski and they are available in Achuthan et al. (1978a).

Chapter V deals with some special structured (n/m/F/f) problems. In section 5.1 we briefly mention the known special structured flow-shop problems with polynomial time complex algorithms. Under section 5.2 we define what we call aggregate backward (forward) cumulative dominance conditions. Using these conditions we prove certain interesting results on the shape of critical paths for an arbitrary non-delay permutation schedule. Under section 5.3 we develop new special structured flow-shop problems using the results of section 5.2 and for these problems we provide polynomial time complex algorithms. The material of this chapter is a joint work with T.S. Arthanari, which is still in the manuscript stage.

CHAPTER I

FLOW-SHOP PROBLEM : DEFINITIONS AND NOTATION

1.0 Scheduling Problem :

There are l tasks to be processed on m machines. The l tasks are partitioned into n sets $J_1, \ldots, J_n$ where set J_j is the set of tasks comprising job j. The set of jobs is denoted by $N = \{1, \ldots, n\}$. The m machines are denoted by M_k, $1 \leq k \leq m$.

A task schedule for task i consists of a finite non-empty set $\{(I_q, M_q)\}$, where each I_q is a finite time interval during which task i is being processed by machine M_q. A schedule σ consists of a set of task schedules, one for each task.

Suppose task i has associated with it a finite number of time intervals $\{(X_q, Y_q)\}$ in its task schedule. Then the start time s_i and completion time c_i of task i are defined by

$$s_i = \operatorname{Min}_q (X_q)$$

$$\text{and} \quad c_i = \operatorname{Max}_q (Y_q) .$$

The start time S_j and completion time C_j of job j under a schedule σ are defined by

$$S_j = \operatorname{Min} \{ s_i \mid i \in J_j \}$$

$$\text{and} \quad C_j = \operatorname{Max} \{ c_i \mid i \in J_j \} .$$

From among all possible schedules, there is assumed to be a non-empty subset F of feasible schedules. In practice, this set F can be defined by a wide variety of constraints, such as

- Precedence constraints : task i must be completed before task j is started which implies that in the schedule σ $c_i \leq s_j$.
- Conflict constraints : task i and task j cannot be processed simultaneously which implies that the time intervals in the task schedule for task i should be disjoint with the time intervals in the task schedule for task j.
- Change-over constraints : if task j follows task i on machine M_q, task j cannot start until at least t_{ij} units of time after the completion of task i.

By no means the constraints listed are exhaustive. The set F might be defined through many more kinds of constraints.

The objective function f is a real valued (usually non-negative) function defined on the set F of feasible schedules.

The scheduling problem is to choose a σ^* from F such that $f(\sigma^*) \leq f(\sigma)$ for every σ in F.

1.1 Common Objective Functions :

Associated with job j there is a real number r_j called the release time of job j. A feasible schedule σ should satisfy $s_i \geq r_j$ for every task i in J_j. Associated with job j there are a target start time a_j ($\geq r_j$) and a target finish time or due date b_j ($> a_j$). In some of the scheduling problems a feasible schedule is allowed to violate its target start time and due date, with a penalty expressed by its objective function.

In defining the objective functions, we use the start time s_j and the completion time C_j of job j implicitly understanding the schedule σ playing role. Common objective functions studied in scheduling theory are :

- Maximum completion time : $C_{max} = \max_{1 \leq j \leq n} C_j$.

- Maximum flow time : $F_{max} = \max_{1 \leq j \leq n} F_j$

  where flow time of job j = $F_j = C_j - r_j$.

- Maximum lateness : $L_{max} = \max_{1 \leq j \leq n} L_j$.

  where lateness of job j = $L_j = C_j - b_j$.

- Maximum tardiness : $T_{max} = \max_{1 \leq j \leq n} T_j$

  where tardiness of job j = $T_j = \max (0, C_j - b_j)$.

- Maximum earliness : $E_{max} = \max_{1 \leq j \leq n} E_j$

  where earliness of job $j = E_j = \max (0, a_j - S_j)$.

  Note that this definition of earliness is different from that of Conway et al.(1967).

- Maximum penalty on earliness and tardiness :

  $$\max \left\{ g(E_{max}), h(T_{max}) \right\}$$

  where g and h are monotonically non-decreasing continuous functions such that $g(0) = h(0) = 0$.

- Number of tardy jobs : $\left| \left\{ j \mid T_j > 0 \right\} \right|$.

- Weighted completion time : $C_w = \sum_{j=1}^{n} w_j C_j$

  where w_j is the weight associated with job j.

- Similarly weighted flowtime, lateness and tardiness can be defined.

The objective functions we just described are not exhaustive, but they include the objective functions discussed in future chapters of this thesis. Following the notation of Conway et al. (1967), we say that an objective function f is a <u>regular measure of performance</u> if

(i) $f(\sigma)$ can be expressed as a function of the job completion times : $f(\sigma) = f(C_1, C_2, \ldots, C_n)$ where C_j is the completion time of the job j under schedule σ ,

and (ii) $f(C_1, \ldots, C_n) > f(C'_1, \ldots, C'_n)$ only if $C_j > C'_j$ for at least one j, $1 \leq j \leq n$.

It should be noted that all the objective functions listed by us are regular measures of performance except for the maximum earliness and the maximum penalty on earliness and tardiness.

1.2 Flow-shop Problem :

Scheduling problems can be classified into several classes such as Job-shop, Open-shop, Flow-shop, Single machine, Parallel machines etc. For details of the definitions of these classifications we refer to Conway et al.(1967), Baker (1974) and Rinnooy Kan (1976).

In this thesis we will restrict our attention to Flow-shop scheduling problem which is defined through the following assumptions.

A 1. The n jobs and the m machines are known without any ambiguity.

A2. Associated with job j there are exactly m tasks in J_j and i^{th} task of job j can be performed only on machine M_i, $1 \leq i \leq m$.

A3. The i^{th} task of job j can start on machine M_i only after the completion of $(i-1)^{th}$ task of job j on machine M_{i-1}, $2 \leq i \leq m$. Thus the m machines of the flow-shop are arranged in the order $M_1, \ldots, M_m$ according to the technological ordering of the m tasks which is same for every job j.

A4. The machine M_i is continuously available from time ∂_i (≥ 0), for processing, until all the tasks of all the jobs are completed.

A4'. For machine M_i, $1 \leq i \leq m$, we have $\partial_i = 0$.

A5. All the tasks of job j are released at time r_j(≥ 0) for processing.

A5'. We take $r_j = 0$ for each job j, $1 \leq j \leq n$.

A6. The processing time of the i^{th} task of job j on machine M_i is denoted by p_{ji} (≥ 0). Processing time is taken to be finite, deterministic and includes the set up time.

A7. No two distinct jobs can be simultaneously processed on the same machine.

A8. Task splitting is not permitted, that is , once a job j is loaded on a machine M_i, it is continuously processed for duration p_{ji} till finish on that machine.

In a flow-shop problem, a schedule can be represented by a starting time matrix $S_{n \times m} = ((s_{ji}))$ and a job order matrix $Q_{n \times m} = ((q_{ui}))$ where s_{ji} is the starting time of job j on machine M_i and $(q_{1i}, \ldots, q_{ni})^T$ is a permutation of the jobs in N such that q_{ui} is the u^{th} job taken for processing on machine M_i.

Definition 1.2.1 : A pair (S, Q) represents a feasible schedule if and only if for all j and i we have

$$s_{ji} \geq \max\left\{s_{j,(i-1)} + p_{j,(i-1)} \, , \, s_{ki} + p_{ki}\right\} \qquad \ldots \quad (1.2.1)$$

where $q_{ui} = j$; $q_{(u-1),i} = k$; $s_{j0} + p_{j0} = r_j$ and when $u = 1$, $s_{ki} + p_{ki} = \partial_i$.

This definition does not use the assumptions A4' and A5'. For any feasible schedule (S,Q), the completion time matrix is defined by $C_{n \times m} = ((c_{ji})) = S + P$ where $P = ((p_{ji}))$. Note that c_{ji} is the completion time of job j on machine M_i under the feasible schedule (S, Q). Thus the start time S_j and the completion time C_j of job j under the feasible schedule (S, Q) are given by $S_j = s_{j1}$ and $C_j = c_{jm}$.

Definition 1.2.2 : A feasible schedule (S, Q) is called a feasible permutation schedule if all the columns of Q are same. Thus a feasible permutation schedule can be represented by a starting time matrix $S_{n \times m}(\pi)$ where $\pi = (\pi(1), \ldots, \pi(n))$ is a permutation of the jobs in N and the k^{th} row of $S(\pi)$ corresponds to the job $\pi(k)$, $1 \leq k \leq n$. Here π is the order in which all the jobs are processed on all the machines.

Observe that for a feasible permutation schedule $S(\pi)$, the condition (1.2.1) reduces to : for all j and i we have

$$s_{\pi(j),i} \geq \max \left\{ s_{\pi(j),(i-1)} + p_{\pi(j),(i-1)},\ s_{\pi(j-1),i} + p_{\pi(j-1),i} \right\} \quad \ldots \quad (1.2.2)$$

where $s_{\pi(j),0} + p_{\pi(j),0} = r_{\pi(j)}$ and $s_{\pi(0),i} + p_{\pi(0),i} = d_i$.

Definition 1.2.3 : A feasible schedule (S, Q) is called a feasible non-delay schedule if every inequality in (1.2.1) holds as an equality. A feasible permutation schedule $S(\pi)$ is called a feasible non-delay permutation schedule if every inequality in (1.2.2) holds as an equality.

Remark 1.2.4 : Given a job order matrix Q, the corresponding feasible non-delay schedule (S,Q) is unique. Similarly for a given permutation π, the corresponding feasible non-delay permutation schedule $S(\pi)$ is unique.

We make the following additional assumption for the flow-shop scheduling problem.

A9. No passing is allowed, that is, the job order matrix Q should have identical columns. In otherwords we shall consider the set of feasible permutation schedules as the set of feasible schedules.

Following the notation of Conway et al.(1967) we denote a flow-shop scheduling problem with assumptions A1 to A9 inclusive of A4' and A5' by $(n/m/F/f)$ where f represents the objective function to be minimized. When assumption A5' is relaxed we denote the problem by $(n/m/F/r_j \geq 0 / f)$.

1.3 Further Definitions and Notation :

Definition 1.3.1 : Given a $(n/m/F/f)$ problem, we denote by $\left(\sum_{u \in X} w_u M_u , \sum_{v \in Y} w'_v M_v \right)$ the two machine flow-shop problem defined in the following lines. X and Y are subsets of $\{1, \ldots, m\}$. w_u and w'_v are real weights associated with u and v in $\{1, \ldots, m\}$. There are n jobs given by the set N. For every job j in N the processing times on first and second machines are denoted by A_j and B_j respectively, where

$$A_j = \sum_{u \in X} w_u p_{ju}$$

and $$B_j = \sum_{v \in Y} w'_v p_{jv} .$$

Definition 1.3.2 : Given a $(n/2/F/f)$ problem, with processing times of job j on first and second machines as A_j and B_j respectively, we call $\pi = (\pi(1), \ldots, \pi(n))$ a Johnson's permutation if

$$\min (A_{\pi(i)}, B_{\pi(j)}) \leq \min (A_{\pi(j)}, B_{\pi(i)})$$

for every i and j such that $1 \leq i < j \leq n$.

Remark 1.3.3 Johnson (1954) : In a $(n/2/F/F_{max})$ problem, a Johnson's permutation π is an optimal solution of the problem, even when assumption A4' is relaxed.

Notation 1.3.4 : Let $\pi = (\pi(1), \ldots, \pi(n))$ be an arbitrary permutation of jobs in N and $Q \subseteq N$. The restriction of π to Q is a permutation of jobs in Q, denoted by π_Q and it is obtained from π by dropping the jobs not in Q.

Remark 1.3.5 Szwarc (1974) : Let π be a Johnson's permutation of a $(n/2/F/f)$ problem. Let $Q \subseteq N$. Then π_Q is a Johnson's permutation of the $(|Q|/2/F/f)$ problem where this problem is obtained from the $(n/2/F/f)$ problem restricting our attention to the jobs in Q.

Definition 1.3.6 : A permutation $\sigma = (\sigma(1), \ldots, \sigma(q))$ of a subset of jobs in N is called a partial (complete) permutation if $q < (=) n$. Without ambiguity in usage, we shall denote the set of jobs in a partial permutation σ by σ itself. Given a partial permutation σ, we call $\sigma\pi$ a completion of σ if π is a permutation of jobs in $\bar{\sigma} = N - \sigma$.

Notation 1.3.7 : In a $(n/m/F/f)$ problem, given a partial or complete permutation σ, let $t(\partial; \sigma; k)$ denote the completion time of the last job in σ on machine M_k under the feasible non-delay permutation

schedule σ where $\partial = (\partial_1, \ldots, \partial_m)$ is a m-tuple giving the earliest times the machines are available for processing the jobs in σ. Whenever $\partial = (0, \ldots, 0)$ we shall denote $t(\partial ; \sigma ; k)$ simiply by $t(\sigma; k)$.

Remark 1.3.8 : It is well known from Conway et al.(1967) that

$$t(\sigma j; k) = \max \{t(\sigma j; k-1), t(\sigma; k)\} + p_{jk}$$

where $j \notin \sigma$ and $t(\emptyset ; k) = 0 = t(\sigma j; 0)$.

Notation 1.3.9 : In a $(n/m/F/r_j \geq 0/f)$ problem, for a partial or complete permutation σ, we extend the notation 1.3.7. Let $t(\partial ; r; \sigma; k)$ denote the completion time of the last job in σ on machine M_k under the feasible non-delay permutation schedule σ, where ∂ and r are tuples giving the machine available and job release times respectively.

Definition 1.3.10 Szwarc (1971) : Let σ be a partial permutation. Let i and j be jobs such that $i \neq j$ and $i, j \notin \sigma$. Let $\Delta_k = t(\sigma i j ; k) - t(\sigma j ; k)$ for $1 \leq k \leq m$. We say σij dominates σj through Szwarc's dominance criteria (SDC) if

$$\Delta_{k-1} \leq \Delta_k \leq p_{ik} \quad \text{for} \quad 2 \leq k \leq m.$$

Remark 1.3.11 Szwarc (1971) : Let σij dominate σj through SDC. Then

$$t(\sigma ij \pi_1 \pi_2 ; m) \leq t(\sigma j \pi_1 i \pi_2 ; m)$$

where π_1 and π_2 are arbitrary partial permutations such that $\pi_1 \cap \pi_2 = \emptyset$ and $\pi_1 \cup \pi_2 = N - \sigma \cup \{i, j\}$. Thus SDC can be used as an elimination criterion for eliminating the partial permutation σj in the branch-and-bound search procedure for the $(n/m/F/F_{max})$ problem.

Definition 1.3.12 : In a $(n/m/F/r_j \geq 0/f)$ problem, given a partial or complete permutation $\sigma = (\sigma(1), \ldots, \sigma(q))$ we define the following critical path network. This network is represented in a matrix form with $(q+1)$ rows and $(m+1)$ columns as in Figure 1.3.1. Each cell (j,i) of this network represents an activity of duration :

Zero (dummy activity) if $j=0$, $i=0$;

δ_i (the time machine M_i is available) if $j=0$, $1 \leq i \leq m$;

$r_{\sigma(j)}$ (the release time of job $\sigma(j)$) if $1 \leq j \leq q$, $i=0$;

and

$p_{\sigma(j),i}$ (the processing time of job $\sigma(j)$ on machine M_i) if $1 \leq j \leq q$, $1 \leq i \leq m$.

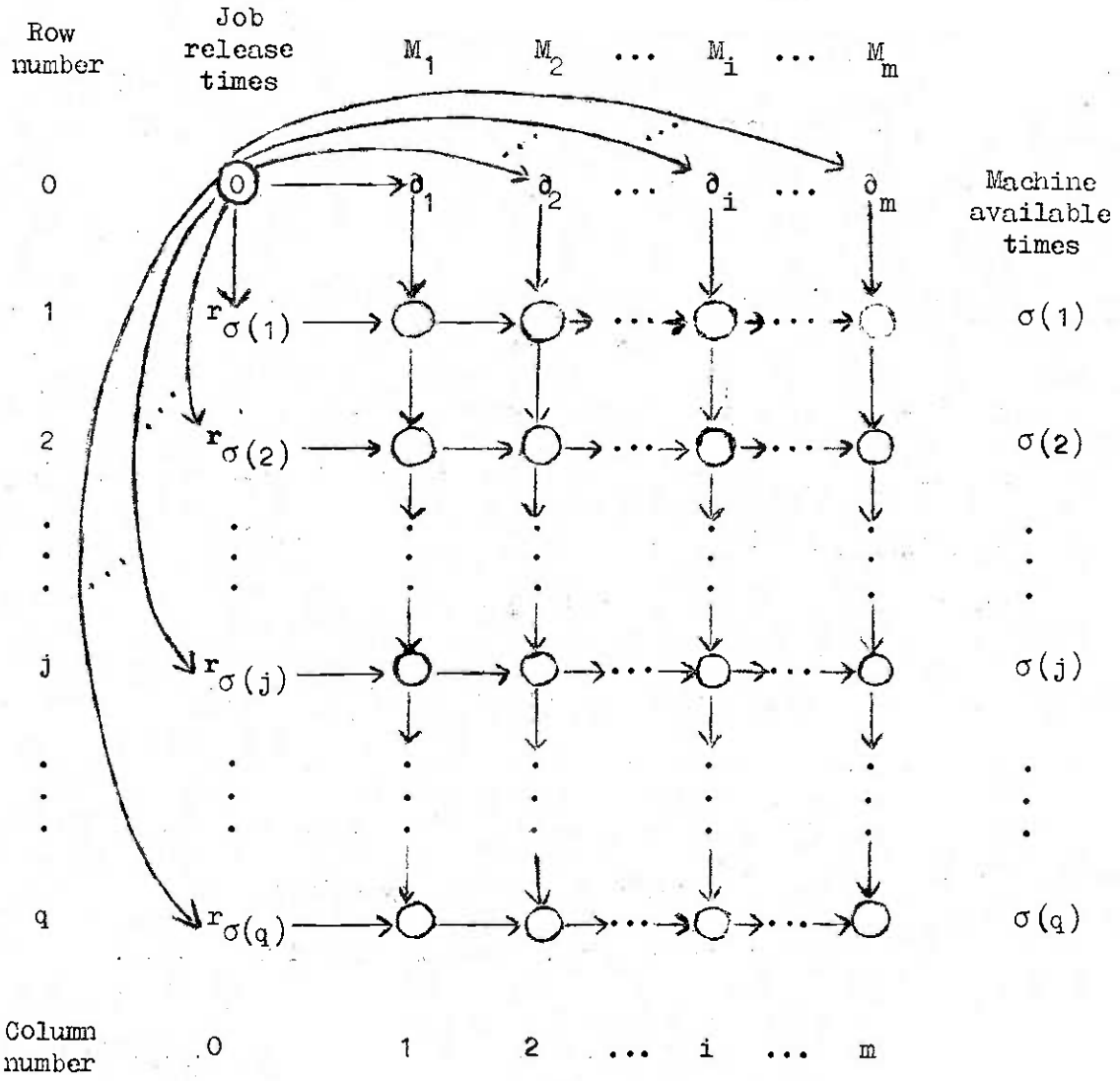


Figure 1.3.1 : Critical path network of σ.

Each cell (j,i) has its immediate successor cells as :

(j,k), $1 \leq k \leq m$ and (l,i), $1 \leq l \leq q$ if j=0 and i=0;

(j+1,i) if j=0 and $1 \leq i \leq m$;

(j,i+1) if $1 \leq j \leq q$ and i=0;

$(j,i+1)$ and $(j+1,i)$ if $1 \leq j \leq q-1$ and $1 \leq i \leq m-1$;

$(j+1,i)$ if $1 \leq j \leq q-1$ and $i = m$;

and $(j,i+1)$ if $j=q$ and $1 \leq i \leq m-1$.

Since the cell $(0,0)$ corresponds to dummy activity, note that all the activities corresponding to the cells in column zero and row zero can be started simultaneously at time zero.

Definition 1.3.13 : Consider the critical path network of a partial or complete permutation $\sigma = (\sigma(1), \ldots, \sigma(q))$. A sequence γ of cells is called a segment if each cell (j,i) of γ (except the last) is followed by one of its immediate successor cell. The length of the segment γ is defined as the sum of the activity durations corresponding to the cells in γ.

In the critical path network of Figure 1.3.1 note that any forward movement from a cell (j,i) can be only of either right-hand or downward turn. Using this some times it is convenient to describe a segment γ by a sequence of R and D symbols that indicate its right-hand and downward turns. For example Figure 1.3.2. illustrates a DRDR segment.

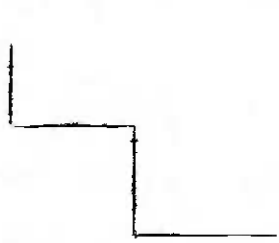


Figure 1.3.2 : DRDR segment.

Definition 1.3.14 : A segment γ with first cell as $(0,0)$ and last cell as (j,m) is called a j-path of the network and its length is the length of the segment γ. A j-critical path of the network is a longest j-path of the network.

Remark 1.3.15 : In a $(n/m/F/r_j \geq 0 / f)$ problem, for any partial or complete permutation $\sigma = (\sigma(1), \ldots, \sigma(q))$, $t(\partial; r; \sigma; m)$ is given by the length of a q-critical path of the critical path network of σ.

Remark 1.3.16 : In a $(n/m/F/f)$ problem, for a partial or complete permutation $\sigma = (\sigma(1), \ldots, \sigma(q))$ the corresponding critical path network is given by dropping the activities in row zero and column zero of Figure 1.3.1. Thus in this case the activity corresponding to the cell $(1,1)$ alone can be started at time zero. Further $t(\sigma; m)$ is given by the length of a q-critical path of this network. It should be observed that our definitions of segment and q-critical path reduce to the definitions of segment and critical path respectively in a $(n/m/F/F_{max})$ problem given by Szwarc (1978).

Definition 1.3.17 : Given a $(n/m/F/f)$ problem, we define a reverse flow-shop problem denoted by $(n/m^R/F/f)$ which has the same set of jobs, machines and the processing times. But the technological ordering of the machines is $M_m, M_{m-1}, \ldots, M_1$ instead of $M_1, \ldots, M_m$.

Definition 1.3.18 : Given a permutation $\sigma = (\sigma(1), \ldots, \sigma(n))$, the reverse permutation σ^R of σ is defined by

$$\sigma^R = (\sigma(n), \sigma(n-1), \ldots, \sigma(1)).$$

Remark 1.3.19 Szwarc (1971) : Let the k^{th} machine of $(n/m^R/F/f)$ problem be denoted by $M_{k'}$ where $k' = m-k+1$. For any arbitrary complete permutation σ in the $(n/m^R/F/f)$ problem, let $t^R(\sigma; k')$ denote the completion time of the last job in σ on machine $M_{k'}$ under the feasible non-delay permutation schedule σ. It is easy to verify that for any arbitrary complete permutation σ we have

$$t(\sigma; m) = t^R(\sigma^R; m').$$

Thus any optimal permutation π of the problem $(n/m/F/F_{max})$ gives an optimal permutation π^R of the problem $(n/m^R/F/F_{max})$.

Definition 1.3.20 : We call (σ_1, σ_2) a two sided partial permutation if σ_1 and σ_2 are partial permutations such that $\sigma_1 \cup \sigma_2 \subset N$, $\sigma_1 \cap \sigma_2 = \emptyset$ and $|\sigma_1| + |\sigma_2| < n$. Note that in a two sided partial permutation (σ_1, σ_2) the first $|\sigma_1|$ positions are fixed; the last $|\sigma_2|$ positions are fixed and the middle $(n-|\sigma_1|-|\sigma_2|)$ positions are not fixed. $\sigma_1 \pi \sigma_2$ is called a completion of the two sided partial permutation (σ_1, σ_2) if π is a permutation of the jobs in $N-(\sigma_1 \cup \sigma_2)$.

1.4 Lower Bounds in a $(n/m/F/F_{max})$ Problem :

Given a partial permutation σ, we need lower bounds on $F_{max}(\sigma\pi)$ for all possible completions $\sigma\pi$ of σ while solving the $(n/m/F/F_{max})$ problem by a branch-and-bound search procedure. In the literature a number of lower bounds have been suggested. Lageweg et al. (1978) gives a fine classification scheme for lower bounds that generates most of the known bounds in the literature. Here we introduce the notation of their classifications.

In a $(n/m/F/F_{max})$ problem given a partial permutation σ we define several flow-shop problems, as explained below, for the jobs in $\bar{\sigma} = N - \sigma$. Choose any two machines M_u and M_v such that $1 \leq u \leq v \leq m$. For machine M_i, $1 \leq i \leq m$ and $i \neq u, v$, relax the assumption A7 to say that the machine M_i can perform simultaneously any number of jobs. Thus these machines M_i, $i \neq u$ and v become non-bottleneck machines whereas machines M_u and M_v remain as bottleneck machines. Now the non-bottleneck machines can be replaced by at most three non-bottleneck machines denoted by $M^*_{.u}$, M^*_{uv} and $M^*_{v.}$ where processing times of job j in $\bar{\sigma}$ are defined as :

$$q_{j.u} = \max_{1 \leq l \leq u} \left\{ t(\sigma; l) + \sum_{k=l}^{u-1} p_{jk} \right\} ,$$

$$q_{juv} = \sum_{k=u+1}^{v-1} p_{jk} ,$$

and $q_{jv.} = \sum_{k=v+1}^{m} p_{jk}$.

This gives a flow-shop problem of scheduling jobs in $\bar{\sigma}$ on machines $M^*_{.u}$, M_u, M^*_{uv}, M_v, $M^*_{v.}$ in that order where again F_{max} is to be minimized. When u=v this problem involves at most three machines including the bottleneck machine M_u.

Any lower bound of the optimal F_{max} of this flow-shop problem provides a valid lower bound on F_{max} ($\sigma\pi$) for all possible completions $\sigma\pi$ of σ in the $(n/m/F/F_{max})$ problem. Any such new flow-shop problem can be characterized by a string $\curvearrowright$ of at most five symbols from $\{ \square , 0, * \}$ where

- $\square$ indicates a bottleneck machine;
- 0 indicates a non-bottleneck machine on which the various processing times are taken into account;
- * indicates elimination of a non-bottleneck machine M^*_α by adding $r_\alpha = \min_{j \notin \sigma} \{ q_{j\alpha} \}$ to a lower bound of optimal F_{max} of the remaining problem. Here note that α could be any one of .u, uv, v. .

Now we obtain a lower bound $LB(\sigma; u, v, \curvearrowright)$ by finding the F_{max} value (denoted by $LB^*(\sigma; u, v, \curvearrowright)$) of the optimal solution to the problem on machines M_u and M_v of type $\square$ and possible machines

M_α^* of type O, and adding to it terms r_α for machines M_α^* of type *. If $u \neq v$ $LB^*(\sigma; u, v, \curvearrowright)$ can be strengthened by exploiting the fact that M_v is not available until $t(\sigma; v)$.

Let $Z = \{(u,v) \mid 1 \leq u \leq v \leq m\}$. For any $S \subseteq Z$ we get a lower bound

$$LB(\sigma; S, \curvearrowright) = \max_{(u,v) \in S} \{LB(\sigma; u, v, \curvearrowright)\}.$$

Define $W = \{(u,m) \mid 1 \leq u \leq m-1\}$.

Through experimental investigation Lageweg et al.(1978) established that computationally efficient results were obtained with search algorithms incorporating $LB(\sigma; W, * \square O \square *)$ and the elimination criteria SDC. In some of the future chapters we will use the lower bound $LB(\sigma; W, * \square O \square *)$. A method to compute $LB(\sigma; W, * \square O \square *)$ is described in the following lines.

For every $(u,m) \in W$ note that $* \square O \square *$ actually reduces to $* \square O \square$ since there are only m machines in the original problem. Now solve the three machine flow-shop problem for jobs in $\bar{\sigma}$ on machines M_u, M_{um}^*, M_m (denoted by $\square \; O \; \square$). This problem can be solved (Conway et al.(1967),pages 94-95) by obtaining the Johnson's permutation π of the two machine flow-shop problem where processing times on first and second machines for job j in $\bar{\sigma}$ are defined as

$p_{ju} + q_{jum}$ and $q_{jum} + p_{jm}$ respectively. Evaluate the $F_{max}(\pi)$ for the problem $\square\ O\ \square$ and denote it by $LB^{*}(\sigma; u, m, *\square O\square *)$. To this add $r_{.u}$ and get $LB(\sigma; u, m, *\square O\square *)$. Now $LB(\sigma; W, *\square O\square *)$ is easily obtained from its definition.

.5 Computational Complexities :

For years, optimization problems have been studied with the view to provide algorithms for solving them. The subject of computational complexities has developed significantly in the process of understanding the computational efficiency of an algorithm. For rigorous definitions of the terms and concepts related to computational complexities, refer to Aho et al.(1974), Garey et al.(1979) and Lenstra et al.(1978). For recent results on computational complexities of the scheduling problems refer to Garey et al.(1979), Rinnooy Kan (1976) and Ullman (1976). Here we provide lucid definitions of the terms we use, following Horowitz et al.(1978), without formally defining an algorithm.

Given an algorithm to solve a problem our interest is to study the frequency of execution of each step, assuming the time for one execution of each step to be a constant. More precisely, our interest is to derive an upper bound on this frequency of executions, in the worst case (i.e. when the algorithm is applied to an instance of the problem leading to maximum number of executions of a step). Usually these bounds are expressed using the following 'O' notation.

Definition 1.5.1 : $O(f(n))$ denotes the set of all $g(n)$ such that there exist positive constants C, n_0 such that $|g(n)| \leq C f(n)$ for all $n \geq n_0$.

An algorithm we have been talking about has the property that at each step of the algorithm, the next step to be executed is uniquely determined. Such algorithms are termed as deterministic algirithms and they can be executed by any of the computers available in the world. Let us relax this deterministic property and permit an algorithm to choose any outcome out of a limited but specified set of possibilities in some steps of the algorithm. Such steps will be specified by a choice function "Choice (S)" where S gives the set of possible outcomes. Algorithms using this choice function are termed as non-deterministic (n-d) algorithms. Conventionally a n-d algorithm is such that it provides the answer to a question in the form of 'yes' or 'no'. Hence every optimization problem of interest should be framed as a decision problem with 'yes' or 'no' answer, so that we can conceive a n-d algorithm to solve the related decision problem. In fact one can claim that solving an optimization problem is at least as hard as solving the related decision problem. A computer which can execute a n-d algorithm is called a non-deterministic (n-d) computer. The concept of n-d computer is fictitiohs since such a computer is not available in the real world. We assume the following

about a n-d computer. A n-d algorithm terminates with answer 'no' to a decision problem, if and only if, there exist no set of choices leading to an answer 'yes' to the decision problem. In otherwards a n-d computer is capable of making correct choices in a n-d algorithm whenever the answer to a decision problem is 'yes'. Further we assume that a n-d computer is capable of making shortest sequence of choices that leads to an answer 'yes'. For a decision problem with answer 'no', a n-d algorithm is capable of terminating with answer 'no' in one unit of time.

We use n as a uniform parameter to measure complexity and usually we will take n as the length of the input to the algorithm.

Definition 1.5.2 : The time required by a n-d algorithm performing on any given input is the minimum number of steps needed to reach an answer 'yes' if there exists a sequence of choices leading to such an answer. A n-d algorithm is of complexity $O(f(n))$ if for all inputs of size n, $n \geq n_o$, that result in an answer 'yes' the time required is at most $C.f(n)$ for some constants C and n_o.

Definition 1.5.3 Satisfiability problem : Let $x_1, x_2, \ldots,$ denote boolean variables (i.e. they take values true or false). Let $\bar{x}_i$ denote the negation of x_i. A literal is either a variable or its negation. A formula in the propositional calculus is an expression

that can be constructed using literals and the operations and and or . An example of such formulas is $(x_1 \wedge x_2) \vee (x_3 \wedge \bar{x}_4)$ where $\vee$ denotes or and $\wedge$ denotes and. The satisfiability problem is to determine if a formula is true for any assignment of truth values to the variables.

Definition 1.5.4 : An algorithm is of polynomial time complexity if its computing time is $O(p(n))$ for every input of size n and some fixed polynomial $p(\ )$.

Definition 1.5.5 : P is the set of all decision problems solvable by a deterministic algorithm in polynomial time. NP is the set of all decision problems solvable by a n-d algorithm in polynomial time.

Remark 1.5.6 Cook (1971) : Satisfiability problem is in P if and only if P = NP.

Definition 1.5.7 : If L_1 and L_2 are problems, L_1 reduces to L_2 (written as $L_1 \propto L_2$) if and only if there is a way to solve L_1 by a deterministic polynomial time complex algorithm using a deterministic algorithm that solves L_2 in polynomial time. Note that '$\propto$' is a transitive relation.

Definition 1.5.8 : A problem L is NP-hard if and only if satisfiability $\propto$ L. A problem L is NP-complete if and only if it is NP-hard and L $\in$ NP.

Remark 1.5.9 : Generally most of the optimization problems reworded as decision problems would be trivially in NP. Hence to prove that the decision problem is NP-complete, we need to prove L $\propto$ the decision problem where L is a known NP-complete problem.

Remark 1.5.10 : In view of Cook's result in Remark 1.5.6. and the tremendous futile effort by many people it is very unlikely that a problem in NP will possess polynomial time complex deterministic algorithm. The reason for our interest in polynomial time complex algorithms lies in the difficulties faced otherwise (for example, in the case of exponential function) when n, the input length becomes large.

CHAPTER II

$(n/3/F/F_{max})$ PROBLEM

2.0 Introduction :

The three machine flow-shop scheduling problem was first considered by Johnson (1954) and he solved two special cases of the problem. Wagner and Story (1963) gave an integer programming formulation of the $(n/3/F/F_{max})$ problem. Lomnicki (1965) suggested a branch-and-bound procedure to solve this problem. Ignall and Schrage (1965) also proposed a branch-and-bound procedure to solve the $(n/3/F/F_{max})$ problem. A good account of the basic work done on this problem is available in the books by Conway et al(1967), Baker (1974) and Rinnooy Kan (1976).

Garly et al(1976) proved that $(n/3/F/F_{max})$ problem is NP-complete and hence it is very unlikely that a polynomial time complex algorithm could exist for this problem. Thus it becomes important to characterise special cases with polynomial time complex algorithms and/or develop heuristic rules which work satisfactorily in the general set up. Many authors have contributed in these two aspects of the $(n/3/F/F_{max})$ problem and we will cite their references in sections 2.1 and 2.2. In this chapter we discuss four new special cases of the $(n/3/F/F_{max})$ problem under section 2.1. Subsequently in section 2.2 we suggest six new heuristic rules and compare their performances with the known heuristic rules in the literature.

Out of the assumptions listed in section 1.2, we make only the assumptions A1 to A7 inclusive of A4' and A5'. Under these assumptions we know that (Conway et al(1967)) it is enough to consider only the non-delay permutation schedules while solving a three machine flow-shop problem with the objective to minimize F_{max}. We deviate from the notation introduced in Chapter I and follow the original notation of Johnson (1954) for the $(n/3/F/F_{max})$ problem. $N = \{1, \ldots, n\}$ is the set of jobs. For each job j in N, A_j, B_j and D_j are the processing times (≥ 0) on the machines M_1, M_2 and M_3 respectively.

$(n/3/F/F_{max})$ problem is : Find a permutation π^* which minimizes $F_{max}(\pi)$ over the set of all permutations $\pi = (\pi(1), \ldots, \pi(n))$ of the jobs in N where $F_{max}(\pi)$ is the length of the n-critical path (Definition 1.3.14.) of the critical path network of π.

2.1 Special Cases with Polynomial Time Complex Algorithms :

The Table 2.1.1 describes the known special cases of the $(n/3/F/F_{max})$ problem for which polynomial time complex algorithms have been proposed. The exact description of the algorithms can be obtained from the respective references.

TABLE 2.1.1 : Known Special Cases of the $(n/3/F/F_{max})$ Problem

Case	Description of the conditions	Reference	Remarks
1	either $B_i \leq A_j$ $\forall$ $i \neq j$ or $B_i \leq D_j$ $\forall$ $i \neq j$	Johnson (1954)	Burdyuk(1969), Arthanari (1974) and Grabowski et al. (1975) generalised this case to larger number of machines.
2	Any one of the following three conditions holds. (i) $B_i \leq A_j$, $B_i+D_i \leq A_j+B_j$ for all $i \neq j$. (ii) $A_i \leq B_j$, $D_i \leq B_j$ for all $i \neq j$. (iii) $A_i + B_i \leq B_j + D_j$, $B_i \leq D_j$ for all $i \neq j$.	Grabowski et al. (1975)	Szwarc(1968) solved the case 2(ii). Grabowski et al. (1975) deals with a generalisation of this case to larger number of machines. Burdyuk (1969) and Arthanari (1974) solved certain generalisations which imply the conditions of Grabowski et al(1975) in the case of larger number of machines.
3	$B_i \geq A_j$ $\forall$ $i \neq j$	Arthanari et al. (1971)	Szwarc(1974) and (1974a) streamlined the algorithm of Arthanari et al(1971) under this case.
4	$B_i \geq D_j$ $\forall$ $i \neq j$	- do -	- do -
5	Let π be an optimal permutation of the two machine flow-shop problem (M_1+M_2, M_2+M_3) with maximum flow as $t(\pi)$. Let $F_{max}(\pi) = t(\pi) - \sum_{i=1}^{n} B_i$.	Szwarc (1974)	π is an optimal permutation of the $(n/3/F/F_{max})$ problem.

contd.../

TABLE 2.1.1 (continued)

Case	Description of the conditions	Reference	Remarks
6	For $i \neq j$, define $\alpha = \min(A_i, B_j) - \min(A_j, B_i)$, $\beta = \min(B_i, D_j) - \min(B_j, D_i)$. for all $i \neq j$, $\alpha\beta \geq 0$.	Burns et al. (1975)	Szwarc(1977) modified Burns et al(1975) algorith Further Szwarc's(1978) case "$B_i = B \; \forall i$ and $\exists \pi \ni$ $A_{\pi(1)} \leq \cdots \leq A_{\pi(n)}$ and $D_{\pi(1)} \geq \cdots \geq D_{\pi(n)}$" implies the conditions of this case. In fact $B_i \geq \max(A_i, D_i) \; \forall$ i impli the conditions of this ca Achuthan(1977) generalise this case to larger numbe of machines.
7	$B_i \leq \min(A_i, D_i) \; \forall$ i	Szwarc (1977a)	

Let $(l_1, \ldots, l_n)$, $(i_1, \ldots, i_n)$ and $(k_1, \ldots, k_n)$ be sequences of 1, 2, ..., n such that $A_{l_1} \leq \cdots \leq A_{l_n}$, $B_{i_1} \leq \cdots \leq B_{i_n}$ and $D_{k_1} \leq \cdots \leq D_{k_n}$ respectively. The new special cases will be defined using these sequences.

Case I : $\sum_{r=1}^{q} B_{i_r} \geq \sum_{r=0}^{q-1} A_{l_{n-r}}$ for the smallest possible q .

This condition implies that the sum of any q B_i's is greater than or equal to the sum of any q A_j's. When q=1, this case reduces to case 3

of Table 2.1.1. The following Example 2.1.1 is a $(4/3/F/F_{max})$ problem satisfying the conditions of this case with q=2. It is easy to verify that Example 2.1.1 does not satisfy the conditions of any of the cases from 1 to 7 in Table 2.1.1.

Example 2.1.1 :

Job	Processing times on		
	M_1	M_2	M_3
1	4	15	5
2	6	11	10
3	8	12	6
4	10	7	9

Given any permutation $\pi = (\pi(1), \ldots, \pi(n))$, let Figure 2.1.1 represent the critical path network (defined in 1.3.12) with reference to π and the $(n/3/F/F_{max})$ problem.

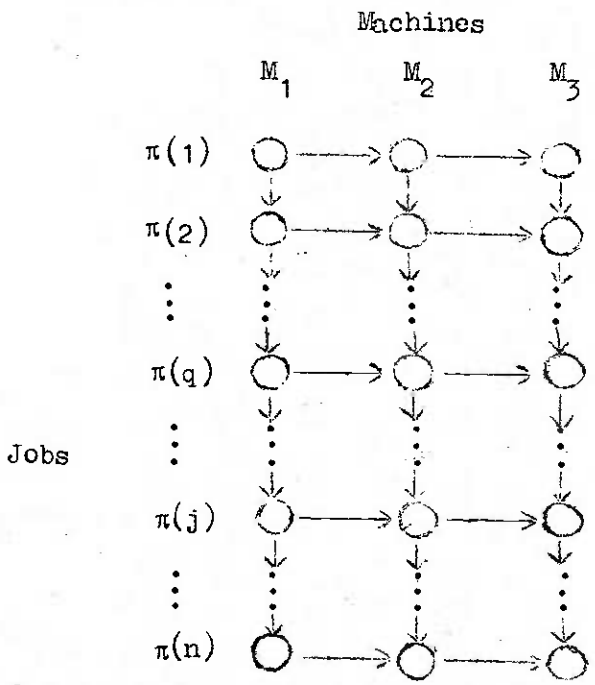


Figure 2.1.1 : Critical path network of π for the $(n/3/F/F_{max})$ problem.

Proposition 2.1.1 : Under case I, for any i and j such that $q+1 \leq i \leq j \leq n$, the DRD segment (Definition 1.3.13) connecting the end cells (1,1) and (j,2) through (i,1) and (i,2) is of smaller length than the RD segment connecting the end cells (1,1) and (j,2).

Proof : Length of the DRD segment connecting end cells (1,1) and (j,2) through (i,1) and (i,2) $= \sum_{r=1}^{i} A_{\pi}(r) + \sum_{r=i}^{j} B_{\pi}(r)$

$$\leq A_{\pi(1)} + \sum_{r=1}^{i-1} B_{\pi}(r) + \sum_{r=i}^{j} B_{\pi}(r), \text{ (by case I)}$$

$$=$$ Length of the RD segment connecting end cells (1,1) and (j,2).

This completes the proof. //

Proposition 2.1.2 : Under case I, for j such that $q+1 \leq j \leq n$, any j-critical path of π (Definiti n 1.3.14) should make a right turn in column 'one' at a cell (i,1) for some i such that $1 \leq i \leq q$.

Proof : Suppose the contrary. Let a j-critical path be a DRDRD segment with end cells (1,1) and (j,3), first right turn at (i,1) and second right turn at (1,2) where $q+1 \leq i \leq 1 \leq j$. Then the length of this j-critical path is equal to the sum of the lengths of the DRD segment connecting end cells (1,1) and (1,2) and the D segment connecting end cells (1,3) and (j,3). Now by Proposition 2.1.1, the DRD segment has

length less than the RD segment connecting end cells (1,1) and (1,2). Hence this j-critical path has length smaller than the RDRD segment connecting end cells (1,1) and (j,3) with its second right turn at (1,2). This contradicts the definition of a j-critical path. Hence the proposition is proved.//

Theorem 2.1.3 : Let $\alpha = (\alpha(1), \ldots, \alpha(n))$ be a Johnson's permutation (Definition 1.3.2) of the two machine flow-shop problem (M_2, M_3). Let Q be an arbitrary subset of N such that $|Q| = q$. Then under case I,

(a) From among the set of all permutations of the n jobs with first q jobs restricted to Q, we need to consider only the permutations of the form $(\sigma, \alpha_{\bar{Q}})$ where σ is an arbitrary permutation of the jobs in Q, $\bar{Q} = N - Q$ and $\alpha_{\bar{Q}}$ is the restriction of α to $\bar{Q}$ (Notation 1.3.4).

(b) Define $I = \{i_1, \ldots, i_q\} \cup \{l_n, l_{n-1}, \ldots, l_{n-q+1}\}$.

Let $\bar{I} = N - I$. If $Q \subseteq \bar{I}$ then from among the set of all permutations of the n jobs with first q jobs restricted to be in Q, we need to consider only the permutations

$(u(j), u(1), \ldots, u(j-1), u(j+1), \ldots, u(q), \alpha_{\bar{Q}})$, $1 \leq j \leq q$

and $A_{u(j)} < A_{u(1)}$ for $j \neq 1$ where

$$\alpha_Q = (u(1), \ldots, u(q)).$$

Proof : Part(a) : For an arbitrary permutation σ of jobs in Q, let $\sigma\pi$ be a completion of σ (Definition 1.3.6). From Proposition 2.1.2 a n-critical path of $\sigma\pi$ should pass through either cell $(q,2)$ or cell $(q,3)$. Therefore finding an optimal completion of σ reduces to solving a two machine flow-shop problem (M_2, M_3) for the jobs in $\bar{Q}$ given that machines M_2 and M_3 are available only at times $\partial_2 = t(\sigma; 2)$ and $\partial_3 = t(\sigma; 3)$ respectively. Now by Remarks 1.3.3 and 1.3.5 $\alpha_{\bar{Q}}$ is an optimal solution of this two machine flow-shop problem (M_2, M_3). Thus the proof of Part(a) is complete.

Part(b) : Let v and v' belong to $\bar{I}$. From the conditions of case I, we have,

$$\sum_{r=1}^{q-1} B_{i_r} < \sum_{r=0}^{q-2} A_{l_{n-r}} \quad \text{and} \quad \sum_{r=1}^{q-1} B_{i_r} + B_v \geq \sum_{r=0}^{q-2} A_{l_{n-r}} + A_{v'} .$$

Therefore $B_v > A_{v'}$ for any v and v' in $\bar{I}$.

Consider $\pi = (\sigma, \alpha_{\bar{Q}})$ where σ is a permutation of jobs in Q. Using $B_v > A_{v'}$ in Figure 2.1.1 with reference to π note that : For any l such that $1 \leq l \leq n$, a DRD segment connecting the end cells $(1,1)$ and $(l,2)$ is of smaller length than the RD segment connecting the end cells $(1,1)$ and $(l,2)$. Consequently the length of a n-critical path of π is equal to the sum of $A_{\sigma(1)}$ and the length of a n-critical path of π with reference to the two machine flow-shop problem (M_2, M_3). Now Part(b) follows from the properties (Remark 1.3.5) of the Johnson's permutation α. //

Using Theorem 2.1.3, we present a branch-and-bound procedure to solve the $(n/3/F/F_{max})$ problem under case I. After introducing the required notation we present the procedure in a stylistic convention followed by Knuth (1973), Kernighan and Plauger (1974) and others. The notation in general follow Rinnooy Kan (1976).

- Throughout the procedure, the best solution π^* found so far provides an upper bound $F_{max}(\pi^*)$ on the value of the optimal solution.

- A branching rule b associates with a partial permutation $\sigma = (\sigma(1), \ldots, \sigma(s))$ a family $b(\sigma)$ of partial permutations where $b(\sigma) = \{\sigma j \mid j \notin \sigma\}$. Note $|b(\sigma)| = n - s$. The elements of $b(\sigma)$ are called descendants of σ.

- A bounding rule lb associates with a partial permutation $\sigma = (\sigma(1), \ldots, \sigma(s))$ a lower bound $lb(\sigma) \leq F_{max}(\sigma\pi)$ for every $\sigma\pi$, a completion of σ, where

$$lb(\sigma) = \begin{cases} \max\{lb(\sigma(1), \ldots, \sigma(s-1)), lb^1(\sigma)\}, & \text{if } s \geq 1 \\ lb^1(\sigma), & \text{if } s=0 \end{cases}$$

and $lb^1(\sigma) = LB(\sigma; W, * \square 0 \square *)$ defined in section 1.4 of Chapter I. Note that $lb^1(\sigma)$ itself is a lower bound. Elimination of σ occurs if $lb(\sigma) \geq F_{max}(\pi^*)$.

- A dominance rule d associates with a partial permutation σ a subset of $b(\sigma)$ denoted by $d(\sigma)$. Define $d(\sigma) = \{ \sigma j \mid \sigma ij$ dominates σj through SDC $\}$ (see Definition 1.3.10.). To construct $d(\sigma)$ we have to check SDC for each pair (i,j) such that $i \neq j$ and $i,j \in \bar{\sigma}$. Dominance cycles can occur and have to be avoided while constructing $d(\sigma)$. Note that the partial permutations in $d(\sigma)$ can be eliminated.

- A predicate ρ associates with a partial permutation $\sigma = (\sigma(1), \ldots, \sigma(s))$ a true or false value. Define

$$\rho(\sigma) = \begin{cases} \text{true}, & \text{if } |\sigma| = q, \\ \text{false}, & \text{if } |\sigma| < q. \end{cases}$$

When $\rho(\sigma)$ is true, $\mu = (\sigma, \alpha_{\bar{\sigma}})$ is a complete permutation for which $F_{max}(\mu)$ is evaluated. Here $\alpha_{\bar{\sigma}}$ is the restriction of α to $\bar{\sigma}$ where α is defined in Theorem 2.1.3. Improvement of π^* occurs if $F_{max}(\pi^*) > F_{max}(\mu)$.

- A search strategy chooses a partial permutation σ from the collection of generated partial permutations which are so far neither eliminated nor led to branching. We use frontier search where a partial permutation with minimal lower bound is selected for further branching.

During the search, parameters na and nb count the number of partial permutations that are eliminated and that lead to branching respectively. We define the operation " : f $\in$ " in the statement " s : f $\in$ S" to mean that s : = s* where $f(s^*) = \min_{s \in S} f(s)$. Note that ' :$\in$ ' indicates an arbitrary choice. Let X denote the set of all partial permutations. We assume that "Procedure JP (A,B,N, π)" gives a Johnson's permutation π of the $(n/2/F/F_{max})$ problem where processing times of job i on first and second machines are given by A_i and B_i respectively. In the search procedure we use Procedure JP (A,B,N, π) as a subroutine. It is well known from the literature (Rinnooy Kan (1976)) that Procedure JP (A,B,N, π) can be stated such that its computational complexity is O(n log n) (see Definitions 1.5.1, 1.5.2 and 1.5.4).

We have presented the branch-and-bound procedure for case I in a simple way, assuming that a set of computations are done at the root node ($\emptyset$) and the remaining at the node σ when it is generated. Further, with reference to a generated node σ, we assume that, we store $|\sigma|$; $t(\sigma; k)$, $1 \leq k \leq 3$; $lb^1(\sigma)$; $lb(\sigma)$; σ and $\bar{\sigma}$. This storage helps in simplifying computations while branching from σ.

Procedure 'B & B for case I' ($\emptyset$,A,B,C,F_{max}, α, π^*, β, b, lb, lb^1, d, ρ, na, nb)

1. Begin Local Y;Y';Y";$\bar{B} \subset X$; $\bar{D} \subset \bar{B}$; σ, $\gamma \in X$; LB : $X \longrightarrow R$;
2. call 'Procedure JP (A+B, B+D, N, β)';
3. call 'Procedure JP (B, D, N, α)';

4. π^* : $F_{max} \in \{\beta, \alpha\}$; na := nb := 0; Y := $\emptyset$;

5. $LB^1(\emptyset)$:= $lb^1(\emptyset)$; $LB(\emptyset)$:= $lb(\emptyset)$;

6. if $LB(\emptyset) \geq F_{max}(\pi^*)$ then na := 1 else Y := $\{\emptyset\}$;

7. While $Y \neq \emptyset$ do

8. begin $Y' := \{\sigma \mid \sigma : LB \in Y\}$;

9. $\sigma : LB^1 \in Y'$;

10. nb := nb+1; $\bar{B} := b(\sigma)$; $\bar{D} := d(\sigma)$; $Y := (Y - \sigma) \cup (\bar{B} - \bar{D})$;

11. while $\bar{B} - \bar{D} \neq \emptyset$ do

12. begin $\gamma : \in \bar{B} - \bar{D}$; $\bar{B} - \bar{D} := \bar{B} - \bar{D} - \{\gamma\}$;

13. if $\rho(\gamma)$ is true then $\pi^* : F_{max} \in \{\pi^*, \gamma\alpha_\gamma\}$;

14. else $LB^1(\gamma) := lb^1(\gamma)$; $LB(\gamma) := lb(\gamma)$;

15. end

16. $Y'' := \{\sigma' \mid \sigma' \in Y;\ LB(\sigma') \geq F_{max}(\pi^*)\}$;

17. na := na + $|Y''|$; Y := Y - Y'' ;

18. end

19. end 'B & B for case I'.

The computational complexity of the Procedure 'B & B for case I' is analysed in the following. Recollect that $W = \{(1,3), (2,3)\}$ in the definition of lower bound $LB(\sigma ; W, * \square o \square *)$. Note that for a given permutation π the value of $F_{max}(\pi)$ in a $(n/m/F/F_{max})$ problem is computed in $O(mn)$ steps.

Computations performed once at the root node : Obtaining Johnson's permutations β and α of the two machine flow-shop problems $(M_1+M_2,\ M_2+M_3)$ and $(M_2,\ M_3)$ respectively. This involves a procedure in $O(2n \log n)$ steps.

Computations performed at the node corresponding to σ :

(i) When $|\sigma| = s < q$, calculation of $LB(\sigma; W, *\square\ O\ \square *)$ involves, constructing $\beta_{\bar{\sigma}}$ and $\alpha_{\bar{\sigma}}$ and evaluating $F_{max}(\beta_{\bar{\sigma}})$ and $F_{max}(\alpha_{\bar{\sigma}})$ for the flow-shop problems $\square\ O\ \square$ and $\square\ \square$ respectively. This is accomplished in $O((2n+5)(n-s))$ steps.

(ii) When $|\sigma| = s < q$, construction of the set $d(\sigma)$ is performed in $O(3(n-s)(n-s-1))$ steps.

(iii) When $\rho(\sigma)$ is true the construction of $\mu = (\sigma, \alpha_{\bar{\sigma}})$ and calculation of $F_{max}(\mu)$ are performed in $O((n+3)(n-q))$ steps.

Now the number of nodes generated with s jobs fixed can be at most equal to $s\ !\ {}^{n}C_{s}$. Further due to predicate ρ a generated node has at most $(q-1)$ jobs fixed. Let $f(s)$ denote the computational time for the calculations performed with reference to a node with s jobs fixed. Then the total computational time with reference to the nodes generated can be at most equal to $\sum_{s=1}^{q-1} s\ !\ {}^{n}C_{s}\ f(s)$. Further, the time taken for the computations performed with reference to the nodes with q jobs fixed can be at most $q\ !\ {}^{n}C_{q}\ f(q)$. Note that $f(s)$ is a 2^{nd} degree polynomial function of n for all values of $s \leq q$. Thus

for $s \leq q$, $s! \; {}^{n}C_{s} \; f(s)$ is a $(s+2)^{th}$ degree polynomial function of n. Therefore the computational time complexity (Definition 1.5.2) of the procedure 'B & B for case I' is $O(n^{q+2})$. Hence for a fixed q and n much larger than q, we have a polynomial time complex algorithm (Definition 1.5.4) for case I.

Case II : $\sum_{r=1}^{q} B_{i_r} \geq \sum_{r=0}^{q-1} D_{k_{n-r}}$ for the smallest possible q.

This condition implies that the sum of any q B_i's is greater than or equal to the sum of any q D_j's. When q=1, this case reduces to the case 4 of Table 2.1.1. The following $(4/3/F/F_{max})$ example satisfies the conditions of this case with q=2 where-as it does not satisfy the conditions of any of the cases 1 to 7 in Table 2.1.1 and that of case I for $q \leq 3$.

Example 2.1.2 :

Job	Processing times on		
	M_1	M_2	M_3
1	4	15	5
2	12	11	9
3	9	12	6
4	10	7	9

This case reduces to case I for the reverse flow-shop problem $(n/3^{R}/F/F_{max})$ (see Definition 1.3.17). An optimal solution of the $(n/3/F/F_{max})$ problem under this case can be obtained by solving the

$(n/3^R/F/F_{max})$ problem through the procedure of case I (see Remark 1.3.19).

Case III : $\sum_{r=1}^{q} B_{i_r} \geq \sum_{r=0}^{q-1} A_{l_{n-r}}$ for the smallest possible q ;

$\sum_{r=1}^{q'} B_{i_r} \geq \sum_{r=0}^{q'-1} D_{k_{n-r}}$ for the smallest possible q'

and $q + q' \leq n$.

This case is a combination of cases I and II. Further, when $q = q' = 1$, this case reduces to case 2 (ii) of Table 2.1.1. The following example demonstrates that this case with $q = q' = 2$ does not satisfy the conditions of the cases 1 to 7 in Table 2.1.1.

Example 2.1.3 :

Job	Processing times on		
	M_1	M_2	M_3
1	4	15	5
2	6	11	9
3	8	12	6
4	10	7	9

Given any permutation $\pi = (\pi(1), \ldots, \pi(n))$ let Figure 2.1.2 represent the critical path network of π with reference to the $(n/3/F/F_{max})$ problem.

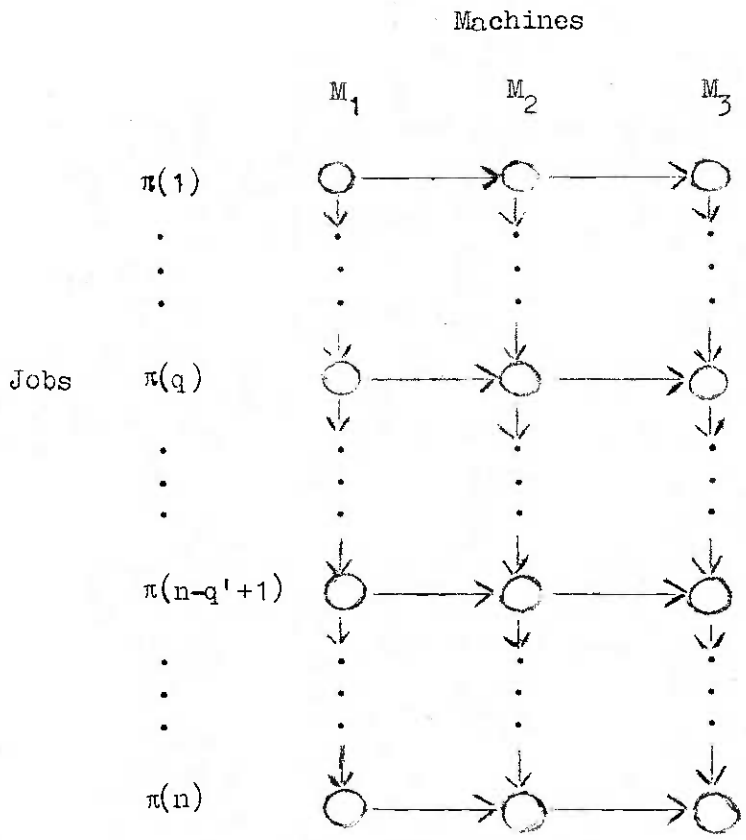


Figure 2.1.2 : Critical path network of π for $(n/3/F/F_{max})$ problem.

Under the condition " $\sum_{r=1}^{q'} B_{i_r} \geq \sum_{r=0}^{q'-1} D_{k_{n-r}}$ ", the following proposition is analogous to proposition 2.1.1 and hence we omit its proof.

Proposition 2.1.4 : Assume $\sum_{r=1}^{q'} B_{i_r} \geq \sum_{r=0}^{q'-1} D_{k_{n-r}}$ for the smallest possible q'. Then for any i and j such that $1 \leq i \leq j \leq n-q'$, the DRD segment connecting the end cells (i,2) and (n,3) through (j,2) and (j,3) is of smaller length than the DR segment connecting the end cells (i,2) and (n,3). //

Proposition 2.1.5 : Under case III, a n-critical path of π should make a right turn in column one at a cell $(i,1)$ for some i such that $1 \leq i \leq q$ and another right turn in column two at a cell $(j,2)$ for some j such that $n - q' + 1 \leq j \leq n$.

Proof : Using Propositions 2.1.1 and 2.1.4, the proof proceeds in the same lines as that of Proposition 2.1.2. //

Theorem 2.1.6 : Let $\beta = (\beta(1), \ldots, \beta(n))$ and $\alpha = (\alpha(1), \ldots, \alpha(n))$ be Johnson's permutations of the two machine flow-shop problems (M_1, M_2) and (M_2, M_3) respectively. Let Q and Q' be arbitrary subsets of N such that $Q \cap Q' = \emptyset$, $|Q| = q$ and $|Q'| = q'$. Then under case III, from among the set of all permutations with first q jobs from the set Q and the last q' jobs from the set Q', it is enough to consider one permutation $(\beta_Q, \sigma, \alpha_{Q'})$ where σ is an arbitrary permutation of the jobs in $N - (Q \cup Q')$.

Proof : Consider an arbitrary permutation π such that the first q jobs and the last q' jobs are from Q and Q' respectively. Invoking Proposition 2.1.5, a n-critical path of π passes through the cells $(q,2)$ and $((n-q'+1), 2)$. Thus the length of a n-critical path of π is the sum of the lengths of :

(i) a q-critical path of $(\pi(1), \ldots, \pi(q))$ in the two machine flow-shop problem (M_1, M_2),

(ii) segment D connecting the end cells $(q+1,2)$ and $(n-q',2)$, and (iii) a q'-critical path of $(\pi(n-q'+1), \ldots, \pi(n))$ in the two machine flow-shop problem (M_2, M_3).

For a fixed Q and Q', use Remark 1.3.5 and observe that,

- β_Q minimizes the length of a q-critical path in (i),
- the length of the segment D in (ii) is a constant = $\sum_{j \in N-(Q \cup Q')} B_j$
- $\alpha_{Q'}$ minimizes the length of a q'-critical path in (iii).

Now Theorem 2.1.6 follows.//

Using Theorem 2.1.6 we suggest a branch and bound procedure for case III. We discuss only the salient features of the branch-and-bound procedure avoiding a formal presentation of it.

- The best solution π^* found so far provides an upper bound $F_{max}(\pi^*)$ on the value of the optimal solution. Initially π^* is taken to be the best among the Johnson's permutations of the two machine flow-shop problems (M_1, M_2), (M_1+M_2, M_2+M_3) and (M_2, M_3). This is performed in $O(3n(3+\log n))$ steps at the root node $(\emptyset, \emptyset)$.

- A branching rule b associates with a two sided partial permutation (σ_1, σ_2) (Definition 1.3.20) a family $b(\sigma_1, \sigma_2)$ of two sided partial permutations where

$$b(\sigma_1, \sigma_2) = \begin{cases} \{(\sigma_1 i, j\sigma_2) \mid i \neq j, i \text{ and } j \notin \sigma_1 \cup \sigma_2\} \text{ if } |\sigma_1| < q, |\sigma_2| < q', \\ \{(\sigma_1 i, \sigma_2) \mid i \notin \sigma_1 \cup \sigma_2\}, \text{ if } |\sigma_1| < q, |\sigma_2| = q', \\ \{(\sigma_1, j\sigma_2) \mid j \notin \sigma_1 \cup \sigma_2\}, \text{ if } |\sigma_1| = q, |\sigma_2| < q'. \end{cases}$$

A node representing a two sided partial permutation (σ_1, σ_2) such that $|\sigma_1| \geq q$ and $|\sigma_2| \geq q'$ will not be generated in our procedure. Note that

$$|b(\sigma_1, \sigma_2)| = \begin{cases} (n - |\sigma_1| - |\sigma_2|)(n - |\sigma_1| - |\sigma_2| - 1) \text{ if } |\sigma_1| < q, |\sigma_2| < q', \\ n - |\sigma_1| - |\sigma_2| \text{ otherwise.} \end{cases}$$

- A bounding rule lb associates with a two sided partial permutation (σ_1, σ_2) a lower bound $lb(\sigma_1, \sigma_2) \leq F_{max}(\sigma_1 \pi \sigma_2)$ for every $\sigma_1 \pi \sigma_2$, a completion of (σ_1, σ_2), where $lb(\sigma_1, \sigma_2)$ is defined in the following lines.

Define (i) $l\partial(\sigma_1, \sigma_2; 1) = t(\sigma_1; 1) + \sum_{i \in \overline{\sigma_1 \cup \sigma_2}} A_i$,

(ii) $l\partial(\sigma_1, \sigma_2; 2) = t(\partial; \beta_{\overline{\sigma_1 \cup \sigma_2}}; 2)$ of Notation 1.3.7 where

$\partial = (t(\sigma_1; 1), t(\sigma_1; 2), t(\sigma_1; 3))$; β is as defined in Theorem 2.1.6 and $\beta_{\overline{\sigma_1 \cup \sigma_2}}$ is the restriction of β to $\overline{\sigma_1 \cup \sigma_2}$ and

(iii) $1\partial(\sigma_1, \sigma_2; 3) = LB(\sigma_1; W, *\square 0\square *)$ defined in section 1.4 of Chapter I with the slight change that the jobs not yet fixed will be $\overline{\sigma_1 \cup \sigma_2}$ instead of $\bar{\sigma}_1$. Now $1\partial(\sigma_1, \sigma_2; k)$, $1 \leq k \leq 3$ defined above is a lower bound of the earliest time the machine M_k is available, after processing all the jobs in $\bar{\sigma}_2$ with the restriction that the jobs in σ_1 are processed first in that order.

Define $lb(\sigma_1, \sigma_2) = t(\hat{\partial}; \sigma_2; 3)$ where

$$\hat{\partial} = (1\partial(\sigma_1, \sigma_2; 1), 1\partial(\sigma_1, \sigma_2; 2), 1\partial(\sigma_1, \sigma_2; 3)).$$

It is easy to verify that $lb(\sigma_1, \sigma_2) \leq F_{max}(\sigma_1 \pi \sigma_2)$ for every $\sigma_1 \pi \sigma_2$, a completion of (σ_1, σ_2). Elimination of (σ_1, σ_2) occurs if $lb(\sigma_1, \sigma_2) \geq F_{max}(\pi^*)$. For a two-sided partial permutation (σ_1, σ_2) let us assume that we store $|\sigma_1|; |\sigma_2|$ $t(\sigma_1; k)$, $1 \leq k \leq 3$; $lb(\sigma_1, \sigma_2)$; (σ_1, σ_2) and $\overline{\sigma_1 \cup \sigma_2}$. Then the computation of $\hat{\partial}$ is performed in $O((3n+8)(n-|\sigma_1| - |\sigma_2|))$ steps. Thus the computation of $lb(\sigma_1, \sigma_2)$ is performed in $O((3n+8)(n-|\sigma_1| - |\sigma_2| + 3|\sigma_2|))$ steps.

- A dominance rule d associates with a two sided partial permutation (σ_1, σ_2) a subset of $b(\sigma_1, \sigma_2)$ denoted by $d(\sigma_1, \sigma_2)$. Given (σ_1, σ_2) let $\beta(r_1)$ (i.e. the job in the r_1^{th} position of β defined in Theorem 2.1.6) be the last job in σ_1 and $\alpha(r_2)$

(i.e. the job in the r_2^{th} position of α defined in Theorem 2.1.6) be the first job in σ_2. Given (σ_1, σ_2), a job $i \in \overline{\sigma_1 \cup \sigma_2}$ is said to satisfy Property P(1) if either $i = \beta(r')$ where $r' < r_1$

or $i = \beta(r')$ where $r' \geq n-q+|\sigma_1| + 2$.

A job $j \in \overline{\sigma_1 \cup \sigma_2}$ is said to satisfy Property P(2)

if either $j = \alpha(r')$ where $r_2 < r'$

or $j = \alpha(r')$ where $r' \leq q' - |\sigma_2| - 1$.

Define,

$$d(\sigma_1,\sigma_2) = \begin{cases} \{(\sigma_1 i, j\sigma_2) \mid i \text{ satisfies P(1) or } j \text{ satisfies P(2)}\} \\ \qquad \text{if } |\sigma_1| < q, |\sigma_2| < q', \\ \{(\sigma_1 i, \sigma_2) \mid i \text{ satisfies P(1)}\} \text{ if } |\sigma_1| < q, |\sigma_2| = q', \\ \{(\sigma_1, j\sigma_2) \mid j \text{ satisfies P(2)}\} \text{ if } |\sigma_1| = q, |\sigma_2| < q'. \end{cases}$$

The two sided partial permutations in $d(\sigma_1,\sigma_2)$ can be eliminated. Given a two sided partial permutation (σ_1,σ_2), $\beta(r_1)$ and $\alpha(r_2)$ can be located in $O(n+2+(n-|\sigma_1| - |\sigma_2|))$ steps and the jobs $j \in \overline{\sigma_1 \cup \sigma_2}$ such that j satisfies either P(1) or P(2) can be collected in $O(2n(n-|\sigma_1| - |\sigma_2|))$ steps. Thus $d(\sigma_1, \sigma_2)$ can be constructed in $O(n+2+(2n+1)(n-|\sigma_1| - |\sigma_2|))$ steps.

- A predicate ρ associates with a two sided partial permutation (σ_1, σ_2) a true or false value. Define

$$\rho(\sigma_1, \sigma_2) = \begin{cases} \text{true}, & \text{if } |\sigma_1| = q \text{ and } |\sigma_2| = q' \\ \text{false}, & \text{otherwise.} \end{cases}$$

When $\rho(\sigma_1, \sigma_2)$ is true, $\mu = (\sigma_1 \pi \sigma_2)$ is an arbitrary completion of (σ_1, σ_2) and this is constructed in $O(n-|\sigma_1| -|\sigma_2|)$ steps. $F_{max}(\mu)$ is computed in $O(3(n-|\sigma_1|))$ steps. Improvement of π^* occurs if $F_{max}(\pi^*) > F_{max}(\mu)$.

- Frontier search selects a two sided partial permutation with minimal lower bound for further branching, from among the partial permutations which have so far been neither eliminated nor led to branching.

Now we shall analyse the computational complexity of this branch and-bound procedure. Before we proceed further observe the following features of the procedure.

(i) A two sided partial permutation (σ_1, σ_2) generated by the procedure is such that either $|\sigma_1| = |\sigma_2| \leq \min(q, q')$

or $|\sigma_1| = q$ and $q < |\sigma_2| \leq q'$ if $q < q'$

or $q' < |\sigma_1| \leq q$ and $|\sigma_2| = q'$ if $q' < q$.

(ii) As a consequence of the dominance rule d, a two sided partial permutation (σ_1, σ_2) generated by the procedure is such that $\sigma_1 = \beta_{\sigma_1}$ and $\sigma_2 = \alpha_{\sigma_2}$. Further the last job in σ_1 will be $\beta(r)$ where $r \leq n-q+|\sigma_1|$ and the first job in σ_2 will be $\alpha(r)$ where $r \geq q'-|\sigma_2|+1$.

Let $N(s_1, s_2)$ denote the number of two sided partial permutations (σ_1, σ_2) generated by the procedure such that $|\sigma_1| = s_1$ and $|\sigma_2| = s_2$. Then we have

$$N(s,s) = {}^{n-q+s}C_s \cdot {}^{n-s-q'+s}C_s = {}^{n-q+s}C_s \cdot {}^{n-q'}C_s \quad \text{if } s \leq \min(q,q'),$$

$$N(q,s) = {}^{n}C_q \cdot {}^{n-q-q'+s}C_s, \quad \text{if } q < s \leq q',$$

$$N(s,q') = {}^{n}C_{q'} \cdot {}^{n-q-q'+s}C_s, \quad \text{if } q' < s \leq q,$$

$$N(s_1,s_2) = 0 \qquad \text{otherwise.}$$

Denote the computational time by $f(s_1, s_2)$, for the calculations performed with reference to a two sided partial permutation (σ_1, σ_2) with $|\sigma_1| = s_1$ and $|\sigma_2| = s_2$. Then the total computational time with reference to the nodes generated can be at most equal to

$$\sum_{s=0}^{\min(q,q')} {}^{n-q+s}C_s \cdot {}^{n-q'}C_s \; f(s,s) + T$$

where,

$$T = \begin{cases} \sum_{s=q+1}^{q'} {}^{n}C_{q} \cdot {}^{n-q-q'+s}C_{s} \cdot f(q,s) & \text{if } q < q' \\ 0 & \text{if } q = q' \\ \sum_{s=q'+1}^{q} {}^{n}C_{q'} \cdot {}^{n-q-q'+s}C_{s} \cdot f(s,q') & \text{if } q > q'. \end{cases}$$

Clearly this computational time is of order $O(n^{q+q'+2})$ since $f(s,s)$, $f(q,s)$ and $f(s,q')$ are of order $O(n^2)$. Thus the computational complexity of this procedure is $O(n^{q+q'+2})$ and hence for fixed q and q' the procedure is of polynomial time complexity. Observe that the procedure of either case I or II is applicable to case III as well. In the worst instance,from the above analysis it is clear that the method of oase III requires more computational time than that of the methods for cases I and II. However it should be observed that, on an average, the method of case III is expected to work better.

Case IV : either $\sum_{r=1}^{q} A_{1_r} \geq \sum_{r=0}^{q-1} B_{i_{n-r}}$ for the smallest possible q

or $\sum_{r=1}^{q} D_{k_r} \geq \sum_{r=0}^{q-1} B_{i_{n-r}}$ for the smallest possible q.

When $q = 1$, this case reduces to the case 1 of Table 2.1.1.

Theorem 2.1.7 : Under the case IV, for any arbitrary permutation π, if a n-critical path of π makes a right turn at cell $(i,1)$, $1 \leq i \leq n-1$, then it will make another right turn at cell $(j,2)$ for $i \leq j \leq \min(i + q - 1, n)$.

Proof : The proof is simple using the given conditions of case IV in Figure 2.1.1. //

Now we discuss in greater detail a subcase of case IV when $q=2$ and $B_i = B$ for every $i \in N$.

Subcase of IV : $q = 2$; $B_i = B$ for every $i \in N$.

At first look, since there exists exactly one job i such that $A_i < B$, one notices that deleting job i the problem satisfies conditions of case (1) in Table 2.1.1. Hence one might feel that a Johnson's permutation of the two machine flow-shop problem (M_1+M_2, M_2+M_3) provides a 'good' near optimal solution for the $(n/3/F/F_{max})$ problem under this subcase (see Johnson (1954)).

The following Example 2.1.4. shows that a problem under subcase of case IV need not imply any of the cases in Table 2.1.1 and cases I, II, and III.

Example 2.1.4 :

Job	Processing times on		
	M_1	M_2	M_3
1	3000	5000	2000
2	7000	5000	3000
3	7000	5000	5000
4	8000	5000	7000
5	9000	5000	6000

Note that for the Example 2.1.4, the Johnson's permutation of (M_1+M_2, M_2+M_3) is $\pi = (4,5,3,2,1)$ with $F_{max}(\pi) = 43,000$ where-as the optimal solution is $\alpha = (4,5,3,1,2)$ with $F_{max}(\alpha) = 42,000$. Thus it might be useful to discuss an exact method for solving this subcase.

For an arbitrary permutation π, we introduce the following notation in the critical-path network of π. A path from cell $(1,1)$ to cell $(n,3)$ having its right turns in the first column at cell $(i,1)$ and in the second column at cell $(i,2)$ is denoted by P_{i1}, $1 \leq i \leq n$. A path from cell $(1,1)$ to cell $(n,3)$ having its right turns in the first column at cell $(i-1, 1)$ and in the second column at cell $(i,2)$ is denoted by P_{i2}, $2 \leq i \leq n$. The length of a path P is denoted by $L(P)$. For an arbitrary permutation π, invoking Theorem 2.1.7, we get

$$F_{max}(\pi) = \max_{1 \leq i \leq n} L(P_i)$$

where $L(P_i) = \max\left\{L(P_{i1}), L(P_{i2})\right\}$

for $1 \leq i \leq n$ with the convention that $P_{12} = \emptyset$ and hence $L(P_{12}) = 0$.

We will use the following theorem due to Smith (1956) in our subsequent theorems.

Theorem 2.1.8 Smith (1956) : Assume f is a real valued function defined on the set of all permutations of n objects. A sufficient condition that $f(\pi^*) \leq f(\pi)$ for all permutations π is that :

(A) There is a real valued function g of ordered pairs of elements such that if π is any permutation and π' a permutation obtained from π by the interchange of the i^{th} and $(i+1)^{th}$ elements in π, then

$$f(\pi) \leq f(\pi') \text{ if } g(\pi(i), \pi(i+1)) \leq g(\pi(i+1), \pi(i))$$

and

(B) π^* is such that the i^{th} object precedes the j^{th} object in π^* if $g(i,j) \leq g(j,i)$.

Theorem 2.1.9 : Under the given conditions of the subcase of case IV, $g(i,j) = \min(A_i+B, B+D_j, A_i+D_j)$ satisfies the condition (A) of Smith's Theorem 2.1.8., where $f(\pi) = F_{max}(\pi)$.

Proof : Let $\pi = (\pi(1), \ldots, \pi(n))$ be an arbitrary permutation and π' be obtained from π by interchanging $\pi(r)$ and $\pi(r+1)$ for some r

such that $1 \leq r \leq n-1$. Further assume that $g(\pi(r), \pi(r+1)) \leq g(\pi(r+1), \pi(r))$. Then we shall prove that $F_{max}(\pi) \leq F_{max}(\pi')$. Denote P_{i1}, P_{i2} and P_i corresponding to π' by P'_{i1}, P'_{i2} and P'_i respectively. From the critical path networks of π and π', using the fact that $B_i = B$, $\forall i$, it is easy to show that

$$L(P_i) = L(P'_i) \quad \text{for} \quad 1 \leq i \leq n, \quad i \neq r, \quad r+1.$$

Further observe that $L(P_{r2}) = L(P'_{r2})$. Thus to prove $F_{max}(\pi) \leq F_{max}$ we should prove that

$$\max\left\{L(P_{r1}),\ L(P_{r+1,1}),\ L(P_{r+1,2})\right\} \leq \max\left\{L(P'_{r1}),\ L(P'_{r+1,1}),\ L(P'_{r+1,2})\right.$$

From the critical path networks, we can write that

$$\max\left\{L(P_{r1}),\ L(P_{r+1,1}),\ L(P_{r+1,2})\right\}$$

$$= \sum_{i=1}^{r+1} A_{\pi(i)} + 2B + \sum_{i=r}^{n} D_{\pi(i)} + \max\left\{-A_{\pi(r+1)} - B,\ -B - D_{\pi(r)},\right.$$

$$\left. - A_{\pi(r+1)} - D_{\pi(r)}\right\}$$

and

$$\max\left\{L(P'_{r1}),\ L(P'_{r+1,1}),\ L(P'_{r+1,2})\right\}$$

$$= \sum_{i=1}^{r+1} A_{\pi(i)} + 2B + \sum_{i=r}^{n} D_{\pi(i)} + \max\left\{-A_{\pi(r)} - B,\ -D_{\pi(r+1)} - B,\right.$$

$$\left. - A_{\pi(r)} - D_{\pi(r+1)}\right\}.$$

Now using the hypothesis $g(\pi(r), \pi(r+1)) \leq g(\pi(r+1), \pi(r))$ the theorem follows. //

In the following theorem we use the notation '$\longleftarrow$' where $i \longleftarrow j$ if and only if $g(i,j) \leq g(j,i)$.

Theorem 2.1.10 : Under the subcase of case IV, let $N_1 \subseteq N$, $A_r = \min_{i \in N_1} A_i$ and $D_s = \min_{i \in N_1} D_i$.

If $A_r \leq D_s$, define

$M = \{ i \in N_1 \mid i \neq s \text{ and } r, \; A_i + D_r \leq \min (B, D_i) + A_r \}$.

Then, (a) $i \in M \Longrightarrow i \longleftarrow r$.

(b) $i \notin M, \; i \in N_1 \Longrightarrow r \longleftarrow i$.

(c) $i \in M, \; j \notin M, \; j \in N_1 \Longrightarrow i \longleftarrow j$.

If $A_r > D_s$, define

$M_1 = \{ i \in N_1 \mid i \neq s \text{ and } r, \; A_s + D_i < \min (B, A_i) + D_s \}$.

Then, (d) $i \in M_1 \Longrightarrow s \longleftarrow i$.

(e) $i \notin M_1, \; i \in N_1 \Longrightarrow i \longleftarrow s$.

(f) $i \in M_1, \; j \notin M_1, \; j \in N_1 \Longrightarrow j \longleftarrow i$.

Proof : Let $A_r \leq D_s$. From the properties of A_r, D_s and M the proofs of (a) and (b) are trivial. To prove (c), for any pair of jobs i and j such that $i \in M$, $j \notin M$, $j \in N_1$, we should show that $g(i,j) \leq g(j,i)$.

Since $i \in M$, we have

$$A_i + D_r \leq \min (B, D_i) + A_r \qquad \ldots \qquad (2.1.1)$$

Thus $A_i + D_r \leq A_r + B \leq B + D_s \leq B + D_r$.

Hence $A_i \leq B$ and therefore we get

$$g(i,j) = \min (B, D_j) + A_i \qquad \ldots \quad (2.1.2)$$

Since $j \notin M$ and $j \in N_1$ we have

$$A_j + D_r > \min (B, D_j) + A_r ,$$

that is

$$\min (B, D_j) - D_r < A_j - A_r . \qquad \ldots \quad (2.1.3)$$

Adding (2.1.1) and (2.1.3) we get

$$\min (B, D_j) + A_i < \min (B, D_i) + A_j . \qquad \ldots \quad (2.1.4)$$

Adding the known relation $B - D_r \leq B - A_r$ to (2.1.1) we get

$$A_i + B \leq \min (B, D_i) + B \leq D_i + B \qquad \ldots \quad (2.1.5)$$

Combination of (2.1.4) and (2.1.5) using (2.1.2) gives $g(i,j) \leq g(j,i)$ and this proves (c).

The proofs of (d), (e) and (f) are in similar lines. This completes the proof of Theorem 2.1.10. //

Using the results of Theorems 2.1.9 and 2.1.10, we provide a Procedure 'subcase of IV' which yields an optimal π^* to the $(n/3/F/F_{max})$ problem under subcase of IV. In the procedure we use

the following notation :

- N_1 stands for the set of jobs not yet fixed in the positions of π^*.
- $r : A_i \in N_1$ stands for choosing a r such that $A_r := \min_{i \in N_1} A_i$. Similarly $s : D_i \in N_1$ is interpreted.
- I_k stands for the last fixed position of the initial fixed portion of π^*.
- L_k stands for the first fixed position of the last fixed portion of π^*.
- The sets M and M_1 are as defined in Theorem 2.1.9.

Procedure 'subcase of IV' (A, B, D, π^*, N)

1. Begin Local N_1; M; M_1; I_k; L_k;
2. $N_1 := N$; $I_k := 0$; $L_k := |N| + 1$;
3. while $N_1 \neq \emptyset$ do
4. $r : A_i \in N_1$; $s : D_i \in N_1$;
5. if $A_r \leq D_s$ then $M = \{ i \in N_1 \mid i \neq s$ and r; $A_i + D_r \leq \min(B, D_i) + A_r \}$;
6. if $M \neq \emptyset$ then call Procedure 'subcase of IV' (A, B, D, β, M)
7. do $i = I_k + 1$, $I_k + |M|$
8. $\pi^*(i) := \beta(i - I_k)$;
9. end

10. $I_k := I_k + |M|$; $N_1 := N_1 - M$;

11. $\pi^*(I_k+1) := r$; $I_k := I_k+1$; $N_1 := N_1 - \{r\}$;

12. else (i.e. $M = \emptyset$) $\pi^*(I_k+1) := r$; $I_k := I_k+1$; $N_1 := N_1 - \{r\}$;

13. else (i.e. $A_r > D_s$) then $M_1 = \{i \in N_1 \mid i \neq s$ and r; $A_s + D_i < \min(B, A_i) + D_s\}$

14. if $M_1 \neq \emptyset$ then call Procedure 'subcase of IV' (A, B, D, α, M_1)

15. do $i = L_k - |M_1|$, $L_k - 1$

16. $\pi^*(i) := \alpha(i - L_k + |M_1| + 1)$;

17. end

18. $L_k := L_k - |M_1|$; $N_1 := N_1 - M_1$;

19. $\pi^*(L_k - 1) := s$; $L_k := L_k - 1$; $N_1 := N_1 - \{s\}$;

20. else (i.e. $M_1 = \emptyset$) $\pi^*(L_k - 1) := s$; $L_k := L_k - 1$; $N_1 := N_1 - \{s\}$;

21. end

22. end Procedure 'subcase of IV' (A, B, D, π^*, N).

Theorem 2.1.11 : π^* generated by the Procedure 'subcase of IV' (A, B, D, π^*, N) is an optimal solution of the $(n/3/F/F_{max})$ problem under subcase of case IV.

Proof: The Theorem follows trivially if we show that π^* satisfies the condition (B) of Smith's Theorem 2.1.8. We accomplish this by induction on $|N|$. For the basis of induction we prove the theorem when $|N| = 2$. Now Procedure 'subcase of IV' terminates after executing either Case(i)

lines 5-12 or Case(ii) lines 13-19. Under Case(i) note that if $M \neq \emptyset$ ($M = \emptyset$) then in lines 6-11, $\pi^*(1)$ and $\pi^*(2)$ are (in line 12, $\pi^*(1)$ is) fixed such that

$$g(\pi^*(1), \pi^*(2)) \leq g(\pi^*(2), \pi^*(1)).$$

This can be checked under Case(ii) as well. Thus the condition(B) of Smith's Theorem is satisfied for π^* when $|N| = 2$. Now make the induction hypothesis that the condition (B) holds for π^* for all $|N| < n$. We prove it for $|N| = n$.

Now either lines 5-12 or lines 13-19 will be executed. Suppose lines 5-12 are executed when $N_1 = N$. Then note by the induction hypothesis that the jobs in M are fixed in correct order among themselves. Now by the results (a) and (b) of Theorem 2.1.10, we observe that job r is fixed correctly relative to all other jobs. Further from result(c) of Theorem 2.1.10, we get that the jobs in M are fixed correctly relative to all other jobs not in M. Similar arguments can be repeated when lines 13-19 are executed. Subsequently the number of jobs in N_1, yet to be fixed is strictly less than n and by induction hypothesis the result follows. //

It is easy to check that the computational complexity of the Procedure 'subcase of IV' is $O(n)$.

2.2 Heuristic Rules :

We define certain general terms before discussing the heuristics for solving the $(n/3/F/F_{max})$ problem.

Definition 2.2.1 : Given a permutation $\pi = (\pi(1), \ldots, \pi(n))$ we say that π' is a Contiguous Binary Switching (C B S) neighbour of π, if π' can be obtained from π by interchanging $\pi(i)$ and $\pi(i+1)$ for some i, $1 \leq i \leq n-1$.

Definition 2.2.2 : Given a permutation $\pi = (\pi(1), \ldots, \pi(n))$, we call π' a Forward Push Switching - FPS (Backward Push Switching - BPS) neighbour of π in case π' can be obtained from π by placing $\pi(i)$ for some i, $2 \leq i \leq n$ ($1 \leq i \leq n-1$) in the first position (last position) and pushing $\pi(k)$, $1 \leq k \leq i-1$ one position forward ($\pi(k)$, $i+1 \leq k \leq n$ one position backward) where-as the remaining jobs occupy same positions under π and π' .

Definition 2.2.3 : Given a permutation π, generate all its (n-1) CBS neighbours and choose the permutation with least F_{max} out of the n permutations in hand. This procedure is called CBS improvement procedure. Similarly FPS and BPS improvement procedures are defined.

Table 2.2.1 presents the commonly used heuristics.

Table 2.2.1 : Common heuristics for $(n/3/F/F_{max})$ problem

Heuristic	Description	Reference	Remarks
H1	A Johnson's permutation of the two machine flow-shop problem (M_1+M_2, M_2+M_3).	Giglio et al. (1964)	Hutchinson et al.(1977) also investigated this rule. Computational complexity of this heuristic is O(n log n).
H2	Decreasing sequence of $D_i - A_i$	Palmer (1965)	Its computational complexity is O(n log n).
H3	H3.1: Increasing sequence of A_i. H3.2: Decreasing sequence of D_i. H3.3: Increasing sequence of $(D_i-A_i)/(A_i+B_i+D_i)$.	McMahon et al. (1967)	Each of them is of computational complexity O(n log n).
H4	Find Johnson's permutations of the two machine flow-shop problems (M_1,M_3) and (M_1+M_2, M_2+M_3). Choose the one with least F_{max} out of the two permutations.	Campbell et al. (1970)	Its computational complexity is O(2n log n+6n).
H5	Linear branch-and-bound procedure. Here branching is always done from one of the newly generated partial permutations with least lower bound of Lomnicki (1965).	Ashour (1970)	Dannenbring(1977) modified this heuristic by using the lower bound due to McMahon et al.(1967). This heuristic is of polynomial computational complexity in n.
H6	Johnson's permutation of the two machine flow-shop problem $(3M_1+2M_2+M_3, M_1+2M_2+3M_3)$.	Dannenbring (1977)	This heuristic is of computational complexity O(n log n). (See Note 2.2.4(ii)

Note 2.2.4: (i) The heuristics H1 to H6 are of polynomial time complexity. The literature includes heuristics which are not of polynomial time complexity such as :

- Baker (1974) : Construct permutations by random sampling and choose the best out of them.
- Page (1961) : Construct permutations by any one of the techniques: Merging, Pairing, Individual exchanging and Group exchanging. Choose the best out of them.

(ii) Dannenbring (1977) applied the CBS-improvement procedure to various heuristics and he termed it close order scarch. He also applied an improvement procedure termed extensive search : If the best permutation π' given by CBS-improvement procedure applied to π is such that $\pi' \neq \pi$ then once again apply CBS-improvement procedure to π'.

(iii) All the above mentioned heuristics except H1 and H3 have been generalised and applied to larger number of machines by the respective authors.

Dannenbring (1977) has compared most of the above mentioned heuristics. He observes that the extensive search applied to H6 provides the best heuristic in the sense that maximum number of problems are solved optimally by this heuristic. However, it should be noted that this heuristic is not of polynomial computational complexity and in fact it could lead to a complete enumeration as illustrated by the following $(3/3/F/F_{max})$ example.

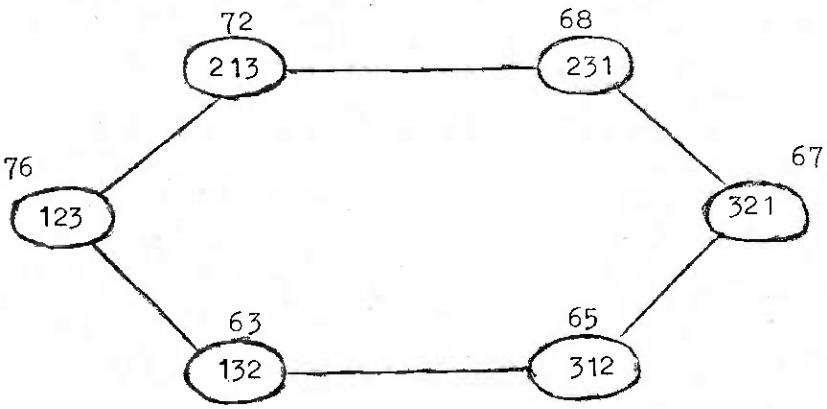
Example 2.2.1 :

Job	Processing times on		
	M_1	M_2	M_3
1	5	13	9
2	1	15	2
3	11	19	24

For the corresponding two machine flow-shop problem ($3M_1+2M_2+M_3$, $M_1+2M_2+3M_3$), presented below, $\pi = (2,1,3)$ is a Johnson's permutation.

Job	$3M_1+2M_2+M_3$	$M_1+2M_2+3M_3$
1	50	58
2	35	37
3	95	121

In the following graph of the $(3/3/F/F_{max})$ problem, a node represents a permutation and two nodes are adjacent if the respective permutations are CBS neighbours. The number given above the node represents the F_{max} corresponding to that node.



From the graph it is easy to see that the extensive search applied to H6 starts with permutation (2,1,3) and enumerates permutations (1,2,3); (2,3,1); (3,2,1); (3,1,2) and (1,3,2) in that order. Further note that (1,3,2) is the unique optimal permutation which is obtained after complete enumeration.

Important desired aspects of a heuristic rule are its nearness to the optimal solution and easiness in the involved computations. Keeping this point in view, our investigation has been restricted to heuristics of popynomial computational time complexity.

2.2.1 Heuristics under investigation :

Lageweg et al.(1978) observed that the lower bound $LB(\sigma ; \hat{W}, * \square *)$ associated with a partial permutation σ where $\hat{W} = \{(u,u) \mid 1 \leq u \leq m\}$ (see section 1.4 of Chapter I) is slightly stronger than the machine based bound due to Lomnicki (1965). Further, the lower bound $LB(\sigma; W, * \square 0 \square *)$ with $W = \{(u,m) \mid 1 \leq u \leq m-1\}$ is stronger than the job-based bound due to McMahon et al.(1967). Through experimental investigation they established that computationally efficient results were obtained by the search algorithms incorporating the SDC as elimination criteria and $LB(\sigma; W, * \square 0 \square *)$ as a lower bound. Hence we modify the heuristic H5 as a linear branch-and-bound procedure using $LB(\sigma ; W, * \square 0 \square *)$ and SDC. We have already seen under case I of section 2.1 that use of SDC and $LB(\sigma; W, * \square 0 \square *)$ involves

computational complexity of $O(3(n-s)(n-s-1))$ and $O((2n+5)(n-s))$ respectively when $|\sigma| = s$. Note that linear branch-and-bound procedure generates at most $(n-s+1)$ nodes with corresponding partial permutation σ satisfying $|\sigma| = s$. Hence the modified H5 will involve $O(2n \log n)$ steps at the root node to find the Johnson's permutations. The computational complexity for the remaining nodes put together can be at most $O\left((2n+5)n + \frac{n(n-1)(n+1)}{3}(2n+8)\right)$ i.e. $O(n^4)$.

Apart from the remaining heuristics of Table 2.2.1, we include in our investigation the following basic heuristics:

H7 : Johnson's permutation of the problem (M_1, M_2).

H8 : Johnson's permutation of the problem (M_2, M_3).

H9 : Johnson's permutation of the problem (M_1, M_3).

H10: Johnson's permutation of the problem $(M_1+M_2,\ \min(M_1, M_2)+M_3)$.

H11: Johnson's permutation of the problem $(M_1+\min(M_2, M_3),\ M_2+M_3)$.

H12: A procedure for solving a special case due to Burns et al. (1975) which we describe below.

In the following procedure, for arbitrary $N_1 \subseteq N$ we use $N_1 \bar{X} N_1$ to mean the set $\{(i,j) \mid i \neq j \text{ and } i,j \in N_1\}$. We use Procedure 'JP (A,B,N,π)' as a subroutine to obtain Johnson's permutation π of a two machine flow-shop problem.

Procedure 'H12 (A,B,D,N, π^*)'

1. Begin Local N_1; M_1; M; I_k; L_k; α; β;
2. call Procedure 'JP (A+B, min(A,B) + D, N, α)';
3. call Procedure 'JP (A + min(B,D), B+D, N, β)';
4. $N_1 := N$; $I_k := 0$; $L_k := |N| + 1$;
5. while $N_1 \neq \emptyset$ do
6. $q : (A_i+B_i) \in N_1$; $r : (B_i+D_i) \in N_1$; $(s,t) : (A_i+D_j) \in N_1 \bar{X} N_1$;
7. $Qmin := \min\{A_q+B_q,\ B_r+D_r,\ A_s+D_t\}$;
8. if $A_q+B_q = Qmin$ then $\pi^*(I_k+1) := q$; $I_k := I_k+1$; $N_1 := N_1-\{q\}$;
9. else if $B_r+D_r = Qmin$ then $\pi^*(L_k-1) := r$; $L_k := L_k-1$; $N_1 := N_1-\{r\}$
10. else if $A_s \leq D_t$ then $M := \{j \mid A_s=A_j,\ j \in N_1,\ j \neq t\}$; $\sigma := \alpha_M$;
11. do i = I_k+1, $I_k + |\sigma|$
12. $\pi^*(i) := \sigma(i - I_k)$;
13. end
14. $I_k := I_k + |\sigma|$; $N_1 := N_1 - M$;
15. else (i.e. $A_s > D_t$) then $M_1 := \{j \mid D_t=D_j,\ j \in N_1,\ j \neq s\}$; $\sigma := \beta_M$
16. do i = $L_k - |\sigma|$, $L_k - 1$
17. $\pi^*(i) := \sigma(i - L_k + |\sigma| + 1)$;
18. end
19. $L_k := L_k - |\sigma|$; $N_1 := N_1 - M_1$;
20. end
21. end Procedure 'H12 (A,B,D,N, π^*)'.

Procedure 'H12(A,B,D,N, π^*)' is of computational complexity $O(n+2n \log n)$. Each of the heuristics H7 to H11 is of computational complexity $O(n \log n)$.

To each of the heuristics H1 to H12, the FPS and BPS improvement procedures were applied. We denote the application of FPS and BPS improvement procedures to heuristic Hj by new heuristics Hj(a) and Hj(b) respectively. Dannenbring (1977) observed that application of CBS improvement procedure (close order search) to H6 also provides satisfactory results. Hence we include in our investigation application of CBS procedure to H6 alone and denote the resulting heuristic by H6(c). Each of the FPS, BPS and CBS improvement procedures when applied to a permutation π, involves computational complexity of $O(3n^2)$.

It should be observed that the following heuristics solve optimally the corresponding special cases of Table 2.1.1:

H1 : case(1), case(5) and case (7) ;

H7(a) : case (3);

H7(b) : case (4);

and H12 : case (6).

2.2.2 The Experiment :

The experiment included 300 problems where the job size n was varied from 5 to 16. For each job size n, 25 problems were generated.

The processing times for the problems were drawn from integer valued discrete uniform distribution in the interval $[1,100]$. It is felt that the uniform distribution provides the more difficult problems to solve (see Dannenbring 1977).

For the computations involved FORTRAN programmes were developed and the computer Honeywell 400 was used.

The heuristic solutions to all the generated problems were obtained. All the problems were solved for exact solutions, using the branch-and-bound method. The best known solutions from the heuristics was used to discard some of the partial permutations during the search procedure of branch-and-bound method. Further the search procedure included SDC as elimination critaria. At any stage of solving a problem, if the number of active partial permutations exceeded 50, the search method terminated without getting the exact solution of the problem. Out of the 300 problems, included in the experiment, 13 were not solved for exact solution and in these cases the best lower bound of the active partial permutations at the termination stage, was taken to be the optimal value of the completion time.

For a given problem, percentage error of a heuristic H is defined by $E_H = 100\ (F_H - F^*)/F^*$ where F_H is the F_{max} evaluated at the permutation given by heuristic H and F^* is the F_{max} evaluated at an exact optimal solution. The percentage error E_H is used as the index for the comparison of various heuristics.

By Chi-square test, it was observed that the performance of a heuristic does not significantly vary with the number of jobs n (at level 10%). Hence the performance of heuristics was studied using all the 300 problems independent of job size n. Table 2.2.2 provides a comparison of various heuristics on the basis of E_H.

Table 2.2.2 : Comparison of various heuristics

Heuristic		Percentage occurrence of			Max E_H (%)	Mean ($\bar{E}_H$)	Standard deviation (σ_{E_H})
Basic	improved	$E_H = 0\%$	$0 \leq E_H \leq 1\%$	$0 \leq E_H \leq 5\%$			
H1		42.9	48.7	71.0	23.34	3.43	4.63
	H1(a)	50.0	57.0	84.7	20.93	2.18	3.34
	H1(b)	50.3	60.0	82.3	15.20	2.06	3.13
H2		15.3	21.0	55.3	24.58	4.94	4.30
	H2(a)	29.7	42.0	79.0	18.97	2.68	3.05
	H2(b)	25.7	34.0	78.0	19.83	3.25	3.56
H3.1		8.0	12.0	28.3	26.54	9.64	6.58
	H3.1(a)	13.0	18.3	40.3	26.54	7.83	6.28
	H3.1(b)	22.3	33.7	63.7	17.58	4.20	4.18
H3.2		5.7	9.7	26.8	32.93	9.86	6.77
	H3.2(a)	21.3	30.0	65.7	22.16	4.10	4.26
	H3.2(b)	12.3	17.3	34.7	31.3	7.89	6.19
H3.3		0.0	0.0	0.0	73.88	38.32	11.55
	H3.3(a)	0.0	0.0	0.0	62.55	27.06	9.86
	H3.3(b)	0.0	0.0	0.0	56.79	26.39	9.12

Table 2.2.2 (continued)

Heuristic		Percentage occurrence of			Mas E_H (%)	Mean ($\bar{E}_H$)	Standard deviation (σ_{E_H})
Basic	improved	$E_H = 0\%$	$0 \leq E_H \leq 1\%$	$0 \leq E_H \leq 5\%$			
H4		52.7	58.7	80.3	23.34	2.42	3.92
	H4(a)	57.0	63.3	87.3	20.13	1.82	3.20
	H4(b)	59.0	66.0	88.0	16.62	1.68	2.92
H5		65.3	78.0	96.0	14.49	0.76	1.82
	H5(a)	68.3	83.0	97.3	11.67	0.64	1.61
	H5(b)	68.3	82.0	98.3	8.57	0.57	1.30
H6		26.3	34.3	73.0	23.34	3.55	4.17
	H6(a)	38.7	48.4	86.3	18.20	2.24	3.16
	H6(b)	41.3	52.0	84.7	15.20	2.01	2.98
	H6(c)	53.0	61.7	87.7	17.35	1.73	2.98
H7		2.3	3.3	16.7	43.20	14.12	9.54
	H7(a)	7.7	11.0	31.0	35.14	10.26	7.94
	H7(b)	14.3	18.0	41.0	32.43	7.60	6.71
H8		5.7	5.9	17.7	46.04	14.98	9.84
	H8(a)	11.3	14.3	38.7	32.98	8.18	6.93
	H8(b)	8.3	10.7	31.3	36.62	10.25	7.88
H9		17.0	20.7	48.3	29.97	4.16	5.56
	H9(a)	24.7	31.7	70.0	18.20	3.72	3.98
	H9(b)	30.7	36.4	69.7	21.71	3.58	4.00
H10		25.7	31.3	47.7	34.57	7.65	7.87
	H10(a)	39.0	47.7	76.0	19.23	3.28	4.33
	H10(b)	30.7	37.3	57.7	27.1	5.60	6.21
H11		27.7	32.0	45.7	33.92	7.39	7.82
	H11(a)	32.3	36.7	56.0	28.32	5.41	6.13
	H11(b)	35.7	44.7	76.0	19.39	3.10	4.04
H12		35.7	44.3	71.3	29.97	3.35	4.32
	H12(a)	42.0	54.3	83.0	20.13	2.12	3.26
	H12(b)	46.7	57.0	87.0	21.71	1.87	2.93

The following remarks are made from Table 2.2.2. To simplify notation we use σ_H to mean σ_{E_H}. Let α_H denote the conditional mean of the distribution of E_H given $E_H > 0$. α_H will provide an idea of the performance of heuristic H when the heuristic does not give an optimal solution for a problem.

Remark 2.2.5 : Comparison between the 14 basic heuristics :

(a) H5 is the best heuristic under any basis of comparison possible from Table 2.2.2. It is interesting to observe that H5 solves 65.3% of the problems optimally and 96% of the problems with $E_{H5} \leq 5\%$. It has $\bar{E}_{H5} = 0.76$ and $\sigma_{H5} = 1.82$. However the fact that H5 involves significantly high computational effort compared to other heuristics, urges us to look for satisfactory heuristics other than H5. This leads to the following observations.

(b) H4 is the next best heuristic with $\bar{E}_{H4} = 2.42$ and $\sigma_{H4} = 3.92$.

(c) H1, H6 and H12 are moderately comparable to H4 with their $\bar{E}_H$ around 3.5 and σ_H around 4.5.

(d) Comparison of the relative cumulative distribution of E_H (not presented in the text) for heuristics H1, H4, H6 and H12 suggests that :

- in the range $0 \leq E_H \leq 4\%$, H4, H1, H12 and H6 are preferred in that order.

- in the range $4\% < E_H \leq 10\%$, H4, H6, H12 and H1 are preferred in that order.

(e) It is of interest to note that

$$\alpha_{H5} = 2.21 < \alpha_{H6} = 4.82 < \alpha_{H4} = 5.11 < \alpha_{H12} = 5.21 < \alpha_{H1} = 6.02$$

(f) On the basis of computational complexity heuristics H1, H6, H12, H4 and H5 will be preferred in that order.

Remark 2.2.6 : Comparison between all the heuristics :

(a) H5(b) is the best heuristic and H5(a) is the next best. Considering the point that these two heuristics involve significant computational effort, we look for heuristics with lesser computational effort and observe the following.

(b) H4(b) is the next best heuristic with $\bar{E}_H = 1.68$ and $\sigma_H = 2.92$.

(c) H6(c), H4(a) and H12(b) are moderately comparable to H4(b) since they have $1.73 \leq \bar{E}_H \leq 1.87$ and $2.93 \leq \sigma_H \leq 3.20$.

(d) We note that,

$$\alpha_{H5(b)} = 1.80 < \alpha_{H5(a)} = 2.14 < \alpha_{H12(b)} = 3.50 < \alpha_{H6(c)} = 3.68 < \alpha_{H4(b)} = 4.10 < \alpha_{H4(a)} = 4.23.$$

(e) On the basis of computational complexity H6(c), H12(b), H4(b), H4(a), H5(b) and H5(a) will be preferred in that order.

Remark 2.2.7 : Impact of improvement procedures :

(a) The benefit realised by the CBS-improvement procedure applied to H6 is significant.

(b) The improvement procedures interact with the basic heuristics. For instance FPS- improvement procedure betters H3.2 significantly and BPS- improvement procedure betters H3.1 significantly whereas the otherway is not true.

(c) The FPS and BPS improvement procedures do not better H5 appreciably.

(d) By applying selective improvement procedures to H4, H6 and H12 we can solve about 88% of the problems with $E_H \leq 5\%$.

3 Combination of heuristics :

By a combination of heuristics in set S we mean the selection of a permutation (where minimum F_{max} is attained) among all the permutations generated by the heuristics in S. Thus a combination through set S defines a new heuristic denoted by CH.

Certain combinations of the basic and/or the improved heuristics are considered with a view to obtaining a computationally easier heuristic than H5 and simultaneously performing nearly as well as H5. Their performances are summarised in Table 2.2.3 which is self explanatory.

Table 2.2.3 : Combination of heuristics

Heuristic	Description of combination (set S)	Percentage occurrence of $E_H=0\%$	$0\leq E_H\leq 1\%$	$0\leq E_H\leq 5\%$	Max E_H (%)	Mean ($\bar{E}_H$)	σ_H	α_H	Computational effort in 'O' notation
CH1	H1(a), H1(b)	58.3	66.7	91.0	13.28	1.35	2.31	3.23	$n \log n+3(2n-1)n$
CH2	H4(a), H4(b)	64.0	70.3	91.7	13.28	1.20	2.24	3.33	$2n \log n + 6n^2$
CH3	H6(a), H6(b)	49.0	60.7	91.7	13.28	1.45	2.26	2.83	$n \log n+3(2n-1)n$
CH4	H12(a),H12(b)	50.7	63.7	92.0	14.73	1.36	2.28	2.76	$2n \log n+2(3n-1)n$
CH5	H1(a), H1(b), H12(a),H12(b)	68.0	79.0	97.0	13.28	0.72	1.66	2.25	$3n \log n+(12n-5)n$
CH6	H1,H4,H6,H12	60.7	68.7	91.3	23.34	1.43	2.94	3.65	$13n + 5n \log n$
CH7	CH6(a),CH6(b)	67.7	75.3	96.0	13.28	0.84	1.82	2.62	$5n \log n +6n^2+7n$
CH8	H4(b), H4(a), H6(c),H12(b)	72.3	81.0	98.0	13.28	0.57	1.47	2.07	$5n \log n+12n^2+n$

Remark 2.2.8 : From Table 2.2.3 we note the following.

(a) Both FPS and BPS improvement procedures put together have significantly bettered the performances of some of the basic heuristics. For instance, CH1 has optimal solutions in around 15% more of the cases compared to H1 (from Table 2.2.2) and infact reduces $\bar{E}_H$, σ_H and α_H significantly. This phenomenon is common to CH2, CH3 and CH4 as well. However this is at an additional computational effort of about $O(6n^2)$.

(b) CH8 is the best heuristic solving 72.3% of the problems optimally and 98% of the problems with $E_H \leq 5\%$. It has $\bar{E}_H = 0.57$ and $\sigma_H = 1.47$. This heuristic needs much less computational effort than H5(b) and performs almost as well as H5(b).

(c) CH5 which is of much less computational complexity than H5 performs better than H5 and infact compares very well with H5(a). This heuristic CH5 is of less computational effort than CH8 as well.

2.2.4 Restrictions on processing times :

Hutchinson et al.(1977) studied the performance of H1 for problems with restrictions on processing times. The restrictions were of two types viz. (1) $B_i \leq \max(A_i, D_i)$ for every job i and (2) $B_i \leq \min(A_i, D_i)$ for every job i. Later Szwarc (1977a) proved that H1 finds optimal solution under the restriction (2). Here we study the performances of H1, H12, H5 and combinations of H1 and H12 under weaker conditions on processing times. The 300 problems were classified into one of the three mutually exclusive groups defined in Table 2.2.4.

Table 2.2.4 : Restrictions on processing times

Groups	Restriction	Number of problems
1	$\sum_i B_i \leq \min\left\{\sum_i A_i, \sum_i D_i\right\}$	107
2	$\min\left\{\sum_i A_i, \sum_i D_i\right\} < \sum_i B_i < \max\left\{\sum_i A_i, \sum_i D_i\right\}$	111
3	$\max\left\{\sum_i A_i, \sum_i D_i\right\} \leq \sum_i B_i$	82
	Total	300

Table 2.2.5 provides the comparison of certain heuristics under the groups defined in Table 2.2.4.

Table 2.2.5 : Comparison of heuristics on problems with restrictions

Group	Heuristic	Percentage occurrence of E_H=0%	Percentage occurrence of $0 \leq E_H \leq 1\%$	Percentage occurrence of $0 \leq E_H \leq 5\%$	Max E_H (%)	Mean ($\bar{E}_H$)	σ_H	α_H	Computational complexity O of
1	H1	67.3	78.5	88.8	13.19	1.18	2.49	3.61	n log n
	H12	46.7	57.9	75.7	15.92	2.44	3.61	4.57	n + 2n log n
	CH1	74.8	84.1	93.4	7.47	0.78	1.84	3.08	n log n + 3(2n-1)n
	CH4	57.0	69.2	88.8	10.74	1.40	2.33	3.27	2n log n + 2(3n-1)n
	CH5	78.5	87.9	95.3	7.47	0.59	1.64	2.76	3n log n+ (12n-5) n
	H5	72.9	80.4	96.3	7.46	0.62	1.45	2.28	n^4
2	H1	45.9	51.3	76.6	18.43	2.81	3.87	5.19	n log n
	H12	33.3	37.8	66.7	13.71	3.74	3.80	5.61	n + 2n log n
	CH1	69.4	70.2	94.6	7.58	1.03	1.94	3.36	n log n + 3(2n-1)n
	CH4	55.0	64.9	93.7	7.89	1.13	1.86	2.52	2n log n + 2(3n-1)n
	CH5	73.0	76.6	99.0	7.04	0.57	1.17	2.11	3n log n + (12n-5)n
	H5	60.36	75.68	94.59	14.49	0.96	2.30	2.42	n^4
3	H1	2.4	6.1	39.0	23.34	7.22	5.41	7.42	n log n
	H12	23.2	35.4	68.3	29.97	4.02	5.49	5.24	n + 2n log n
	CH1	22.0	39.0	81.7	13.28	2.52	2.96	3.23	n log n + 3(2n-1)n
	CH4	36.6	54.9	93.9	14.73	1.62	2.66	2.56	2n log n + 2(3n-1)n
	CH5	47.5	70.7	96.3	13.28	1.08	2.16	2.07	3n log n + (12n-5)n
	H5	62.2	78.6	97.6	9.64	0.67	1.47	1.75	n^4

Remark 2.2.9 : From Table 2.2.5 we observe the following :

(a) CH5 is the best heuristic for problems under groups 1 and 2.

(b) H5 is the best heuristic for problems under group 3.

(c) CH1 which is of lesser computational effort than CH5 performs satisfactorily for problems under groups 1 and 2.

2.2.5 Conclusions of the experimental investigation :

Without restrictions on the processing times, CH8 performs the best and is of moderate computational effort. Taking note of the restrictions on the processing times we recommend the following. Use CH5 for problems under groups 1 and 2. While applying CH5, the property of case 5 in Table 2.1.1 can be exploited. An attempt, for problems under group 3, to combine some heuristics with a view to keeping the computational effort around that of CH5 and still get satisfactory performance, turned out to be futile. Hence we recommend H5 (with more computational effort) for problems under group 3.

CHAPTER III

$(n/m/F/F_{max})$ PROBLEM

3.0 Introduction :

The first branch-and-bound method to solve the $(n/m/F/F_{max})$ problem was provided by Brown et al.(1966). This problem for $m \geq 3$ is proved to be NP complete by Garey et al.(1976) and Lenstra et al. (1977). During the period 1966 till date, probably the $(n/m/F/F_{max})$ problem has enjoyed maximum attention among the scheduling problems. A vast literature on this problem is available. But we do not intend to survey this area. The basic work related to this problem can be obtained from Baker (1974), Conway et al.(1967), Coffman (1976), Rinnooy Kan (1976) and Lenstra (1977(a)). Fairly good understanding of the recent research on this problem can be had from Lageweg et al. (1978), Szwarc (1978), Gonzalz et al.(1978) and Gupta (1976).

In section 3.1 we provide sufficient conditions for a particular permutation to be an optimal solution of the $(n/m/F/F_{max})$ problem. A special case of the $(n/m/F/F_{max})$ problem is studied in section 3.2. Subsequently in section 3.3, we investigate the application of SDC to the $(n/m/F/F_{max})$ and its reverse problem.

In this chapter we make the flow-shop assumptions A1 to A9 inclusive of A4' and A5' (given in section 1.2 of Chapter I).

3.1 Sufficient conditions for an optimal solution :

Johnson (1954) makes the following conjecture for the $(n/3/F/F_{max})$ problem : "If $\pi = (\pi(1), \ldots, \pi(n))$ is a Johnson's permutation for the two machine flow-shop problems (M_1, M_2) and (M_2, M_3), then π is an optimal solution of the $(n/3/F/F_{max})$ problem".

Burns et al.(1976) show by a counter example that the above conjecture is not correct. Further they prove the following modified form of the conjecture : "If $\pi = (\pi(1), \ldots, \pi(n))$ is a Johnson's permutation for the two machine flow-shop problems (M_1, M_2), (M_2, M_3) and (M_1, M_3), then π is an optimal solution of the $(n/3/F/F_{max})$ problem."

In this section we prove the following generalised version of the above result for the $(n/m/F/F_{max})$ problem where $m \geq 3$. "If $\pi = (\pi(1), \ldots, \pi(n))$ is a Johnson's permutation for every two machine flow-shop problem (M_u, M_v), $1 \leq u < v \leq m$, then π is an optimal solution of the $(n/m/F/F_{max})$ problem."

Before we proceed to prove this result we show by the following $(2/m/F/F_{max})$ example that these conditions cannot be weakened.

Example 3.1.1 : Let a be a positive real number. Choose any pair of machines M_u and M_v such that $1 \leq u < v \leq m$ where $m \geq 4$. Define the processing times of jobs 1 and 2 on machines M_u and M_v by :

jobs	processing times	
	p_{iu}	p_{iv}
1	a+3	a+1
2	a+4	a+2

Define the processing times of jobs 1 and 2 on machines M_r, $1 \leq r \leq m$, $r \neq u$ and v as : $p_{1r} = p_{2r} = a$. Now observe that $\pi = (1,2)$ is a Johnson's permutation for every two machine flow-shop problem (M_r, M_s), $1 \leq r < s \leq m$, $(r,s) \neq (u,v)$. But π is not an optimal solution of the $(2/m/F/F_{max})$ problem, since for $\pi' = (2,1)$ we have,

$$F_{max}(\pi') = (m+1)a + 8 < (m+1)a + 9 = F_{max}(\pi).$$

Let Figure 3.1.1 be the critical path network of (j1), given the m tuple $\partial = (\partial_1, \ldots, \partial_m)$ where ∂_i is the time machine M_i is available for processing the jobs, $1 \leq i \leq m$.

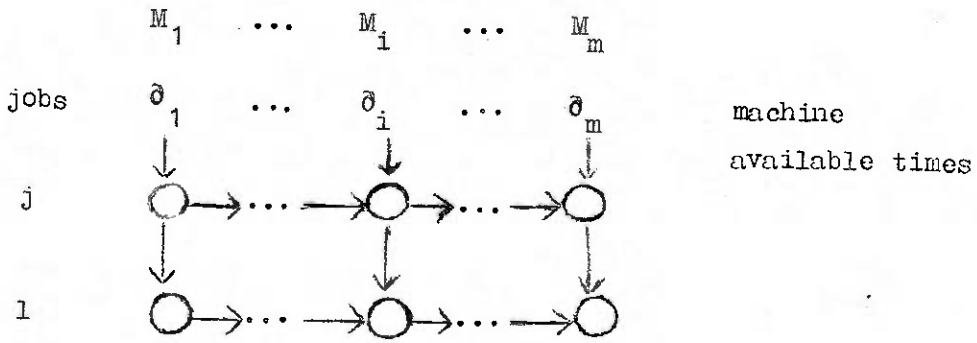


Figure 3.1.1 : Critical path network

From the critical path network it is clear that for $1 \leq k \leq m$,

$$t(\partial; jl; k) = \max_{1 \leq r \leq k} \left\{ \partial_r + \max_{r \leq q \leq k} \left(\sum_{s=r}^{q} p_{js} + \sum_{s=q}^{k} p_{ls} \right) \right\}$$

$$= \max_{1 \leq r \leq k} \left\{ \partial_r + \sum_{s=r}^{k} (p_{js} + p_{ls}) - \min_{r \leq q \leq k} \left(\sum_{s=q+1}^{k} p_{js} + \sum_{s=r}^{q-1} p_{ls} \right) \right\}$$

$$= \max_{1 \leq r \leq k} \left\{ \partial_r + \sum_{s=r}^{k} (p_{js} + p_{ls}) - g_r^k(j,l) \right\} \qquad \ldots(3.1.1)$$

where

$$g_r^k(j,l) = \min_{r \leq q \leq k} \left(\sum_{s=q+1}^{k} p_{js} + \sum_{s=r}^{q-1} p_{ls} \right), \quad 1 \leq r \leq k \leq m \qquad \ldots \qquad (3.1.2)$$

with the convention that $\sum_{s=u}^{v} p_{js} = 0$ for $v < u$.

Lemma 3.1.1 : For a given pair of jobs j and l, and for every pair of machines M_r and M_k such that $1 \leq r < k \leq m$, let

$$\min(p_{jr}, p_{lk}) \leq \min(p_{lr}, p_{jk}) \qquad \ldots \quad (3.1.3)$$

Then

$$g_r^k(j,l) \geq g_r^k(l,j) \quad \text{for every} \quad 1 \leq r \leq k \leq m.$$

Proof : Before proving the lemma, we prove the following :

$$\sum_{s=r}^{k-1} p_{ls} \geq g_r^k(l,j) \quad \text{for} \quad 1 \leq r \leq k \leq m.$$

First note that for M_u and M_k such that $r \leq u < k$, we have

$$p_{lu} \geq \min(p_{lu}, p_{jk}) \geq \min(p_{ju}, p_{lk}).$$

Hence,

$$\sum_{s=r}^{u-1} p_{js} + \sum_{s=u}^{k-1} p_{ls} \geq \min\Big(\sum_{s=r}^{u-1} p_{js} + p_{ju} + \sum_{s=u+1}^{k-1} p_{ls} \;,\; \sum_{s=r}^{u-1} p_{js} + p_{lk} + \sum_{s=u+1}^{k-1} p_{ls} \Big).$$

Here note that the second term in RHS corresponds to the term for $q=u$ in the expression for $g_r^k(1,j)$. Thus starting with $\sum_{s=r}^{k-1} p_{ls}$ i.e. $u=r$ and repeating the above arguments with reference to the first term in RHS for $u = r+1, \ldots, k-1$, we get

$$\sum_{s=r}^{k-1} p_{ls} \geq g_r^k(1,j) \quad \text{for} \quad 1 \leq r \leq k \leq m \qquad \ldots \qquad (3.1.4)$$

Now we prove the lemma by induction. Consider any r and k such that $1 \leq r \leq k \leq m$. Note that

(i) $g_r^r(j,1) = 0 = g_r^r(1,j)$

(ii) $$g_r^{r+1}(j,1) = \min_{r \leq q \leq r+1} \Big(\sum_{s=q+1}^{r+1} p_{js} + \sum_{s=r}^{q-1} p_{ls} \Big)$$

$$= \min(p_{j,r+1}, p_{lr})$$

$$\geq \min(p_{l,r+1}, p_{jr}) \qquad \text{by (3.1.3)}$$

$$= g_r^{r+1}(1,j).$$

Therefore the lemma is true for k=r and r+1. Make the induction hypothesis that

$$g_r^{v}(j,1) \geq g_r^{v}(1,j) \quad \text{for} \quad v = r, r+1, \ldots, k-1.$$

Now we prove the hypothesis for $v = k$ under two cases :

Case(i) : $p_{jk} \geq p_{1k}$. Adding this to $g_r^{k-1}(j,1) \geq g_r^{k-1}(1,j)$ (given by induction hypothesis) we get,

$$g_r^{k-1}(j.1) + p_{jk} \geq g_r^{k-1}(1,j) + p_{1k} \qquad \ldots \quad (3.1.5)$$

From (3.1.2) we can write

$$g_r^{k}(j,1) = \min\left\{ g_r^{k-1}(j,1) + p_{jk} \, , \sum_{s=r}^{k-1} p_{1s} \right\}$$

and

$$g_r^{k}(1,j) = \min\left\{ g_r^{k-1}(1,j) + p_{1k} \, , \sum_{s=r}^{k-1} p_{js} \right\}.$$

From these, using (3.1.4) and (3.1.5), we get

$$g_r^{k}(j,1) \geq g_r^{k}(1,j).$$

This proves the lemma under this case.

Case(ii) : $p_{jk} < p_{1k}$. Using this in (3.1.3) for $1 \leq s \leq k-1$ we have,

$$\left.\begin{aligned} p_{1s} &\geq p_{js} \\ \text{and} \quad p_{jk} &\geq p_{is} \end{aligned}\right\} \qquad \ldots \quad (3.1.6)$$

Now consider a general term of $g_r^k(j,l)$ say, $\sum_{s=q+1}^{k} p_{js} + \sum_{s=r}^{q-1} p_{ls}$

for $r < q < k$. Using (3.1.6) we have for every q such that $r \leq q \leq k$,

$$\sum_{s=q+1}^{k} p_{js} + \sum_{s=r}^{q-1} p_{ls} \geq \sum_{s=q+1}^{k-1} p_{js} + p_{jq} + \sum_{s=r}^{q-1} p_{js} = \sum_{s=r}^{k-1} p_{js} \geq g_r^k(l,j).$$

Therefore $g_r^k(j,l) \geq g_r^k(l,j)$.

This completes the proof of Lemma 3.1.1. //

Remark 3.1.2 : Our proof of Lemma 3.1.1 is the direct proof given in Achuthan (1977). Recently Szwarc (1979) provided a simpler proof of Lemma 3.1.1 , using the property of Johnson's permutation of the two machine flow-shop problem obtained by viewing the machines $M_r, \ldots, M_k$ as jobs and jobs j and l as first and second machines.

Lemma 3.1.3 : Let $\partial = (\partial_1, \ldots, \partial_m)$ give the machine available times of the m machines. For a given pair of jobs j and l, and for any pair of machines M_r and M_k such that $1 \leq r < k \leq m$, let

$$\min(p_{jr}, p_{lk}) \leq \min(p_{lr}, p_{jk}).$$

Then $t(\partial; jl; k) \leq t(\partial; lj; k)$, $1 \leq k \leq m$.

Proof : Lemma 3.1.3 follows immediately from Lemma 3.1.1 and equation (3.1.1).//

Theorem 3.1.4 : Let $\partial = (\partial_1, \ldots, \partial_m)$ be the machine available times of the m machines. For a given pair of jobs j and l, and for every pair of machines M_r and M_k such that $1 \leq r < k \leq m$, let

$$\min (p_{jr}, p_{lk}) \leq \min (p_{lr}, p_{jk}).$$

Let γ be any partial permutation of jobs in $N - \{j,l\}$. Then

$$t(\partial ; jl\gamma ; k) \leq t(\partial ; lj\gamma ; k), \quad 1 \leq k \leq m.$$

Proof : Let $\partial^* = (\partial_1^*, \ldots, \partial_m^*)$ be such that

$$\partial_k^* = t(\partial ; jl ; k), \quad 1 \leq k \leq m.$$

Similarly $\hat{\partial} = (\hat{\partial}_1, \ldots, \hat{\partial}_m)$ is such that

$$\hat{\partial}_k = t(\partial ; lj ; k), \quad 1 \leq k \leq m.$$

Now using the critical path networks of $jl\gamma$ and $lj\gamma$ one can easily see that for $1 \leq k \leq m$ we have,

$$t(\partial ; jl\gamma ; k) = t(\partial^* ; \gamma ; k), \quad \ldots \quad (3.1.7)$$

and $$t(\partial ; lj\gamma ; k) = t(\hat{\partial} ; \gamma ; k). \quad \ldots \quad (3.1.8)$$

From Lemma 3.1.3 we have $\partial_i^* \leq \hat{\partial}_i$ for $1 \leq i \leq m$.

Again using the critical path network of γ we have

$$t(\partial^*\ ;\ \gamma\ ;\ k) \leq t(\hat{\partial};\ \gamma\ ;\ k),\quad 1 \leq k \leq m.$$

Now from (3.1.7) and (3.1.8) Theorem 3.1.4 follows.//

Corollary 3.1.5 : For a given pair of jobs j and l, and for every pair of machines M_r and M_k such that $1 \leq r < k \leq m$, let

$$\min\ (\ p_{jr},\ p_{lk}\ ) \leq \min\ (p_{lr},\ p_{jk}).$$

Let σ be any partial permutation of jobs in $N - \{j,l\}$ and γ another partial permutation of jobs in $N - \sigma - \{j,l\}$. Then

$$t(\sigma jl\gamma\ ;\ k) \leq t(\sigma lj\gamma\ ;\ k),\quad 1 \leq k \leq m.$$

Proof : The corollary follows from Theorem 3.1.4 if we choose ∂ such that $\partial_k = t(\sigma\ ;\ k)$, $1 \leq k \leq m$. //

Remark 3.1.6 : Corollary 3.1.5 gives a dominance rule which could be used while solving a $(n/m/F/F_{max})$ problem by a branch-and-bound search method.

Theorem 3.1.7 : Let the permutation $\pi^* = (1,2,\ \ldots,\ n)$ be satisfying the following condition : If $1 \leq j < l \leq n$, then $\min(p_{jr},\ p_{lk}) \leq \min(p_{lr},\ p_{jk})$ for every pair of machines M_r and M_k such that $1 \leq r < k \leq m$.

Then π^* is an optimal permutation for the $(n/m/F/F_{max})$ problem.

Proof : We want to prove $t(\pi^*; m) \leq t(\pi; m)$ for every permutation π of the jobs in N. Consider any $\pi = (\pi(1),\ldots,\pi(n)) \neq \pi^*$. There exists i such that $\pi(i-1) > \pi(i)$, for, otherwise the permutation π has the property $\pi(1) < \pi(2) < \ldots < \pi(n)$ which implies $\pi(i) = i$ for every i and hence $\pi = \pi^*$. Now construct π' from π by interchanging $\pi(i)$ and $\pi(i-1)$. Using the given condition of the theorem and the result of Corollary 3.1.5, we get

$$t(\pi'; m) \leq t(\pi; m).$$

Now π' either coincides with π^* or it has one fewer pairs of elements which disagree in ordering with those of π^*. One may proceed, inductively, decreasing the number of pairs of elements which disagree in ordering with those of π^* and at each step without increasing the value of the completion time of the last job on the last machine.

This completes the proof of Theorem 3.1.7. //

Corollary 3.1.8 : In a $(n/m/F/F_{max})$ problem, let π be a Johnson's permutation for every two machine flow-shop problem (M_r, M_k), $1 \leq r < k \leq m$. Then π is an optimal permutation of the $(n/m/F/F_{max})$ problem.

Proof : Note that without loss of generality jobs can be renamed such that Theorem 3.1.7 is applicable. Then the corollary follows.//

3.2 A Special Case of the $(n/m/F/F_{max})$ Problem :

The special case considered in this section is a $(n/m/F/F_{max})$ problem where the following conditions called (*) hold for any pair of jobs j and l :

Either

$$\left.\begin{aligned} &\min(p_{jr}, p_{lk}) \geq \min(p_{lr}, p_{jk}) \text{ for } 1 \leq r < k \leq m \\ &\qquad\qquad (r,k) \neq (1,m) \\ \text{Or} \\ &\min(p_{jr}, p_{lk}) \leq \min(p_{lr}, p_{jk}) \text{ for } 1 \leq r < k \leq m \\ &\qquad\qquad (r,k) \neq (1,m) \end{aligned}\right\} \quad (*)$$

Henceforth we refer to this special case as $(n/m/F/*/F_{max})$ problem. When $m = 3$, this special case reduces to case 6 of Table 2.1.1 handled by Burns et al. (1975). We later use the following lemma due to Johnson (1954).

Lemma 3.2.1 Johnson (1954) : Suppose A_i, A_j, A_l, B_i, B_j, B_l are numbers such that

$$\min(A_i, B_j) \leq \min(A_j, B_i) \quad \ldots \quad (3.2.1)$$

and

$$\min(A_j, B_l) \leq \min(A_l, B_j) \quad \ldots \quad (3.2.2)$$

Then, $\min(A_i, B_l) \leq \min(A_l, B_i)$ unless $A_j = B_j$, $B_j < A_i$, $B_j \leq B_i$, $A_j < B_l$ and $A_j \leq A_l$. //

Remark 3.2.2 : Suppose (3.2.1) and (3.2.2) hold. If $\min(A_i, B_1) > \min(A_1, B_i)$ then both (3.2.1) and (3.2.2) hold as equalities.

Remark 3.2.3 : If both (3.2.1) and (3.2.2) hold as strict inequalities then $\min(A_i, B_1) < \min(A_1, B_i)$.

Define the index set I by

$$I = \{(u,v) \neq (1,m) \mid 1 \leq u < v \leq m\}.$$

For a fixed (r,k) such that $1 \leq r < k \leq m$ define

$$I_{r,k} = \{(u,v) \in I \mid 1 \leq u < r \text{ or } u = r < v \leq k-1\},$$

and a collection of two machine flow-shop problems by

$$\mathcal{Z}_{r,k} = \{(M_u, M_v) \mid (u, v) \in I_{r,k}\}.$$

In order to simplify the notation we use $j \overset{u,v}{\models\!=} 1$ to mean

$$\min(p_{ju}, p_{1v}) \leq \min(p_{1u}, p_{jv}).$$

Further in the above expression if '$\leq$' is replaced by $<, \geq, >$ or $=$ then we denote it by $j \overset{u,v}{\models\!-} 1$; $j \overset{u,v}{=\!=} 1$; $j \overset{u,v}{-\!|} 1$ or $j \overset{u,v}{|\!=\!|} 1$ respectively.

Lemma 3.2.4 : Consider a $(n/m/F/*/F_{max})$ problem. Fix (r,k) such that $1 \leq r < k \leq m$ and $(r,k) \neq (1,m)$ and $(1,2)$. Let $Z = Z_{r,k}$.

Let $\pi = (\pi(1), \ldots, \pi(n))$ be a Johnson's permutation for every two machine flow-shop problem (M_u, M_v) in Z. With reference to the problem (M_r, M_k) and permutation π, assume that there exists at least one pair (i', i) such that $1 \leq i' < i \leq n$ and

$$\pi(i') \overset{r,k}{\dashv} \pi(i) \qquad \ldots \qquad (3.2.3)$$

Among such pairs (i', i), choose a pair (i', i) such that $(i - i')$ is minimum.

Then,

$$\pi(i') \overset{u,v}{\models} \pi(i), \ \forall\, (M_u, M_v) \in Z \ . \qquad \ldots \qquad (3.2.4)$$

Further if $i - i' > 1$, then

$$\left.\begin{aligned} \pi(i') &\overset{r,k}{\models} \pi(s) \ ; \\ \pi(s) &\overset{r,k}{\models} \pi(i), \end{aligned}\right\} \ \forall\ s \text{ such that } i' < s < i \qquad \ldots (3.2.5)$$

and

$$\left.\begin{aligned} &\underline{\text{either}} \quad \pi(i') \overset{u,v}{\models} \pi(i'+1), \ \forall\, (M_u, M_v) \in Z \\ &\underline{\text{or}} \quad \pi(i-1) \overset{u,v}{\models} \pi(i), \ \forall\ (M_u, M_v) \in Z \ . \end{aligned}\right\} \qquad \ldots \qquad (3.2.6)$$

Proof : Since π is a Johnson's permutation for every two machine flow-shop problem (M_u, M_v) in Z, we have

$$\pi(i') \overset{u,v}{\models} \pi(i'+1), \ \forall\ (M_u, M_v) \in Z \qquad \ldots \qquad (3.2.7)$$

$$\pi(i-1) \overset{u,v}{\models} \pi(i), \ \forall\, (M_u, M_v) \in Z \ , \qquad \ldots \qquad (3.2.8)$$

$$\text{and} \quad \pi(i') \overset{u,v}{\models} \pi(i), \forall\ (M_u, M_v) \in Z \ . \qquad \ldots \qquad (3.2.9)$$

It is easy to check that condition (*), (3.2.3) and (3.2.9) together prove (3.2.4).

From the choice of the pair (i', i) with reference (M_r, M_k) when $i - i' > 1$, we have

$$\left.\begin{aligned} \pi(i') &\overset{r,k}{\models} \pi(s), \\ \text{and} \quad \pi(s) &\overset{r,k}{\models} \pi(i) \end{aligned}\right\} \ \forall\ s \text{ such that } i' < s < i. \quad \ldots(3.2.10)$$

Now for every s such that $i' < s < i$, invoking Remark 3.2.2, to (3.2.10) and (3.2.3) we get (3.2.5).

It remains to prove (3.2.6). Suppose (3.2.6) is not true. Then in view of (3.2.7) and (3.2.8) there exist $(M_{u'}, M_{v'})$ and $(M_{u''}, M_{v''})$ both belonging to Z such that

$$\pi(i') \overset{u',v'}{\models} \pi(i'+1) \qquad \ldots \qquad (3.2.11)$$

$$\text{and} \quad \pi(i-1) \overset{u'',v''}{\models} \pi(i). \qquad \ldots \qquad (3.2.12)$$

$$\text{First note that,} \quad \pi(i'+1) \overset{u',v'}{\models} \pi(i). \qquad \ldots \qquad (3.2.13)$$

Otherwise, since π is a Johnson's permutation for the problem $(M_{u'}, M_{v'})$ we will have $\pi(i'+1) \overset{u',v'}{\models} \pi(i)$ and to this and (3.2.11) we invoke

Remark 3.2.3 to get a contradiction to (3.2.4) for the problem $(M_{u'}, M_{v'})$.

Similarly for the problem $(M_{u''}, M_{v''})$, we have

$$\pi(i') \;\Big\{\!\!=\!\!=\!\!\Big\}^{u'',v''}\; \pi(i-1). \qquad \ldots \quad (3.2.14)$$

Next we will prove

$$\left.\begin{aligned} & p_{\pi(i),u'} < p_{\pi(i'),v'} , \\ \text{and} \quad & p_{\pi(i),u'} = p_{\pi(i),v'} . \end{aligned}\right\} \qquad \ldots \quad (3.2.15)$$

Proof of (3.2.15) :

First suppose $p_{\pi(i),u'} \geq p_{\pi(i'),v'}$

$\Longrightarrow \min(p_{\pi(i'),u'} , p_{\pi(i),v'}) = p_{\pi(i'),v'}$, (using (3.2.4) for $(M_{u'}, M_{v'}) \in Z$)

$\Longrightarrow \min(p_{\pi(i'),u'} , p_{\pi(i'+1),v'}) < \min(p_{\pi(i'+1),u'} , p_{\pi(i'),v'})$

$$\leq p_{\pi(i'),v'}$$

$$= \min(p_{\pi(i'),u'} , p_{\pi(i),v'}),$$

(using (3.2.11))

$\Longrightarrow p_{\pi(i'+1),v'} < \min(p_{\pi(i'+1),u''}, p_{\pi(i'),v'}) \leq p_{\pi(i'),v'}$

$$= \min(p_{\pi(i'),u'} , p_{\pi(i),v'})$$

(otherwise we will have $p_{\pi(i'),u'} < p_{\pi(i'),u'}$)

$\Longrightarrow p_{\pi(i'+1),v'} < \min(p_{\pi(i'+1),u'} , p_{\pi(i),v'})$

$= \min(p_{\pi(i),u'} , p_{\pi(i'+1),v'}) \leq p_{\pi(i'+1),v'}$

(using 3.2.13)

$\Longrightarrow$ contradiction $\Longrightarrow \underline{p_{\pi(i),u'} < p_{\pi(i'),v'}}$.

Now using $p_{\pi(i),u'} < p_{\pi(i'),v'}$ in (3.2.4) for $(M_{u'}, M_{v'})$

We get $p_{\pi(i),u'} \leq p_{\pi(i),v'}$.

Now suppose $p_{\pi(i),u'} < p_{\pi(i),v'}$

$\Longrightarrow p_{\pi(i),u'} = p_{\pi(i'),u'}$, from (3.2.4) for $(M_{u'}, M_{v'})$ and

$p_{\pi(i),u'} < p_{\pi(i'),v'}$

$\Longrightarrow \min(p_{\pi(i),u'}, p_{\pi(i'+1),v'} = \min(p_{\pi(i'),u'}, p_{\pi(i'+1),v'})$

$< p_{\pi(i'+1),u'}$, from (3.2.11)

$\Longrightarrow \min(p_{\pi(i),u'} , p_{\pi(i'+1),v'}) < \min(p_{\pi(i'+1),u'} , p_{\pi(i),v'})$,

$\Longrightarrow$ contradiction to (3.2.13) $\Longrightarrow \underline{p_{\pi(i),u'} = p_{\pi(i),v'}}$.

Thus (3.2.15) is proved.

Similarly using (3.2.4) for $(M_{u''}, M_{v''})$, (3.2.12) and (3.2.14) we can prove that

$$\left.\begin{aligned} & p_{\pi(i'),v''} < p_{\pi(i),u''} \\ \text{and} \quad & p_{\pi(i'),v''} = p_{\pi(i'),u''} \end{aligned}\right\} \quad \ldots \quad (3.2.16)$$

Now using (3.2.15) and (3.2.16) note that

$$\min(p_{\pi(i'),u''}, p_{\pi(i),v'}) < \min(p_{\pi(i),u''}, p_{\pi(i'),v'})$$

$$\text{i.e.} \quad \pi(i') \overset{u'',v'}{\vdash} \pi(i) . \quad \ldots \quad (3.2.17)$$

Now it is clear from (3.2.17) that $u'' \neq v'$. If $u'' < v'$ then (3.2.3) and (3.2.17) together contradict condition (*). If $u'' > v'$, then note that $u' < v' < u'' < v''$. Therefore $(M_{v'}, M_{u''}) \in Z$. Now (3.2.17) contradicts (3.2.4) for $(M_{v'}, M_{u''})$.

Thus (3.2.6) is proved. This completes the proof of Lemma 3.2.4.//

Remark 3.2.5 : In Lemma 3.2.4, fix $(r,k) = (1,m)$ and assume all other given conditions of Lemma 3.2.4. Further assume that (3.2.4) holds. Now if $i - i' > 1$ then (3.2.5) and (3.2.6) hold. //

Lemma 3.2.6 : In a $(n/m/F/*/F_{max})$ problem, there exists a $\pi = (\pi(1), \ldots, \pi(n))$ which is Johnson's permutation for all the two machine flow-shop problems $(M_u, M_v) \in Z_{1,m}$.

Proof : We provide a constr ctive proof of this lemma. The general step (r,k) explained below corresponds to the two machine flow-shop problem (M_r, M_k). After performing the initial step given below, the general steps (r,k) are performed in the order (1,3), (1,4), ..., (1, m-1); (2,3), (2,4), ..., (2,m); ...; (i, i+1), ..., (i,m); ..., (m-2, m-1), (m-2,m); (m-1,m).

Initial Step : Find a Johnson's permutation $\pi = (\pi(1), \ldots, \pi(n))$ for the two machine flow-shop problem (M_1, M_2) and go to general step (1,3) with this π.

General Step (r,k) :

(a) Let π be the johnson's permutation for every two machine flow-shop problem $(M_u, M_v) \in Z_{r,k}$, got from the preceding general step. Note that $(M_r, M_k) \notin Z_{r,k}$.

(b) If π is a Johnson's permutation for the problem (M_r, M_k), go to the next general step with this π.

(c) If π is not a Johnson's permutation for the problem (M_r, M_k), then there exists atleast one pair (i',i) such that $1 \leq i' < i \leq n$ and $\pi(i') \overset{r,k}{\dashv} \pi(i)$. Among such pairs (i',i) choose the one for which (i - i') is minimum.

Case(i) i - i' = 1 : Obtain $\bar{\pi}$ from π by interchanging $\pi(i')$ and $\pi(i)$. Invoking Lemma 3.2.4 note that (3.2.4) holds and hence $\bar{\pi}$ is a Johnson's permutation for every problem (M_u, M_v) in $Z_{r,k}$.

Case(ii) $i - i' > 1$: Again invoking Lemma 3.2.4 , we note that (3.2.4) and (3.2.6) hold. Hence observe that $\pi' = (\pi'(1), \ldots, \pi'(n))$ is Johnson's permutation for every problem (M_u, M_v) in $Z_{r,k}$ where π' is obtained from π by interchanging.

either $\pi(i')$ and $\pi(i'+1)$

or $\pi(i-1)$ and $\pi(i)$.

Now π' is still not Johnson's permutation for the problem (M_r, M_k). Hence there exists at least one pair (s', s) such that $1 \leq s' < s \leq n$, and $\pi'(s') \xrightarrow{r,k} \pi'(s)$. Among such pairs (s', s), if we choose the one with smallest $(s - s')$, then we have $s - s' = i - i' - 1$. Once again the procedure of Case(ii) can be repeated with reference to π' if necessary. Thus after a finite number of repetitions of Case(ii) and finally by Case(i) we will get,

$$\bar{\pi} = (\pi(1), \ldots, \pi(i'-1), \pi(i'+1), \ldots, \pi(s^*), \pi(i), \pi(i'), \pi(s^*+1), \ldots, \pi(i-1), \pi(i+1), \ldots, \pi(n))$$

such that $\bar{\pi}$ is Johnson's permutation for every problem (M_u, M_v) in $Z_{r,k}$. Note that $i' < s^* < i$.

Now if $\bar{\pi}$ is not Johnson's permutation for the problem (M_r, M_k), we get by (3.2.5) of Lemma 3.2.4 that $\bar{\pi}$ has one less pair of elements than π, not satisfying the condition for the permutation to be Johnson's permutation for the problem (M_r, M_k).

Now step(c) can be repeated with reference to $\bar{\pi}$ if necessary and thus in a finite number of steps we will end up in (b) leading to the next general step. Thus at the end of the last general step (m-1, m) we will get the permutation required in the lemma. This completes the proof of Lemma 3.2.6. //

Lemma 3.2.7 : In a $(n/m/F/F_{max})$ problem, for a given pair of jobs j and l, assume

$$j \overset{u,v}{\models\!\models} l, \quad \forall\, (M_u, M_v) \in Z_{1,m} . \qquad \ldots \qquad (3.2.18)$$

If $j \overset{1,m}{\longrightarrow\!\!|}\ l$ then $p_{ju} = p_{lu}$, $2 \leq u \leq m-1$ and

$$j \overset{u,v}{\models\!\models} l, \quad \forall\, (M_u, M_v) \in Z_{1,m} .$$

Proof : From (3.2.18), for $2 \leq u \leq m-1$ we can write

$$1 \overset{j,l}{\models\!\models} u \qquad \ldots \qquad (3.2.19)$$

and $$u \overset{j,l}{\models\!\models} m. \qquad \ldots \qquad (3.2.20)$$

Further we can write $j \overset{1,m}{\longrightarrow\!\!|}\ l$ as

$$1 \overset{j,l}{\longrightarrow\!\!|}\ m \qquad \ldots \qquad (3.2.21)$$

Now invoking Lemma 3.2.1 and Remark 3.2.2 to (3.2.19), (3.2.20) and (3.2.21) we get $p_{ju} = p_{lu}$,

$$1 \overset{j,l}{\models\!\models} u \text{ and } u \overset{j,l}{\models\!\models} m \text{ for } 2 \leq u \leq m-1.$$

Using $p_{ju} = p_{lu}$, $2 \leq u \leq m-1$ in (3.2.18) we get

$$j \overset{u,v}{\models} 1 \quad \text{for} \quad 2 \leq u < v \leq m-1.$$

Thus Lemma 3.2.7 is proved. //

Theorem 3.2.8 : In a $(n/m/F/*/F_{max})$ problem, there exists a Johnson's permutation of the two machine flow-shop problem (M_1, M_m) which solves the $(n/m/F/*/F_{max})$ problem optimally.

Proof : In the problem $(n/m/F/*/F_{max})$, by Lemma 3.2.6 , there exists $\pi = (\pi(1), \ldots, \pi(n))$ a Johnson's permutation for every two machine flow-shop problem $(M_u, M_v) \in Z_{1,m}$. Consider such a permutation π.

If π is a Johnson's permutation for the problem (M_1, M_m), it is clear by Corollary 3.1.8 that π is an optimal solution for the $(n/m/F/*/F_{max})$ problem. Otherwise we know that there exists a pair (i', i) such that $1 \leq i' < i \leq n$ and

$$\pi(i') \overset{1,m}{\models} \pi(i). \qquad \ldots \quad (3.2.22)$$

Among such pairs (i', i), choose the one which corresponds to the smallest $(i - i')$. Since π is Johnson's permutation for the problems (M_u, M_v) in $Z_{1,m}$ we have

$$\pi(i') \overset{u,v}{\models} \pi(i) \ , \ \forall \ (M_u, M_v) \in Z_{1,m} .$$

Invoking Lemma 3.2.7 to this and (3.2.22) we get ,

$$\pi(i') \overset{u,v}{\Longrightarrow} \pi(i), \forall\ (M_u, M_v) \in Z_{1,m} \quad \ldots \quad (3.2.23)$$

Now invoking Remark 3.2.5 we get that when $i-i' > 1$, (3.2.5) and (3.2.6) hold.

Now the arguments of General step (r,k) of Lemma 3.2.6 can be repeated with reference to general step (1,m) to give the required Johnson's permutation for every two machine flow-shop problem (M_u, M_v), $1 \leq u < v \leq m$. Then by Corollary 3.1.8 this permutation is optimal for the $(n/m/F/*/F_{max})$ problem. This proves Theorem 3.2.8. //

For $m=3$, the $(n/m/F/*/F_{max})$ problem was handled by Burns et al. (1975) and they obtained Procedure 'H12' (given in section 2.2 of Chapter II) to solve the problem. Later Szwarc (1977) proved Theorem 3.2.8 for $m=3$.

Using the results of this section, we provide a procedure to solve the $(n/m/F/*/F_{max})$ problem. For any given $(n/m/F/F_{max})$ problem, the suggested procedure finds an optimal solution if condition (*) holds and otherwise the procedure terminates without finding an optimal solution but indicating that the condition (*) is not satisfied. In describing the procedure we use the following notation.

- ((j,l)) and (j,l) denote the unordered and ordered pairs of elements j and l respectively.

- CBS(i) of π denotes the permutation obtained from π by interchanging $\pi(i)$ and $\pi(i+1)$.

- Procedure 'JP $(p_{j1}, p_{j2}, N, \pi^*)$' finds a Johnson's permutation π^* of the two machine flow-shop problem with job set N and processing times p_{j1} and p_{j2} on first and second machines respectively.

Procedure '$(n/m/F/*/F_{max})$' (p_{jk}, π^*)

1. Begin Local π ; Y; $\bar{Y}$; $\bar{\bar{Y}}$; u; v;
2. call Procedure 'JP $(p_{j1}, p_{j2}, N, \pi^*)$'
3. $\bar{Y} := \{ ((j.l)) \mid j \overset{1,2}{\models} l \}$
4. do for u=1, m-1
5. do for v = u+1, m
6. $\pi := \pi^*$; $Y := \{ ((\pi(s'), \pi(s))) \mid s' < s, \pi(s') \overset{u,v}{\models} \pi(s) \}$;
7. $\bar{Y} := \bar{Y} - Y$; $\bar{\bar{Y}} := \{ (s',s) \mid s' < s, \pi(s') \overset{u,v}{\models} \pi(s) \}$;
8. While $\bar{\bar{Y}} \neq \emptyset$ do
9. $(i',i) : (s-s') \in \bar{\bar{Y}}$
10. if $((\pi(i'), \pi(i'+1))) \in \bar{Y}$ then $\pi^* := CBS\ (i')$ of π ;
11. if else $((\pi(i-1), \pi(i))) \in \bar{Y}$ then $\pi^* := CBS(i-1)$ of π ;
12. else print 'the problem does not satisfy (*)' and stop;

13. when $(i-i') = 1$ then $\bar{Y} := \bar{Y} - \{((i',i))\}$;

14. $\pi := \pi^*$; $\bar{\bar{Y}} := \{(s',s) \mid s' < s,\ \pi(s') \xrightarrow{u,v} \pi(s)\}$;

15. end

16. end

17. end

18. end Procedure '$(n/m/F/*/F_{max})$'.

Remark 3.2.9 : In the Procedure '$(n/m/F/*/F_{max})$' the set $\bar{Y}$ can be constructed in $O(n(n-1))$ steps. Similarly the sets Y and $\bar{\bar{Y}}$ can be constructed in $O(n(n-1))$ steps. Thus we can prove that the procedure is of polynomial computational time complexity.

The special case of this section is characterized through condition (*) where a distinguished pair (1,m) is not warrented to satisfy certain conditions. The following example illustrates that there may not exist a common Johnson's permutation for all the two machine flow-shop problems (M_r, M_k), $1 \leq r < k \leq m$, if the distinguished pair (1,m) is replaced by any $(u,v) \neq (1,m)$ in condition (*).

Example 3.2.1 : Let a be a positive real number. Choose any u and v such that $1 \leq u < v < m$ where $m \geq 4$. Define the processing times of jobs 1 and 2 on machines M_u, M_v and M_m by :

jobs	processing times		
	p_{iu}	p_{iv}	p_{im}
1	a+3	a+1	a+1.5
2	a+4	a+2	a+1

Define the processing times of jobs 1 and 2 on machines M_r, $1 \leq r \leq m$, $r \neq u$, v and m as: $p_{1r} = p_{2r} = a$. Now note that condition (*) is satisfied when the distinguished pair (1,m) is replaced by (u,v) in condition (*). But neither $\pi = (1,2)$ nor $\pi' = (2,1)$ is a Johnson's permutation for every two machine flow-shop problem (M_r, M_k), $1 \leq r < k \leq m$.

Similarly examples can be constructed to take care of the case when the distinguished pair (1,m) is replaced by (u,m) for some u such that $2 \leq u \leq m-1$, in condition (*).

3.3 Use of Szwarc's Dominance Criteria :

Several combinatorial methods have been suggested to solve the $(n/m/F/F_{max})$ problem. One of the methods is to generate the undominated permutations and select the best out of them. Different authors generated the undominated permutations by using different dominance criteria. For details of these methods we refer to Baker (1974) and Gupta (1976). The most effective dominance criteria in use has been SDC given in Definition 1.3.10.

Given a partial permutation σ, if σij dominates σj through SDC note that the partial permutation σj can be dropped while generating the undominated permutations.

In the reverse flow-shop problem $(n/m^R/F/F_{max})$ of Definition 1.3.17 , let $t^R(\sigma ; k')$ denote the completion time of the non-delay partial permutation schedule σ on the machine $M_{k'}$, where $k' = m-k+1$, $1 \leq k \leq m$. Using the symmetric properties between the original and this reverse problem (Remark 1.3.19), Szwarc (1971) suggested the following symmetric dominance criteria.

Definition 3.3.1 : Let μ be a partial permutation and i, j jobs such that $i \neq j$, and $i,j \notin \mu$. Let $\Delta^R_{k'} = t^R(\mu^R i\, j; k') - t^R(\mu^R j; k')$ where $k' = m-k+1$, $1 \leq k \leq m$. We say that jiμ dominates jμ through Szwarc's Symmetric Dominance Criteria (SSDC) in case

$$\Delta^R_{(k-1)'} \leq \Delta^R_{k'} \leq p_{ik'} , \quad 2 \leq k \leq m.$$

Remark 3.3.2 Szwarc(1971) : Let jiμ dominate jμ through SSDC. Then

$$t(\pi_1 \pi_2\, ji\, \mu; m) \leq t(\pi_1\, i\, \pi_2\, j\, \mu; m)$$

where π_1 and π_2 are arbitrary partial permutations such that $\pi_1 \cap \pi_2 = \emptyset$ and $\pi_1 \cup \pi_2 = N - \mu \cup \{i,j\}$.

If $ji\mu$ dominates $j\mu$ through SSDC then the permutations ending with $j\mu$ can be dropped while generating the undominated permutations. Thus SDC (SSDC) can be used to identify jobs that are dominated in the first (last) unassigned positions of a two sided partial permutation (σ_1, σ_2).

Note that for σij to dominate σj through SDC it is necessary for job i to satisfy Property A. Similarly for $ji\mu$ to dominate $j\mu$ through SSDC it is necessary for job i to satisfy Property B, where Properties A and B are given below:

Property A : $p_{i1} \leq p_{ik}$, $2 \leq k \leq m$ ... (A)

Property B : $p_{i1'} \leq p_{ik'}$, $k' = m-k+1$ and $2 \leq k \leq m$,

that is, $p_{im} \leq p_{ik}$, $1 \leq k \leq m-1$. ... (B)

Gupta (1975) computed the probability that a job will satisfy the Property (A) assuming that the processing times for the jobs are drawn from a discrete uniform distribution in the range $[0, \beta]$ where $\beta = 9$, 99 and 999. Then he computed $L(S)$, a lower bound of the expected number of undominated permutations. Further Gupta(1975) compared his results with those of Baker (1975) who assumed continuous uniform distribution for the processing time.

Baker (1975) indicated that one could perform probabilistic analysis, similar to the one done by Gupta (1975), for the method of generation of undominated permutations using SDC and SSDC. Here we study this problem and subsequently compare our results with those of Gupta (1975) and Baker (1975). In the following we present a procedure for generating the undominated permutations using SDC and SSDC. This procedure is in the same lines as the Algorithm 6.3 in page 160 of the book by Baker (1974). This Procedure ' UPM for the $(n/m/F/F_{max})$ problem' uses the following notation.

- At any stage of the procedure π^* is the best known solution found so far and it provides an upper bound $F_{max}(\pi^*)$ on the value of the optimal solution.

- A branching rule b associates with a two-sided partial permutation (σ_1, σ_2) a family $b(\sigma_1, \sigma_2)$ of two sided partial permutations where

$$b(\sigma_1, \sigma_2) = \{ (\sigma_1 i, j \sigma_2) \mid i \neq j, \ i \text{ and } j \notin \sigma_1 \cup \sigma_2 \}$$

$$\text{if } |\sigma_1| + |\sigma_2| \leq n-2.$$

In our procedure we will not be branching from a two-sided partial permutation (σ_1, σ_2) such that $|\sigma_1| + |\sigma_2| > n-2$. Observe that $| b(\sigma_1, \sigma_2) | = (n - |\sigma_1| - |\sigma_2|)(n - |\sigma| - |\sigma_2| - 1)$.

- A dominance rule d associates with a two sided partial permutation (σ_1, σ_2) a subset of $b(\sigma_1, \sigma_2)$ denoted by $d(\sigma_1, \sigma_2)$ when $|\sigma_1| + |\sigma_2| \leq n-2$. Let $I(\sigma_1)$ and $I(\sigma_2)$ be subsets of $N-(\sigma_1 \cup \sigma_2)$ where ,

$$I(\sigma_1) = \left\{ j \notin \sigma_1 \cup \sigma_2 \mid \exists\, i \notin \sigma_1 \cup \sigma_2,\ i \neq j \text{ and } \sigma_1 ij \text{ dominates } \sigma_i j \text{ through SDC} \right\}$$

and

$$I(\sigma_2) = \left\{ j \notin \sigma_1 \cup \sigma_2 \mid \exists\, i \notin \sigma_1 \cup \sigma_2,\ i \neq j \text{ and } ji\sigma_2 \text{ dominates } j\sigma_2 \text{ through SSDC} \right\}$$

Define

$$d(\sigma_1, \sigma_2) = \left\{ (\sigma_1 j, 1\sigma_2) \mid j \in I(\sigma_1) \text{ or } 1 \in I(\sigma_2) \right\}.$$

- At any stage of the procedure, Y denotes the set of two-sided partial permutations from which branching is yet to be performed. At the initial stage of the procedure, Y has the singleton two-sided partial permutation $(\emptyset, \emptyset)$. At the termination stage of the procedure, $Y = \emptyset$.

- As used in the earlier chapters, the operation ": $f \in$" in the statement "s : $f \in S$" will mean that $s := s^*$ where $f(s^*) = \min_{s \in S} f(s)$. Similarly ' : $\in$ ' indicates an arbitrary choice.

Procedure 'UPM for the $(n/m/F/F_{max})$ problem' (p_{jk}, π^*)

1. Begin Local F_{max}; (σ_1, σ_2); π^*; Y; $\bar{B}(\sigma_1, \sigma_2)$; $\bar{D}(\sigma_1, \sigma_2)$; (μ_1, μ_2); μ ;
2. $Y := \{(\emptyset, \emptyset)\}$; $\pi^* := \infty$; $F_{max}(\pi^*) := \infty$;
3. while $Y \neq \emptyset$ do
4. $(\sigma_1, \sigma_2) :\in Y$; $Y := Y - \{(\sigma_1, \sigma_2)\}$; $\bar{B}(\sigma_1, \sigma_2) := b(\sigma_1, \sigma_2)$;
5. $\bar{D}(\sigma_1, \sigma_2) := d(\sigma_1, \sigma_2)$; $\bar{B}(\sigma_1, \sigma_2) := \bar{B}(\sigma_1, \sigma_2) - \bar{D}(\sigma_1, \sigma_2)$;
6. if $|\sigma_1| + |\sigma_2| < n-3$ then $Y := Y \cup \bar{B}(\sigma_1, \sigma_2)$;
7. if else $|\sigma_1| + |\sigma_2| = n-3$ then
8. while $\bar{B}(\sigma_1, \sigma_2) \neq \emptyset$ do
9. $(\mu_1, \mu_2) :\in \bar{B}(\sigma_1, \sigma_2)$; $\bar{B}(\sigma_1, \sigma_2) := \bar{B}(\sigma_1, \sigma_2) - \{(\mu_1, \mu_2)\}$;
10. $\mu := (\mu_1 \; j \; \mu_2)$ for $j \notin \mu_1 \cup \mu_2$;
11. $\pi^* := F_{max} \in \{\pi^*, \mu\}$;
12. end
13. else (i.e. $|\sigma_1| + |\sigma_2| = n-2$) then $\pi^* : F_{max} \in \bar{B}(\sigma_1, \sigma_2) \cup \{\pi^*\}$;
14. end
15. end Procedure 'UPM for the $(n/m/F/F_{max})$ problem.

3.3.1 Probabilistic Analysis :

Define a job i to be a potentially dominant job if it satisfies either Property A or Property B. Assuming the discrete uniform distribution for the processing times, the following theorem gives the probability that a job will be a potentially dominant job.

Theorem 3.3.3 : Let X_i, $1 \leq i \leq m$ be independent, identically distributed random variables having uniform probability distribution defined over the discrite sample space $\{0, 1, \ldots, \beta\}$.

Then the probability $p = \Pr\{\min(X_1, X_m) \leq X_k, \quad 1 \leq k \leq m\}$ is given by

$$p = \frac{1}{(\beta+1)^m} \left\{ (\beta+1)^{m-1} + \sum_{k=1}^{\beta} k\left((k+1)^{m-2} + k^{m-2}\right) \right\}.$$

Proof : We know that

$$\Pr\{X_k = u\} = \frac{1}{\beta+1}, \quad 0 \leq u \leq \beta, \quad 1 \leq k \leq m,$$

and

$$\Pr\{X_k \geq u\} = \frac{\beta - u + 1}{\beta + 1}, \quad 0 \leq u \leq \beta, \quad 1 \leq k \leq m.$$

Now,

$$p = \sum_{u=0}^{\beta} \sum_{v=0}^{\beta} \Pr\{\min(X_1, X_m) \leq X_k, \ 1 \leq k \leq m \mid X_1 = u, \ X_m = v\}$$

$$= \sum_{u=0}^{\beta} \left[\sum_{v=0}^{u-1} \Pr\left\{X_1=u\right\} \cdot \Pr\left\{X_m=v\right\} \cdot \Pr\left\{v \le X_k, \quad 2 \le k \le m-1\right\} \right.$$

$$\left. + \sum_{v=u}^{\beta} \Pr\left\{X_1=u\right\} \cdot \Pr\left\{X_m=v\right\} \cdot \Pr\left\{u \le X_k, \quad 2 \le k \le m-1\right\} \right]$$

$$= \sum_{u=0}^{\beta} \left\{ \sum_{v=0}^{u-1} \frac{1}{1+\beta} \cdot \frac{1}{1+\beta} \cdot \left(\frac{\beta - v + 1}{\beta + 1} \right)^{m-2} + \sum_{v=u}^{\beta} \frac{1}{1+\beta} \cdot \frac{1}{1+\beta} \cdot \left(\frac{\beta - u + 1}{\beta + 1} \right)^{m-2} \right\}$$

$$= \frac{1}{(1+\beta)^m} \sum_{u=0}^{\beta} \left\{ \sum_{v=0}^{u-1} (\beta - v + 1)^{m-2} + (\beta - u + 1)^{m-1} \right\}$$

$$= \frac{1}{(1+\beta)^m} \left\{ \sum_{k=1}^{\beta} k(k+1)^{m-2} + \sum_{k=1}^{\beta+1} k^{m-1} \right\}$$

$$= \frac{1}{(1+\beta)^m} \left\{ (\beta+1)^{m-1} + \sum_{k=1}^{\beta} k \left((k+1)^{m-2} + k^{m-2} \right) \right\}.$$

This proves Theorem 3.3.3. //

Let $p_g = \Pr\left\{X_1 \le X_k, \quad 1 \le k \le m\right\}$. Gupta(1975) computed p_g for various values of β and m. Note that p_g gives the probability that a job will satisfy Property(A). Baker (1975) also computed p_g with uniform discrete distribution replaced by uniform continuous distribution over the interval $[0, \beta]$.

Table 3.3.1 gives the values of p for different m and compares them with the values of p_g given by Gupta(1975) and Baker(1975).

Table 3.3.1 : Probability of a potentially dominant job

m	β = 9		β = 99		β = 999		p_g(Baker)
	p	p_g(Gupta)	p	p_g(Gupta)	p	p_g(Gupta)	
2	1.0000	0.5500	1.0000	0.5050	1.0000	0.5005	0.5000
3	0.7150	0.3850	0.6717	0.3384	0.6672	0.3338	0.3333
4	0.5665	0.3025	0.5067	0.2550	0.5007	0.2505	0.2500
5	0.4764	0.2533	0.4075	0.2050	0.4008	0.2005	0.2000
6	0.4163	0.2208	0.3365	0.1717	0.3336	0.1672	0.1667
7	0.3736	0.1978	0.2961	0.1479	0.2866	0.1434	0.1429

From Table 3.3.1 , it is observed that Baker's computation of p_g can work as good approximation for the discrete case only if $\beta \geq 999$. The values of p computed according to Theorem 3.3.3 are approximately twice that of p_g(Gupta's).

Suppose there are r potentially dominant jobs out of the n jobs and they occupy r positions of a permutation under most favourable conditions. Then there will be no other job that can reduce the set of undominated permutations. Thus when there are r potentially dominant jobs, there will be at least (n-r)! permutations generated. A job is either potentially dominant or not. Therefore, the probability distribution of r potentially dominant jobs is binomial with parameters p and n. Thus the probability of r potentially

dominant jobs out of n jobs is $b_r = {}^{n}C_r\, p^r\, (1-p)^{n-r}, \quad 0 \leq r \leq n.$

Now a lower bound for the expected number of permutations in the undominated set is given by $L(S) = \sum_{r=0}^{k} b_r\, (n-r)!$

$$= n! \sum_{r=0}^{k} \left\{ p^r (1-p)^{n-r} / r! \right\}$$

where k is the number of fixed positions in a two-sided partial permutation up to which SDC or SSDC is used. When n is even, for a two-sided partial permutation (σ_1, σ_2) with $|\sigma_1| + |\sigma_2| = n-2$, we shall not use SDC or SSDC in our procedure. Instead we evaluate the F_{max} value for the two possible completions of (σ_1, σ_2) and choose the best out of them. Thus when n is even we will take the value of k as n-2. When n is odd, for a two-sided partial permutation (σ_1, σ_2) with $|\sigma_1| + |\sigma_2| = n-1$, without using SDC or SSDC, we evaluate the F_{max} value for the unique completion of (σ_1, σ_2). Thus when n is odd we take k as n-1. We compute the value of L(S) for ranging values of m, n and β. The values of L(S) are presented in Table 3.3.2 along with the corresponding values given by Gupta (1975) and Baker (1975). Table 3.3.3 presents the detailed behaviour of the L(S) for various values of m and n when $\beta = 99$.

Table 3.3.2 : Lower bound on the size of the set of undominated permutations

n x m	k	L(S)						Baker
		β = 9		β = 99		β = 999		
			Gupta		Gupta		Gupta	
5x3	4	2.47	19.73	3.34	25.37	3.14	25.98	26.0
5x5	4	11.70	39.10	17.42	49.31	18.09	50.37	50.5
6x3	4	4.22	72.82	6.58	100.72	6.88	103.85	104.2
6x6	4	58.05	213.90	101.99	286.06	103.99	293.66	295.5
7x3	6	9.32	313.64	15.96	466.57	16.85	484.33	486.3
7x7	6	346.27	1378.38	657.32	1955.35	708.62	2017.00	2023.8
10x3	8	157.49	52527.75	408.67	97304.13	449.23	1030.92	-
10x7	8	61278.76	512249.75	165075.70	870997.94	185284.43	912928.25	-

Table 3.3.3 : Values of L(S) when β = 99

m \ n	5	6	7	8	9	10
3	3.34	6.58	15.96	41.91	124.34	408.67
4	9.59	28.87	100.03	395.03	1754.09	8653.60
5	17.42	61.91	256.95	1217.98	6494.78	38480.53
6	25.58	101.99	473.78	2514.90	15017.84	99643.94
7	31.56	133.39	657.32	3701.61	23450.82	165075.70

Table 3.3.2 shows that as the number of jobs increases the size of the set of undominated permutations increases. It may be noted that for machine size 3, the lower bound L(S) is considerably low even for job size 10. On the basis of L(S) one might conclude that the procedure based on both SDC and SSDC may be used for number of jobs less than or equal to 7 and number of machines less than or equal to 7. Gupta (1975) observed that his L(S) may be an underestimate of the actual realisation. Similarly the L(S) computed by us may also be an underestimate of the actual realisation by the procedure suggested. However, suppose we make the assumption that the ratio of Gupta's L(S) and our L(S) remains the same when L(S) is replaced by the actual realisations of the respective procedures. Then from Table 3.3.2 it is clear that the procedure using both SDC and SSDC is far better than the procedure using SDC alone. Lageweg et al.(1978) observed that a branch-and-bound search procedure solves a $(n/m/F/F_{max})$ problem fairly efficiently when SDC is used. Thus from the preceding comment, we expect that a branch-and-bound search procedure with SDC and SSDC incorporated, will provide more computationally satisfactory performance. An experimental comparison of these two branch-and-bound procedures could not be accomplished as a fast computer is not available. However, in the following, we present the salient features necessary to write down the branch-and-bound procedure using both SDC and SSDC. In this suggested procedure we maintain many of the useful features present in the best procedure given by Lageweg et al. (1978).

- The best solution π^* found so far provides an upper bound $F_{max}(\pi^*)$ on the value of the optimal solution. Initially π^* is taken to be the best permutation among the Johnson's permutations of the two machine flow-shop problems $\left(\sum_{k=1}^{l} M_k , \sum_{k=m+1-l}^{m} M_k\right)$ for $1 \leq l \leq m-1$. This requires a procedure in $O(mn \log n)$ steps. This rule was suggested by Campbell et al. (1970) as a heuristic.

- Branching rule b is as defined for the Procedure 'UPM for the $(n/m/F/F_{max})$ problem'.

- A bounding rule lb associates with a two-sided partial permutation (σ_1, σ_2) a lower bound $lb(\sigma_1, \sigma_2) \leq F_{max}(\sigma_1 \pi \sigma_2)$ for every $\sigma_1 \pi \sigma_2$, a completion of (σ_1, σ_2), where $lb(\sigma_1, \sigma_2)$ is defined in the following lines.

Define $1\partial(\sigma_1, \sigma_2; 1) = t(\sigma_1; 1) + \sum_{j \in \overline{\sigma_1 \cup \sigma_2}} p_{j1}$.

For $2 \leq u \leq m$ define,

$$1\partial(\sigma_1, \sigma_2; u) = LB(\sigma_1; (1,u), \square o \square)$$

given in section 1.4 of Chapter I with the slight change that the jobs not yet fixed will be $\overline{\sigma_1 \cup \sigma_2}$ instead of $\bar{\sigma}_1$ and the machines under consideration are M_i, $1 \leq i \leq u$ where M_1 and M_u are the bottleneck machines.

Note that $1\partial(\sigma_1, \sigma_2; i)$, $1 \leq i \leq m$ defined above is a lower bound of the earliest time the machine M_i is available, after processing all the jobs in $\bar{\sigma}_2$ with the restriction that the jobs in σ_1 are processed first in that order.

Let $\hat{\partial} = (\hat{\partial}_1, \ldots, \hat{\partial}_m)$, where

$$\hat{\partial}_i = 1\partial(\sigma_1, \sigma_2; i), \quad 1 \leq i \leq m.$$

Define $lb(\sigma_1, \sigma_2) = t(\hat{\partial}; \sigma_2; m)$.

It is easy to verify that $lb(\sigma_1, \sigma_2) \leq F_{max}(\sigma_1 \pi \sigma_2)$ for every $\sigma_1 \pi \sigma_2$, a completion of (σ_1, σ_2). Elimination of (σ_1, σ_2) occurs if $lb(\sigma_1, \sigma_2) \geq F_{max}(\pi^*)$. Note that to compute $1\partial(\sigma_1, \sigma_2; u)$ we need to find the Johnson's permutations of the two machine flow-shop problems $(\sum_{k=1}^{u-1} M_k, \sum_{k=2}^{u} M_k)$ for $2 \leq u \leq m$. We assume that this is performed at the root node $(\emptyset, \emptyset)$. Then this requires a procedure in $O((m-1)\, n \log n)$ steps. For a two-sided partial permutation (σ_1, σ_2) let us assume that we store $|\sigma_1|$; $|\sigma_2|$; $t(\sigma_1; k)$, $1 \leq k \leq m$; $lb(\sigma_1, \sigma_2)$; (σ_1, σ_2) and $\overline{\sigma_1 \cup \sigma_2}$. Subsequently when the two-sided partial permutation (σ_1, σ_2) is generated, we need $\hat{\partial}$ to be computed and this will be done by a procedure in $O((2n(m-1)+m(m+1))(n-|\sigma_1|-|\sigma_2|))$ steps. Thus $lb(\sigma_1, \sigma_2)$ will be computed in

$$O((2n(m-1) + m(m+1))(n-|\sigma_1| - |\sigma_2|) + m|\sigma_2|) \text{ steps.}$$

- Dominance rule d is as defined for the Procedure 'UPM for the $(n/m/F/F_{max})$ problem'. In addition to the already mentioned storage we assume that we store $t^R(\sigma_2^R; k')$ for $k' = m - k + 1$, $1 \leq k \leq m$. Then it is easy to see that construction of $d(\sigma_1, \sigma_2)$ requires a procedure in $O(2m(n-|\sigma_1'|-|\sigma_2|)(n-|\sigma_1|-|\sigma_2|-1))$ steps.

- Frontier search selects a two-sided partial permutation with minimal lower bound for further branching, from among the partial permutations which have so far been neither eliminated nor led to branching.

- A predicate ρ will be used to recognise a complete permutation so that π^* can be improved, if possible.

Now it is easy to write down the branch-and-bound procedure under consideration.

CHAPTER IV

OPTIMAL FLOW-SHOP SCHEDULING WITH EARLINESS AND TARDINESS PENALTIES

4.0 Introduction :

Sidney (1977) indicated situations of scheduling problems where costs involve both earliness and tardiness of the individual jobs being scheduled. For example, in PERT-CPM projects each job j has an associated early start time a_j and late finish time b_j such that (1) the entire project will have to be started early (or alternatively, extra resources will have to be applied to the predecessors of job j) in order to begin job j before time a_j; or (2) the entire project will be delayed beyond its due date unless job j is completed before time b_j. Another example of such scheduling problems arises in the production of perishable goods.

Sidney (1977) handled the single machine problem under certain assumptions on the target start times, the due dates and the penalty functions of the earliness and tardiness. His objective function was the maximum penalty.

In section 4.1 of this chapter using the same objective function we study the scheduling problem under very general conditions. Subsequently in sections 4.2 to 4.6 we restrict our attention to the flow-shop version of this scheduling problem. In section 4.2 after proving some interesting results, we show that it is not enough to consider

the permutation schedules alone while solving the flow-shop version of this scheduling problem. In section 4.3, for this flow-shop problem we provide sufficient conditions for a particular permutation schedule to be an optimal solution. We prove in section 4.4 that this flow-shop problem is NP-complete. We develop in section 4.5 a branch-and-bound method to solve this flow-shop problem.

4.1 The Maximum Penalty Problem :

Consider the scheduling problem defined in section 1.0 of Chapter I along with the relevant notation. Recollect that F denotes the set of feasible schedules.

Under a feasible schedule σ, the start time and completion time of job j are denoted by S_j and C_j respectively. Associated with job j are its target start time a_j and its due date b_j such that $a_j < b_j$. The earliness E_j and tardiness T_j of job j under a feasible schedule σ are defined by

$$E_j = \max\{a_j - S_j, 0\}$$

$$\text{and} \quad T_j = \max\{C_j - b_j, 0\}.$$

Earliness and tardiness penalties for job j are given by $g(E_j)$ and $h(T_j)$ where g and h are monotonically non-decreasing continuous functions such that $g(0) = h(0) = 0$. For any feasible schedule σ the cost $f(\sigma)$ is given by

$$f(\sigma) = \max\{g(E_1), \ldots, g(E_n), h(T_1), \ldots, h(T_n)\}$$

$$= \max\{g(\max_{1 \le j \le n} E_j), h(\max_{1 \le j \le n} T_j)\} \quad \ldots \quad (4.1.1)$$

The maximum penalty problem is to find a schedule which minimizes the cost $f(\sigma)$ over the set of feasible schedules F.

The maximum tardiness problem is a modification of the maximum penalty problem obtained by treating the a_j's as strict release times of the jobs and taking as the objective the minimization of maximum tardiness. Symbolically the maximum tardiness problem is :

$$\text{minimize} \left\{ \max_{1 \le j \le n} T_j \right\}$$

over the set of feasible schedules F' where

$$F' = \left\{ \sigma \in F \mid \max_{1 \le j \le n} \{E_j\} = 0 \right\}.$$

Note that in the discussions of this section, we do not exclude the possibility of task splitting, simultaneous processing of one job by several machines, job overlap on a machine, etc. in a feasible schedule.

The following two theorems prove the reduction of a maximum penalty problem to a maximum tardiness problem under the translation assumption given below.

Translation assumption : If $\sigma \in F$ and y is a real number, then $(\sigma + y) \in F$, where $\sigma + y$ is obtained from σ by adding y to the lower and upper bounds of every interval describing the task schedules of every job.

This assumption excludes arrival times and due dates which cannot be violated, but permits arrival times and due dates which can be violated subject to penalty.

Using the properties of continuity and monotonicity of g and h it is easy to see that for any given non-negative Δ , the following system (4.1.1) is feasible.

$$\left.\begin{aligned} E^* + T^* &= \Delta \\ g(E^*) &= h(T^*) \\ E^* , T^* &\geq 0 \end{aligned}\right\} \quad \ldots \quad (4.1.1)$$

The system (4.1.1) has a unique solution if the assumption of non-decreasing monotonicity of g and h is replaced by strictly increasing monotonicity.

Theorem 4.1.1 : Suppose Δ is the optimal value of the objective function for the maximum tardiness problem. Then there exists a solution to the maximum penalty problem with the objective function value $g(E^*) = h(T^*)$ where E^* and T^* are a solution to (4.1.1) .

Proof : In an optimal schedule σ to the maximum tardiness problem, $E_j = 0$ and $T_j \leq \Delta$ for all jobs j, with $T_i = \Delta$ for at least one job (say) i. Construct a new schedule from σ by reducing the start time of every task (or in the case of job-splitting, every segment of every task) by E^*. In otherwords consider the schedule $\sigma - E^*$. Clearly, the maximum earliness in this new schedule $\sigma - E^*$ is no greater than E^*, and the maximum tardiness is precisely $\Delta - E^* = T^*$. Since the new schedule $\sigma - E^*$ is in F(by the translation assumption), the theorem follows immediately.//

Theorem 4.1.2 : A lower bound on the optimal value for the maximum penalty problem is given by $g(E^*)$ where (E^*, T^*) is a solution of (4.1.1) for Δ , the optimal value of the objective function for the maximum tardiness problem.

Proof : Suppose that a schedule σ for the maximum penalty problem exists with maximum penalty less than $g(E^*) = h(T^*)$. Let the maximum earliness and tardiness in this schedule σ be denoted by E_{max} and T_{max}; it follows that $E_{max} < E^*$ and $T_{max} < T^*$. Construct a new schedule by increasing the start time of every task (or in the case of job-splitting, every segment of every task) by E_{max}. In the resulting schedule $\sigma + E_{max}$ we have, $E_j = 0$ and $T_j \leq T_{max} + E_{max} < T^* + E^* = \Delta$ for all jobs j. Since this new schedule $\sigma + E_{max}$ is in F', it contradicts the optimality of Δ for the maximum tardiness problem. Hence Theorem 4.1.2 is proved. //

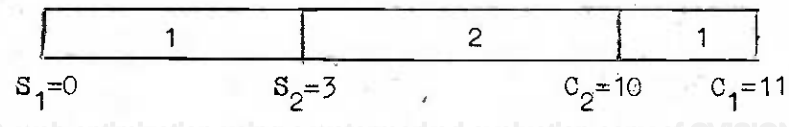
Remark 4.1.3 : Theorem 4.1.1 provides an algorithm for converting an optimal solution to the maximum tardiness problem into an optimal solution to the maximum penalty problem. The complexity of this algorithm is no greater than the maximum of the complexity of calculating E^* and the number of intervals in the optimal schedule.

The translation assumption is violated when all the jobs and machines are available only at time zero for scheduling. Under this situation, the results of Theorems 4.1.1 and 4.1.2 cannot be proved as shown by the following counter example. In this example, we assume that the machine cannot process two distinct jobs simultaneously.

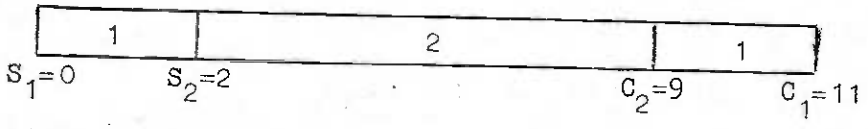
Example 4.1.1 : Single machine maximum penalty problem with $g(x) = x = h(x)$ for $x \geq 0$.

Job j	Processing time p_j	Target times a_j	b_j
1	4	0	9.5
2	7	3	8

The unique optimal solution (with $\Delta = 2$) of the corresponding single machine maximum tardiness problem with strict arrival times is presented in the following figure.



Application of Theorem 4.1.1 to this schedule does not produce an optimal solution to the maximum penalty problem as the following figure represents a better schedule σ with $f(\sigma) = 1.5$.



Application of Theorem 4.1.1 fails to produce an optimal solution to the maximum penalty problem, since shifting backwards the starting times by $E^* > 0$ is impossible for some jobs, when machines and jobs are not available earlier than time zero.

4.1.1 The Maximum Tardiness Problem :

In this subsection we provide a procedure to solve the maximum tardiness problem under certain specific assumptions. Using the notation of the scheduling problem introduced in section 1.0 of Chapter I, we define the set of feasible schedules F through the following assumptions :

B1 : There are no precedence constraints between the jobs in N, to be satisfied by a schedule.

B2 : Associated with task i in J_j the processing time is a non-negative real number denoted by p_{ij}.

B3 : There may be precedence constraints between the tasks in J_j. These precedence constraints are expressed through a precedence network $G_j = (\hat{J}_j , A_j)$ where

- $\hat{J}_j = J_j \cup \{0', 1'\}$ is the set of nodes such that 0' and 1' represent start and finish tasks respectively each with zero processing times.
- $(0', i)$ and $(i, 1')$ belong to A_j for every i in J_j.
- For i and k in J_j, (i,k) is in A_j if and only if the task k can be started only after the completion of the task i.
- There exists no cycle in G_j, i.e. there is no sequence of nodes $i_1, i_2, \ldots, i_k$ such that $k \geq 3$, $i_k = i_1$ and (i_s, i_{s+1}) is in A_j for $1 \leq s \leq k-1$.

B4 : Task splitting is permitted. In otherwords pre-emption resume is allowed.

B5 : Simultaneous processing of a task i in J_j by two or more distinct machines is not allowed.

B6 : Associated with job j in N, we are given target start time a_j and due date b_j such that $a_j < b_j$. All the tasks in J_j can be started only after a_j.

Recollect from section 1.0 of Chapter I that under a schedule σ, we denote the starting time and completion time of a task i in J_j by s_i and c_i respectively. Further the starting time and completion time of job j are denoted by S_j and C_j respectively. Thus a feasible schedule σ in F satisfies the following conditions :

- $S_j \geq a_j$ for every job j in N (by B6).
- Given the precedence network G_j, if $(i,k) \in A_j$ then $s_k \geq c_i$ (by B3).
- Let $\{(X_q^i, Y_q^i)\}$ be the finite collection of intervals describing the task schedule for task i in J_j. Then the intervals in this collection are disjoint intervals (by B5). Further, $p_{ij} = \sum_q (Y_q^i - X_q^i)$ (by B2, B4 and B5).

The maximum tardiness problem is to find a schedule σ^* that minimizes $T_{max}(\sigma) = \max_{1 \leq j \leq n} T_j$ over the set of feasible schedules F.

Note that in this subsection, we do not exclude the possibility of simultaneous processing of one job by several machines in a feasible schedule.

Given precedence network G_j, start target time a_j and due date b_j we associate with every task i in $\hat{J}_j$ target start time a_i^j and due date b_i^j as follows :

Define,

$$a_{0'}^j = a_j ,$$

$$a_i^j = \max\left\{a_k^j + p_{kj} \mid (k,i) \in A_j\right\} , \forall\, i \neq 0' \text{ and } i \in \hat{J}_j ,$$

$$b_{1'}^j = b_j ,$$

$$b_i^j = \min\left\{b_k^j - p_{kj} \mid (i,k) \in A_j\right\} , \forall\, i \neq 1' \text{ and } i \in \hat{J}_j .$$

Define a set of feasible schedules F' by

$$F' = \{ \sigma \in F \mid s_i \geq a_i^j \ \forall \ i \text{ and } j \text{ such that } i \in J_j \}.$$

Lemma 4.1.4 : Given any maximum tardiness problem we have

$$F' = F.$$

Proof : By the definition of F' we know that $F' \subseteq F$. Thus the lemma is proved if we show that for any $\sigma \in F$ we have $s_i \geq a_i^j$, $\forall$ i and j such that $i \in J_j$. For a fixed j, from the precedence network G_j note that $a_i^j \geq a_j \ \forall \ i \in J_j$ and there exists at least one task i* in J_j such that $a_{i*}^j = a_j$. Now let σ be an arbitrary schedule from F. Then $S_j = \min_{i \in J_j} s_i \geq a_j$. Therefore we have $s_{i*} \geq a_{i*}^j$ for all i* such that $a_{i*}^j = a_j$. This provides a basis for induction. Make the induction hypothesis that $s_k \geq a_k^j$ for every predecessor k of i in the network G_j. Now we will prove the hypothesis for $i \in J_j$. For every $k \in J_j$ such that $(k,i) \in A_j$ we have,

$$\begin{aligned} s_i &\geq c_k && \text{(by B3)} \\ &\geq s_k + p_{kj} && \text{(by B2, B4 and B5)} \\ &\geq a_k^j + p_{kj} && \text{(by the induction hypothesis)} \end{aligned}$$

Now invoking the definition of a_i^j we get $s_i \geq a_i^j$. This completes the proof of Lemma 4.1.4. //

Under any schedule $\sigma \in F$, the tardiness of task i in J_j is given by $T_i^j = \max \{ c_i - b_i^j , 0 \}$ and the tardiness of job j is given by $T_j = \max \{ C_j - b_j , 0 \}$ where $C_j = \max_{i \in J_j} c_i$.

Lemma 4.1.5 : For an arbitrary schedule σ in F,

$$T_{max}(\sigma) = \max_{1 \le j \le n} T_j = \max_{1 \le j \le n} \left\{ \max_{i \in J_j} T_i^j \right\} .$$

Proof : The proof of this lemma immediately follows, if we prove that $T_j = \max_{i \in J_j} T_i^j$ for every j, $j \in N$. For a fixed j, let $D = \left\{ i \in J_j \mid b_i^j = b_j \right\}$. From the precedence network G_j we note that $D \neq \emptyset$. If $D = J_j$ we have nothing to prove. Further, for every $i \in J_j$ and $i \notin D$ there exists a sequence of nodes $i_1, i_2, \ldots, i_l$ in G_j such that $l \ge 2$; $i_1 = i$; $i_l \in D$ and $(i_r, i_{r+1}) \in A_j$, $b_{i_r}^j = b_{i_{r+1}}^j - p_{i_{r+1}, j}$ for every r, $1 \le r \le l-1$. Now using the fact that $(i_r, i_{r+1}) \in A_j$ we have

$$c_{i_r} \le s_{i_{r+1}} \le c_{i_{r+1}} - p_{i_{r+1}, j}$$

and hence, $T_{i_r}^j \le T_{i_{r+1}}^j$ for every r, $1 \le r \le l-1$.

Thus we get $\max_{i \in J_j} T_i^j = \max_{i \in D} T_i^j$ . ... (4.1.2)

Again from the precedence network G_j note that for every $i \in J_j$ and $i \notin D$ we have $c_i \leq c_k$ for some $k \notin D$. Therefore

$$C_j = \max_{i \in J_j} c_i = \max_{i \in D} c_i . \quad \ldots \quad (4.1.3)$$

Now from (4.1.2), (4.1.3) and the definition of the set D we get

$$T_j = \max_{i \in J_j} T_i^j .$$

This proves Lemma 4.1.5. //

In the following we outline a procedure to solve the maximum tardiness problem. Now we replace assumption B5 by a more stringent assumption B5' : There is exactly one machine out of the m machines, which can perform the task i in J_j , $j \in N$. In otherwards the set of all tasks can be partitioned into m sets corresponding to the m machines which can perform them.

We make one more assumption B7 : No two distinct tasks can be simultaneously processed on machine M_q , $1 \leq q \leq m$. Let the collection of all tasks be denoted by $B = \{(i,j) \mid i \in J_j, j \in N\}$. Suppose there are α distinct target start times a_i^j associated with the tasks in B. Let these α distinct target start times be arranged as $\partial_1 < \partial_2 < \ldots < \partial_\alpha$. In the construction of an optimal schedule, we review the schedule at time points $\partial_1, \ldots, \partial_\alpha$ and at the completion times of the tasks

in B which are already scheduled. At the time of reviewing we will be fixing some of the intervals of the task schedules of the tasks in B.

Let a typical time of reviewing be denoted by θ. We introduce the following notation with reference to the time point θ.

- $p_{ij}(\theta)$ is the balance processing time associated with task (i,j) at time point θ. Note that $p_{ij} - p_{ij}(\theta)$ gives the portion of the processing time of task (i,j) completed before time point θ.

- $F(\theta) = \left\{ (i,j) \in B \mid p_{ij}(\theta) = 0 \right\}$ is the set of tasks which are completed on or before time point θ.

- $D(\theta) = \left\{ (i,j) \in B - F(\theta) \mid a_i^j \leq \theta \text{ and } (k,j) \in F(\theta) \text{ if } (k,i) \in A_j \right\}$ gives the set of uncompleted tasks available before time θ such that all its preceding tasks are completed.

- $\bar{D}(\theta) = \left\{ (i,j) \mid b_i^j = \min\left\{ b_k^l \mid (k,l) \in D(\theta) \right\} \right\}$ gives the subset of $D(\theta)$, which has tasks with least due date.

- $\hat{\theta} = \min\left\{ \theta_i \mid \theta_i > \theta, \quad 1 \leq i \leq \alpha \right\}$ is the next possible review time among the various θ_i, $1 \leq i \leq \alpha$.

At the initial stage of the procedure set $\theta = \theta_1$ and $p_{ij}(\theta_1) = p_{ij}$ for every $i \in J_j$ and $j \in N$. At the review time θ if $F(\theta) = B$ then we have scheduled all the tasks and hence we stop. Otherwise,

at the review time ∂, we schedule the tasks by the following rule :

(i) Schedule as many tasks as possible from $\bar{D}(\partial)$ on the appropriate machines.

(ii) If all the tasks in $\bar{D}(\partial)$ are scheduled then redefine $\bar{D}(\partial)$ with the tasks in $D(\partial) - \bar{D}(\partial)$.

Repeat the steps (i) and (ii) until either no more jobs can be scheduled or all jobs in $D(\partial)$ are scheduled.

Let $S(\partial)$ be a subset of $D(\partial)$ such that the tasks in $S(\partial)$ are scheduled on the appropriate machines at time point ∂.

Now fix the next time point of review as $\partial^* = \min\{\hat{\partial}, \bar{\partial}\}$ where

$$\bar{\partial} = \min\{ p_{ij}(\partial) \mid (i,j) \in S(\partial) \}.$$

Then for each (i,j) in $S(\partial)$ we include the time interval $(X_i^q = \partial, Y_i^q = \partial^*)$ on the appropriate machine M_q in the task schedule of task (i,j). Further we update the balance processing times of the tasks, at time point ∂^* by

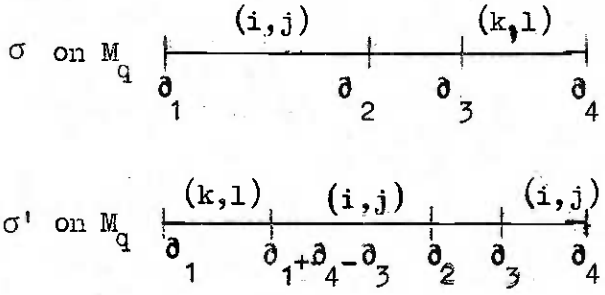
$$p_{ij}(\partial^*) = \begin{cases} p_{ij}(\partial) - (\partial^* - \partial), & (i,j) \in S(\partial) \\ p_{ij}(\partial), & (i,j) \notin S(\partial). \end{cases}$$

The above procedure is repeated until all the tasks in B are completed.

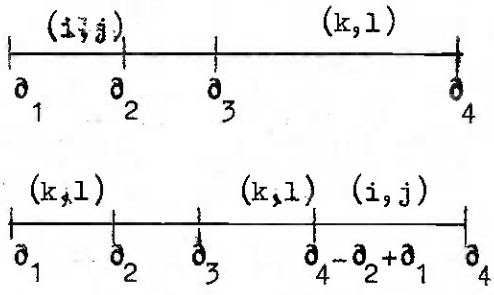
Theorem 4.1.6 : A schedule σ^* constructed by the procedure just described solves the maximum tardiness problem.

Proof : Using Lemma 4.1.4 and the definitions of a_i^j and b_i^j it is easy to see that the schedule σ^* is in F, the set of feasible schedules. From the construction of σ^* observe that at any point of time δ, on machine M_q we schedule the task with least due date from among those which can be processed at time δ on that machine. We show in the following argument that any arbitrary schedule σ violating this property need not be considered, while searching for an optimal solution of the maximum tardiness problem. If σ is an arbitrary schedule violating this property, then there exist tasks (i,j) and (k,l) and machine M_q such that under the schedule σ we have (i) $(X_q^i = \delta_1 , Y_q^i = \delta_2)$ and $(X_q^k = \delta_3 , Y_q^k = \delta_4)$ are intervals during which tasks (i,j) and (k,l) respectively are processed on machine M_q (ii) $\delta_1 < \delta_3$ (iii) $b_k^l < b_i^j$ and (iv) both tasks (i,j) and (k,l) are available at time δ_1 for processing them on machine M_q. First observe that $\delta_2 \leq \delta_3$, since the machine M_q can handle atmost one task at a time. Now define a new schedule σ' from σ by modifying the intervals (δ_1, δ_2) and (δ_3, δ_4) corresponding to the tasks (i,j) and (k,l) on machine M_q as explained in the following figure.

Case(i) $\partial_4 - \partial_3 \leq \partial_2 - \partial_1$



Case(ii) $\partial_4 - \partial_3 > \partial_2 - \partial_1$



Now using Lemma 4.1.5 it is easy to see that $T_{max}(\sigma') \leq T_{max}(\sigma)$ since σ' is same as σ except for the above change in task schedules of tasks (i,j) and (k,l). Thus, starting from any arbitrary schedule σ repeating the above arguments as many times as required, we can reach a schedule σ^* that can be generated by the procedure.

Using similar arguments with $b_k^l = b_i^j$, we can show that any two distinct schedules, generated by the procedure, have the same objective function value. This proves Theorem 4.1.6. //

Remark 4.1.7 : It should be noted that when m=1 and $|J_j| = 1 \; \forall \; J \in N$, the suggested procedure reduces to the procedure of solving the single machine maximum tardiness problem discussed in page 82 of the book by Baker (1974).

4.2 The Maximum Penalty Flow-Shop Problem :

The maximum penalty flow-shop (MPF-S) problem is the maximum penalty problem with the flow-shop assumptions A1 to A8 inclusive of

A4' and A5' given in section 1.2 of Chapter I. Given a MPF-S problem, there are two related problems viz.

(i) maximum tardiness strict job release time flow-shop (MTSJRF-S) problem obtained by relaxing assumption A5' and taking $r_j = a_j$, $\forall\ j \in N$ in the assumption A5.

(ii) maximum tardiness flow-shop (MTF-S) problem obtained by ignoring the a_j's.

In view of the assumptions A4' and A5' and the Example 4.1.1., note that the MPF-S problem cannot be solved by solving the related MTSJRF-S problem.

Henceforth, we discuss the maximum penalty flow-shop problem in greater detail.

Let (S,Q) be a feasible schedule (Definition 1.2.1.). Then the earliness and tardiness of job j will be given by

$$E_j = \max\left\{ a_j - s_{j1} , 0 \right\}$$

$$\text{and} \quad T_j = \max\left\{ c_{jm} - b_j , 0 \right\}.$$

The maximum penalty associated with a feasible schedule (S,Q) is given by

$$f((S,Q)) = \max\left\{ g\left(\max_{1 \le j \le n} E_j \right) , h\left(\max_{1 \le j \le n} T_j \right) \right\}.$$

Remark 4.2.1 : Consider a MPF-S problem. If (S,Q) and (S',Q') are two feasible schedules such that

$$\max_{1 \le j \le n} T_j \le \max_{1 \le j \le n} T'_j \quad \text{and} \quad \max_{1 \le j \le n} E_j \le \max_{1 \le j \le n} E'_j$$

then $f((S,Q)) \le f((S',Q'))$.

Proof : Using the properties of penalty functions g and h the proof of Remark 4.2.1 follows immediately. //

It is easy to verify that f is a regular measure of performance if $g(x) = 0$ for every $x \ge 0$. Conway et al. (1967) have shown that, while solving a two machine flow-shop problem for a regular measure of performance, it is enough to consider the permutation schedules alone. Now a natural question arises : can we consider the permutation schedules alone while solving a two machine MPF-S problem? In the following we prove results related to this question.

Lemma 4.2.2 : Consider a MPF-S problem. Let (S',Q') be a feasible schedule such that jobs i and j with $a_j \le a_i$ ($b_j \le b_i$) appear on machines M_1 and M_2 (M_{m-1} and M_m) as explained in Figure 4.2.1 (4.2.3). Construct a schedule (S,Q) from (S',Q') where i and j are interchanged on machine M_1 (M_m) as explained in Figure 4.2.2 (4.2.4). Then

$$f((S,Q)) \le f((S', Q')).$$

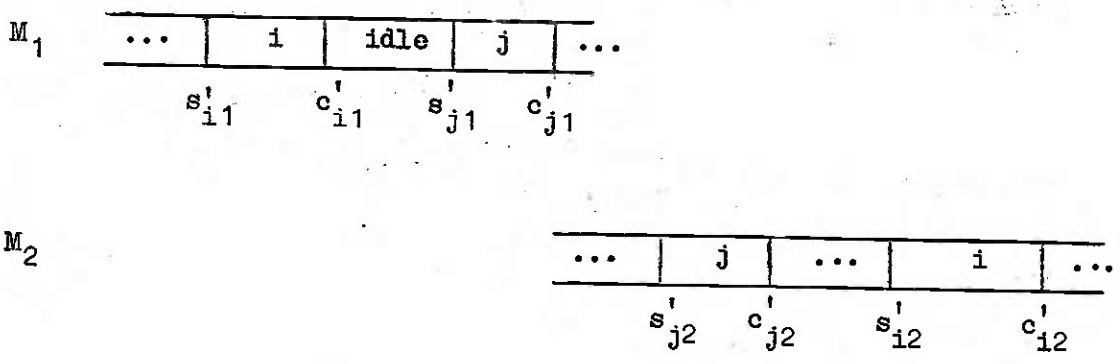


Figure 4.2.1

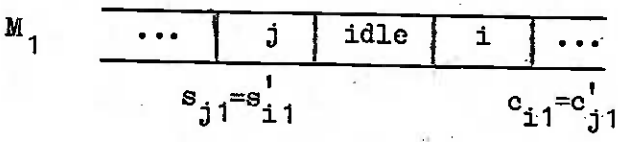
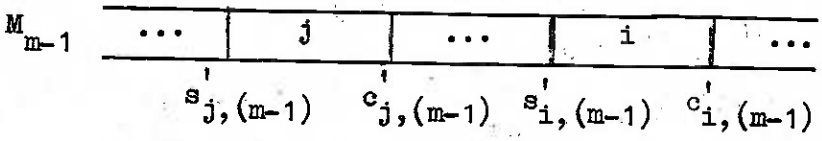


Figure 4.2.2



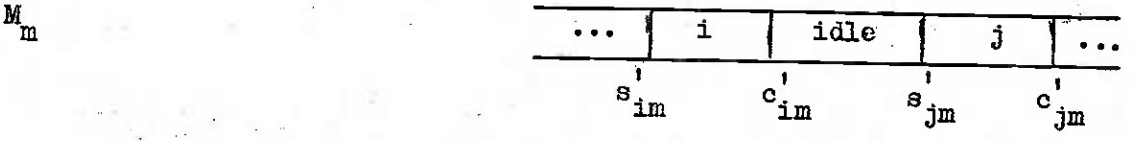


Figure 4.2.3

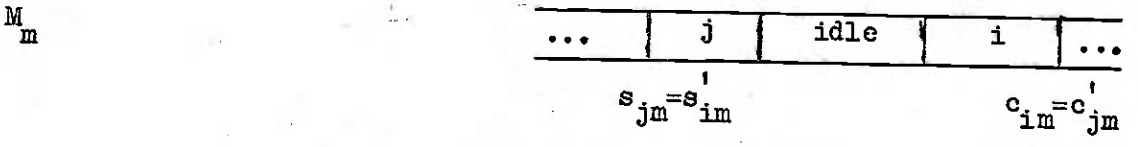


Figure 4.2.4

Proof : The result follows easily from Remark 4.2.1, if we notice that $s_{j1} = s'_{i1}$ and $c_{i1} = c'_{j1}$ from Figure 4.2.2 in the case $a_j \leq a_i$; $s_{jm} = s'_{im}$ and $c_{im} = c'_{jm}$ from Figure 4.2.4 in the case $b_j \leq b_i$ and in both the cases all other jobs remain undisturbed while defining (S,Q) from (S',Q').//

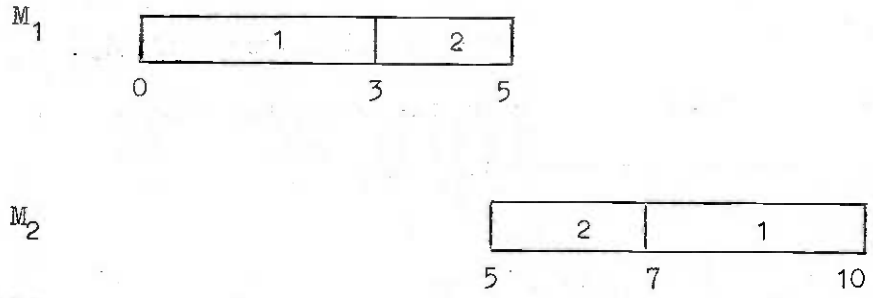
In Lemma 4.2.2 the case $a_j \leq a_i$ ($b_j \leq b_i$) is similar to Theorem 5.1 (5.2) proved for any regular measure of performance (F_{max}), in the book by Conway et al. (1967). It is easy to see that Lemma 4.2.2 holds for the MTSJRF-S problem as well.

When $|N| = 2$, using Lemma 4.2.2 one can easily note that it is enough to consider the permutation schedules alone, while solving a two machine MPF-S problem with the property that "$a_i < a_j \Longrightarrow b_i \leq b_j$". The following Example 4.2.1 shows that it is not enough to consider the permutation schedules alone when this property is relaxed.

Example 4.2.1 : Two machine MPF-S problem with $g(x) = x = h(x)$ for $x \geq 0$.

Job	Processing times		Target times	
i	p_{i1}	p_{i2}	a_i	b_i
1	3	3	0	10
2	2	2	3	7

The unique optimal schedule with objective function value as zero is sketched in the following figure.



After proving some relevant results, we provide an example of three jobs-two machines MPF-S problem to show that even with the property "$a_i < a_j \Longrightarrow b_i \leq b_j$", it is not enough to consider the permutation schedules alone when the number of jobs is greater than or equal to three.

Now, for a MPF-S problem we shall develop a method to get the best permutation schedule for a given permutation $\pi = (\pi(1), \ldots, \pi(n))$. Given a MPF-S problem we note the following for the related MTSJRF-S and MTF-S problems.

In a MTSJRF-S problem, for any arbitrary permutation $\pi = (\pi(1), \ldots, \pi(n))$ the corresponding best permutation schedule is provided by the non-delay permutation schedule consistent with π (see Definition 1.2.3). Using Notation 1.3.9, in a MTSJRF-S problem, under a non-delay permutation schedule π, the tardiness of job $\pi(j)$ is given by

$$\bar{T}_{\pi(j)} = \max \left\{ t(0;\ a;\ \pi(1),\ \ldots,\ \pi(j);\ m) - b_{\pi(j)},\ 0 \right\}$$

where $0 = (0,\ \ldots,\ 0)$ and $a = (a_1,\ \ldots,\ a_n)$ are tuples giving the machine available and job release times respectively. Given the non-delay permutation schedule consistent with π the corresponding objective function value in the MTSJRF-S problem is denoted by

$$\Delta(\pi) = \max_{1 \leq j \leq n} \bar{T}_{\pi(j)}. \qquad \ldots \quad (4.2.1)$$

In a MTF-S problem, for any permutation $\pi = (\pi(1),\ \ldots,\ \pi(n))$ the corresponding best permutation schedule is provided by the non-delay permutation schedule consistent with π and the corresponding objective function value is given by

$$\hat{T}(\pi) = \max_{1 \leq j \leq n} \hat{T}_{\pi(j)} \qquad \ldots \quad (4.2.2)$$

where $\hat{T}_{\pi(j)}$ is the tardiness of job $\pi(j)$ and using Notation 1.3.7, we have, $\hat{T}_{\pi(j)} = \max \left\{ t(\pi(1),\ \ldots,\ \pi(j);\ m) - b_{\pi(j)},\ 0 \right\}$.

In a MPF-S problem, given any permutation $\pi = (\pi(1),\ \ldots,\ \pi(n))$ and a corresponding arbitrary permutation schedule $S(\pi)$ we note the following. For any i and j such that $i \leq j$ we have, $E_{\pi(i)} \geq a_{\pi(i)} - s_{\pi(i),1}$ and $T_{\pi(j)} \geq c_{\pi(j),m} - b_{\pi(j)}$. Adding these two inequalities we get,

$$E_{\pi(i)} + T_{\pi(j)} \geq a_{\pi(i)} - b_{\pi(j)} + c_{\pi(j),m} - s_{\pi(i),1} . \quad \ldots (4.2.3)$$

From Definitions 1.2.2 and 1.2.3 for $i \leq j$, we see that $c_{\pi(j),m} - s_{\pi(i),1} \geq$ the completion time of the last job in $(\pi(i), \ldots, \pi(j))$ on machine M_m under the non-delay permutation schedule $(\pi(i), \ldots, \pi(j))$, i.e.,

$$c_{\pi(j),m} - s_{\pi(i),1} \geq t(\pi(i), \ldots, \pi(j); m). \quad \ldots (4.2.4)$$

Using (4.2.4) in (4.2.3) for any i and j such that $i \leq j$ we have,

$$E_{\pi(i)} + T_{\pi(j)} \geq a_{\pi(i)} + t(\pi(i), \ldots, \pi(j); m) - b_{\pi(j)} . \quad \ldots (4.2.5)$$

Further, using the critical path network of π (Figure 1.3.1) we can show that

$$t(0; a; \pi(1), \ldots, \pi(j); m) = \max_{1 \leq i \leq j} \left\{ a_{\pi(i)} + t(\pi(i), \ldots, \pi(j); m) \right\} .$$

$$\ldots (4.2.6)$$

Now using (4.2.1), (4.2.5) and (4.2.6) it is easy to see that there exist i^* and j^* such that $i^* \leq j^*$ and

$$E_{\pi(i^*)} + T_{\pi(j^*)} \geq \Delta(\pi) . \quad \ldots (4.2.7)$$

Let E^*_π and T^*_π be non-negative numbers such that

$$\left.\begin{aligned} E^*_\pi + T^*_\pi &= \Delta(\pi) \\ \text{and} \quad g(E^*) &= h(T^*) . \end{aligned}\right\} \quad \ldots (4.2.8)$$

Lemma 4.2.3 : In a MPF-S problem, for every permutation schedule $S(\pi)$ with job order $\pi = (\pi(1), \ldots, \pi(n))$ we have,

$$f(S(\pi)) \geq \max \left\{ g(E^*_\pi), \; h(\hat{T}(\pi)) \right\} \qquad \ldots \quad (4.2.9)$$

where E^*_π is given by (4.2.8).

Proof : For any permutation schedule $S(\pi)$ with job order π, from (4.2.7) and (4.2.8) we have, for some $i^* \leq j^*$,

$$E_{\pi(i^*)} + T_{\pi(j^*)} \geq \Delta(\pi) = E^*_\pi + T^*_\pi .$$

Thus it must be true that either $E_{\pi(i^*)} \geq E^*_\pi$ or $T_{\pi(j^*)} \geq T^*_\pi$. Hence, either $g(E_{\pi(i^*)}) \geq g(E^*_\pi)$ or $h(T_{\pi(j^*)}) \geq h(T^*_\pi)$. But we know from (4.2.8) that $g(E^*_\pi) = h(T^*_\pi)$ and therefore we get

$$g(E^*_\pi) \leq \max \left\{ g(E_{\pi(i^*)}), \; h(T_{\pi(j^*)}) \right\} \leq f(S(\pi)). \qquad \ldots \quad (4.2.10)$$

In (4.2.4) setting $i=1$ and using $s_{\pi(1),1} \geq 0$ we get

$$c_{\pi(j),m} \geq t(\pi(1), \ldots, \pi(j); m).$$

Therefore, $T_{\pi(j)} \geq \hat{T}_{\pi(j)}$, $1 \leq j \leq n$, and hence

$$h(\hat{T}(\pi)) \leq h\Big(\max_{1 \leq j \leq n} T_{\pi(j)} \Big) \leq f(S(\pi)). \qquad \ldots \quad (4.2.11)$$

Now combining (4.2.10) and (4.2.11), Lemma 4.2.3 is proved. //

Remark 4.2.4 : Lemma 4.2.3 holds with reference to a partial permutation $\pi = (\pi(1), \ldots, \pi(q))$ and a corresponding arbitrary partial permutation schedule $S(\pi)$ when $\hat{T}(\pi)$ and $\Delta(\pi)$ are defined just with the help of the partial permutation π.

Henceforth, in a MPF-S problem, for a given permutation $\pi = (\pi(1), \ldots, \pi(n))$ we consider only the non-delay permutation schedule consistent with π where the machine available and job release times are given by the tuples $0 = (0, \ldots, 0)$ and $r = (r_1, \ldots, r_n)$ respectively with $r_j = \max(a_j - E^*_\pi, 0)$, $1 \leq j \leq n$, where E^*_π is given by (4.2.8). This non-delay permutation schedule will be denoted by $S(\pi)$ without any ambiguity.

Theorem 4.2.5 : In a MPF-S problem, for any given permutation $\pi = (\pi(1), \ldots, \pi(n))$, the non-delay permutation schedule $S(\pi)$ consistent with the job release times $r_j = \max(a_j - E^*_\pi, 0)$, $1 \leq j \leq n$, is one of the best among the permutation schedules consistent with the job order π. That is

$$f(S(\pi)) = \max\left\{g(E^*_\pi), h(\hat{T}(\pi))\right\}.$$

Proof : From the properties of $S(\pi)$ we have

$$S_{\pi(j)} = s_{\pi(j),1} \geq r_{\pi(j)} \geq a_{\pi(j)} - E^*_\pi \quad \text{for every } j$$

which implies that $E_{\pi(j)} \leq E^*_\pi$ for every j and hence we get

$$g\Big(\max_{1 \le j \le n} E_{\pi(j)}\Big) \le g(E^*_\pi) \le \max\Big\{g(E^*_\pi),\ h(\hat{T}(\pi))\Big\}. \quad \ldots \quad (4.2.12)$$

Now for the schedule $S(\pi)$, using (4.2.6) we have for $1 \le j \le n$,

$$C_{\pi(j)} = t(0;\ r;\ \pi(1),\ \ldots,\ \pi(j);\ m)$$

$$= \max_{1 \le i \le j} \Big\{ r_{\pi(i)} + t(\pi(i),\ \ldots,\ \pi(j);\ m) \Big\}$$

say

$$= r_{\pi(u)} + t(\pi(u),\ \ldots, \pi(j);\ m). \quad \ldots \quad (4.2.13)$$

Case (i). $r_{\pi(u)} = a_{\pi(u)} - E^*_\pi$. Now from (4.2.13) we get

$$C_{\pi(j)} = a_{\pi(u)} - E^*_\pi + t(\pi(u),\ \ldots,\ \pi(j);\ m)$$

$$\le t(0;\ a;\ \pi(1),\ \ldots,\ \pi(j);\ m) - E^*_\pi, \text{ using (4.2.6).}$$

Thus

$$C_{\pi(j)} - b_{\pi(j)} \le t(0;\ a;\ \pi(1),\ \ldots,\ \pi(j);\ m) - b_{\pi(j)} - E^*_\pi$$

$$\le \bar{T}_{\pi(j)} - E^*_\pi, \quad \text{using definition of } \bar{T}_{\pi(j)},$$

$$\le \Delta(\pi) - E^*_\pi, \quad \text{using (4.2.1).}$$

Now using the fact (4.2.8) that $T^*_\pi = \Delta(\pi) - E^*_\pi \ge 0$, we get $T_{\pi(j)} \le T^*_\pi$. Thus under case (i) we have (once again using (4.2.8))

$$h(T_{\pi(j)}) \le h(T^*_\pi) = g(E^*_\pi) \le \max\Big\{g(E^*_\pi);\ h(\hat{T}(\pi))\Big\}. \quad \ldots \quad (4.2.14)$$

Case (ii). $r_{\pi(u)} = 0$. Then from (4.2.13) we have

$$C_{\pi(j)} = t(\pi(u), \ldots, \pi(j); m)$$

$$\leq t(\pi(1), \ldots, \pi(j); m), \text{ from the definition of } t(\pi; m).$$

But we know that $C_{\pi(j)} \geq t(\pi(1), \ldots, \pi(j); m)$ and therefore

$$C_{\pi(j)} = t(\pi(1), \ldots, \pi(j); m).$$

Now we get $T_{\pi(j)} = \hat{T}_{\pi(j)}$ and hence

$$h(T_{\pi(j)}) = h(\hat{T}_{\pi(j)}) \leq \max\left\{g(E^*_\pi), h(\hat{T}(\pi))\right\}. \quad \ldots \quad (4.2.15)$$

Combining cases (i) and (ii) through (4.2.14) and (4.2.15) we get

$$h\left(\max_{1 \leq j \leq n} T_{\pi(j)}\right) \leq \max\left\{g(E^*_\pi), h(\hat{T}(\pi))\right\}. \quad \ldots \quad (4.2.16)$$

Now (4.2.12), (4.2.16) and Lemma 4.2.3 together give us

$$f(S(\pi)) = \max\left\{g(E^*_\pi), h(\hat{T}(\pi))\right\}.$$

This proves Theorem 4.2.5. //

Remark 4.2.6 : In a MPF-S problem, for a given permutation π, the construction of $S(\pi)$ can be done with computational complexity no greater than the maximum of the complexity of computing $\Delta(\pi)$, $\hat{T}(\pi)$ and E^*_π.

The following Example 4.2.2 shows that while solving a MPF-S problem it is not enough to consider the permutation schedules alone for $n \geq 3$ even if the property "$a_i < a_j \Longrightarrow b_i \leq b_j$" holds.

Example 4.2.2 : Three job - two machine MPF-S problem with $g(x) = x = h(x)$ for $x \geq 0$

Job i	Processing times		Target times	
	p_{i1}	p_{i2}	a_i	b_i
1	12	2	0	21.75
2	3	20	3	38
3	4	8	6	39

For various permutations π one can get the corresponding non-delay permutation schedules $S(\pi)$ and note the following.

(i) The permutation schedule $S(\pi^*) = \begin{pmatrix} 0 & 3 \\ 3 & 23 \\ 15 & 25 \end{pmatrix}$ where

$\pi^* = (2,1,3)$ has $f(S(\pi^*)) = 3.25$ and

$f(S(\pi^*)) < f(S(\pi))$ for every permutation $\pi \neq \pi^*$.

(ii) The feasible schedule $S' = \begin{pmatrix} 7 & 19 \\ 0 & 21 \\ 3 & 7 \end{pmatrix}$ with job order matrix $Q' = \begin{pmatrix} 2 & 3 & 1 \\ 3 & 1 & 2 \end{pmatrix}$ has $f((S',Q')) = 3$ and

$f((S',Q')) < f(S(\pi^*))$. This shows that while solving a MPF-S

problem for $n \geq 3$, it is not enough to consider the permutation schedules alone, even if the property "$a_i < a_j \Longrightarrow b_i \leq b_j$" holds.

Henceforth we make the assumption A9 (of section 1.2 of Chapter I) as well. Therefore we will consider only the permutation schedules while solving the MPF-S problem. Following the notation of Conway et al. (1967) we will denote the MPF-S, MTSJRF-S and MTF-S problems with assumption A9 by (n/m/F/f), $(n/m/F/a_j \geq 0 / T_{max})$ and $(n/m/F/T_{max})$ respectively.

Sidney (1977) introduced the (n/1/./f) problem and developed an $O(n^2)$ optimal algorithm assuming that "$a_i < a_j \Longrightarrow b_i \leq b_j$" for any pair of jobs i and j. Lakshmanan et al. (1978) stream-lined Sidney's algorithm into an $O(n \log n)$ algorithm for the (n/1/./f) problem.

Using the Example 4.2.2 we make the following interesting observations.

(1) The permutation schedule $S(\bar{\pi}) = \begin{pmatrix} 0 & 12 \\ 12 & 15 \\ 19 & 35 \end{pmatrix}$ where $\bar{\pi} = (1,2,3)$ has $f(S(\bar{\pi})) = 4$. Note that the permutation $\bar{\pi}$ is the non-decreasing arrangement of a_j's and b_j's. Further, $f(S(\pi^*)) < f(S(\bar{\pi}))$. Hence there does not exist an optimal permutation schedule (among the permutation schedules alone) that has the

property that jobs are ordered simultaneously by non-decreasing a_j and by non-decreasing b_j. This shows that Theorem 1 in page 64 of Sidney (1977), which is true for a (n/1/./f) problem, does not hold for a (n/2/F/f) problem.

(2) For a related problem $(3/2/F/a_j \geq 0/T_{max})$, from among the permutation schedules, the optimal solution is unique and it is given by the permutation $\bar{\pi} = (1,2,3)$ with $\Delta(\bar{\pi}) = 4$. For the other related problem $(3/2/F/T_{max})$ also, the optimal permutation schedule is unique and it is given by the permutation $\alpha = (3,1,2)$ with $\hat{T}(\alpha) = 1$. Now observe that

$$f(S(\pi^*)) < \min\{f(S(\bar{\pi})), f(S(\alpha))\}.$$

This shows that neither of the optimal permutation schedules of the related problems gives rise to an optimal permutation schedule of the (n/2/F/f) problem.

4.3 Sufficient Conditions for an Optimal Solution for the (n/m/F/f) Problem :

Recollect that we are denoting the MPF-S problem with the additional assumption A9 by (n/m/F/f) problem. In this section we provide sufficient conditions for a particular permutation π to yield an optimal solution for the (n/m/F/f) problem.

Let Figure 4.3.1 be the critical path network of (j,1), where the n-tuple $a = (a_1, \ldots, a_n)$ and m-tuple $0 = (0, \ldots, 0)$ give the job release and machine available times.

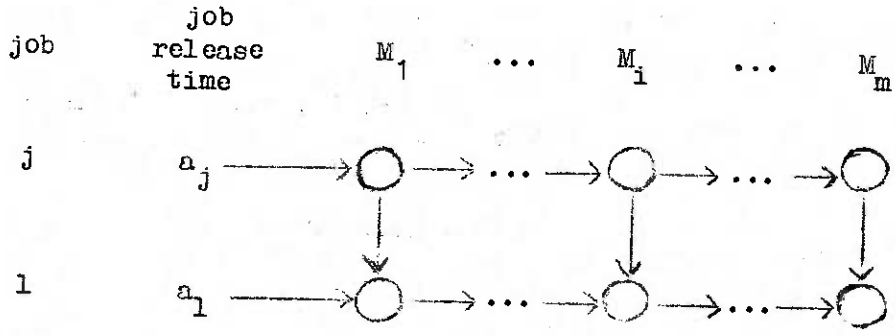


Figure 4.3.1 : Critical path network of (j,l)

From this critical path network we can easily see that for $1 \leq k \leq m$ we have,

$$t(0;a;jl;k) = \max \left\{ a_j + t(jl;k),\ a_l + \sum_{u=1}^{k} p_{lu} \right\}. \quad \ldots \quad (4.3.1)$$

Similarly for $1 \leq k \leq m$ we get

$$t(0;a;lj;k) = \max \left\{ a_l + t(lj;\ k),\ a_j + \sum_{u=1}^{k} p_{ju} \right\}. \quad \ldots \quad (4.3.2)$$

The following Lemma 4.3.1 is a generalisation of Lemma 3.1.3.

Lemma 4.3.1 : Let $\partial = (\partial_1, \ldots, \partial_m)$ and $a = (a_1, \ldots, a_n)$ give the machine available and job release times respectively. For a given pair of jobs j and l, and for any pair of machines M_r and M_k such that $1 \leq r < k \leq m$, let

$$\min (p_{jr} , p_{lk}) \leq \min (p_{lr} , p_{jk}).$$

Further assume that $a_j \leq a_1$.

Then, $t(\partial; a; jl; k) \leq t(\partial; a; lj; k)$, $1 \leq k \leq m$.

Proof : From the corresponding critical path networks note that for $1 \leq k \leq m$ we have,

$$t(\partial; a; jl; k) = \max\left\{ t(\partial; jl; k),\ t(0; a; jl; k)\right\} \quad \ldots (4.3.3)$$

and

$$t(\partial; a; lj; k) = \max\left\{ t(\partial; lj; k),\ t(0; a; lj; k)\right\} . \quad \ldots (4.3.4)$$

Now invoking Lemma 3.1.3 when $\partial = 0$ we get $t(jl;k) \leq t(lj;k)$. Using this along with the facts $a_j \leq a_1$ and $a_1 + \sum_{u=1}^{k} p_{lu} \leq a_1 + t(lj;k)$ in (4.3.1) and (4.3.2) we get

$$t(0; a; jl; k) \leq t(0; a; lj; k).$$

Once again applying Lemma 3.1.3 for any ∂ we have $t(\partial; jl; k) \leq t(\partial; lj; k)$. Now using these in (4.3.3) and (4.3.4) the proof of Lemma 4.3.1 follows. //

The following Theorem 4.3.2 is a generalisation of Theorem 3.1.4.

Theorem 4.3.2 : Let $\partial = (\partial_1, \ldots, \partial_m)$ and $a = (a_1, \ldots, a_n)$ give the machine available and job release times respectively. For a given

pair of jobs j and l, and for every pair of machines M_r and M_k such that $1 \leq r < k \leq m$, let

$$\min (p_{jr}, p_{lk}) \leq \min (p_{lr} , p_{jk}).$$

Further assume that $a_j \leq a_l$. Let γ be any partial permutation of jobs in $N - \{j, l\}$.

Then, $t(\partial; a; jl\gamma; k) \leq t(\partial; a; lj\gamma; k), \quad 1 \leq k \leq m.$

Proof : Define $\partial^* = (\partial^*_1 , \ldots, \partial^*_m)$ and $\hat{\partial} = (\hat{\partial}_1, \ldots, \hat{\partial}_m)$

by $\partial^*_k = t(\partial; a; jl; k), \quad 1 \leq k \leq m,$

and $\hat{\partial}_k = t(\partial; a; lj; k), \quad 1 \leq k \leq m.$

Using Lemma 4.3.1 we get $\partial^*_k \leq \hat{\partial}_k , \quad 1 \leq k \leq m.$ From the critical path networks of $jl\gamma$ and $lj\gamma$ we can write for $1 \leq k \leq m$,

$$t(\partial; a; jl\gamma ; k) = t(\partial^* ; a; \gamma ; k) \quad \ldots \quad (4.3.5)$$

and

$$t(\partial; a; lj\gamma ; k) = t(\hat{\partial}; a;\gamma ; k). \quad \ldots \quad (4.3.6)$$

Now using the fact $\partial^*_i \leq \hat{\partial}_i , \quad 1 \leq i \leq m$ in (4.3.5) and (4.3.6) the proof of Theorem 4.3.2 follows. //

Theorem 4.3.3 : Suppose $\pi = (\pi(1), \ldots, \pi (n))$ is a permutation such that $a_{\pi(1)} \leq \cdots \leq a_{\pi(n)}$ and $b_{\pi(1)} \leq \cdots \leq b_{\pi(n)}$. Further assume

that π is a Johnson's permutation for the two machine flow-shop problems (M_u, M_v), $1 \leq u < v \leq m$. Then the non-delay permutation schedule consistent with π and strict job release times is an optimal permutation schedule for the $(n/m/F/a_j \geq 0/T_{max})$ problem.

Proof : Assume to the contrary, that a non-delay permutation schedule $\sigma l j \theta$ minimizes T_{max} (in the $(n/m/F/a_j \geq 0/T_{max})$ problem) where j precedes l in π. Note that $a_j \leq a_l$ and $b_j \leq b_l$. In the non-delay permutation schedule consistent with $\sigma l j \theta$, job l cannot start on machine M_k until

$$\partial_k = t(0; a; \sigma; k), \quad 1 \leq k \leq m.$$

Now applying Theorem 4.3.2 we get, for any partial permutation γ of jobs in θ, the inequality

$$t(\partial; a; jl\gamma; k) \leq t(\partial; a; lj\gamma; k), \quad 1 \leq k \leq m.$$

But from the critical path networks of $\sigma jl\gamma$ and $\sigma lj\gamma$ we have for $1 \leq k \leq m$

$$t(0; a; \sigma jl\gamma; k) = t(\partial; a; jl\gamma; k)$$

and

$$t(0; a; \sigma lj\gamma; k) = t(\partial; a; lj\gamma; k).$$

Thus we get $t(0; a; \sigma jl\gamma; k) \leq t(0; a; \sigma lj\gamma; k)$, $1 \leq k \leq m$.

From this result and the given facts $a_j \leq a_1$ and $b_j \leq b_1$ it follows that, the maximum tardiness of the non-delay permutation schedule consistent with $\sigma j 1 \theta$ is no more than that for the non-delay permutation schedule consistent with $\sigma 1 j \theta$. Now Theorem 4.3.3 follows. //

Corollary 4.3.4 : Let $\pi = (\pi(1), \ldots, \pi(n))$ satisfy the conditions of Theorem 4.3.3. Then the non-delay permutation schedule consistent with π (ignoring the job release times) is an optimal permutation schedule to the $(n/m/F/T_{max})$ problem.

In the following Theorem 4.3.5, we use E^*_π defined through (4.2.8) for any arbitrary permutation π.

Theorem 4.3.5 : Suppose $\pi = (\pi(1), \ldots, \pi(n))$ is a permutation such that $a_{\pi(1)} \leq \cdots \leq a_{\pi(n)}$ and $b_{\pi(1)} \leq \cdots \leq b_{\pi(n)}$. Further assume that π is a Johnson's permutation for the two machine flow-shop problems (M_u, M_v), $1 \leq u < v \leq m$. Then the non-delay permutation schedule $S(\pi)$ consistent with π and the job release times $r_j = \max(a_j - E^*_\pi, 0)$, $1 \leq j \leq n$, is an optimal permutation schedule for the $(n/m/F/f)$ problem.

Proof : Using Theorem 4.3.3 and Corollary 4.3.4 we can see that π produces optimal permutation schedules to both the related problems $(n/m/F/a_j \geq 0/T_{max})$ and $(n/m/F/T_{max})$. Therefore, for an arbitrary

permutation σ we have,

$$\Delta(\pi) \leq \Delta(\sigma) \quad \text{and} \quad \hat{T}(\pi) \leq \hat{T}(\sigma).$$

Using $\Delta(\pi) \leq \Delta(\sigma)$ and the monotonically non-decreasing property of the functions g and h in (4.2.8) we get $g(E^*_\pi) \leq g(E^*_\sigma)$.

Now invoking Theorem 4.2.5 we have $f(S(\pi)) \leq f(S(\sigma))$.

This proves Theorem 4.3.5. //

Remark 4.3.6 : If there exists a pair of jobs j and l such that $a_j < a_l$ and $b_j > b_l$ then no π satisfies the conditions of Theorems 4.3.3 and 4.3.5.

4.4 Complexity of the (n/m/F/f) Problem :

For a (n/1/./f) problem there exist (as shown by Sidney (1977) and Lakshmanan et al. (1978))polynomial time complex algorithms if we make the additional assumption that "$a_j < a_l \Longrightarrow b_j \leq b_l$". In this section, we prove that a general (n/m/F/f) problem is NP-complete (see Definition 1.5.8) when this additional assumption is relaxed. In fact (n/m/F/f) problem is NP-complete for $m \geq 2$, even under the additional assumption.

In order to prove the NP-completeness of the (n/m/F/f) problem we reword the problem as follows : Given the data of the (n/m/F/f) problem and a non-negative number D we ask the question - does there exist a feasible permutation schedule $S(\pi)$ such that $f(S(\pi)) \leq D$?

From Remark 1.5.9 we observe that the essential part of a proof of NP-completeness is : Given a specific input x for problem X (which is known to be NP-complete), we must show how to construct a corresponding input y to problem Y (which is to be proved as NP-complete), such that the answer to y is "yes" if and only if the answer to x is "yes". Further, the input length of y, as well as the time taken to construct y, must each be bounded by a polynomial function of m(x), where m is the input length measure for problem X. We do only this part of the proof of NP-completeness for the results we derive. Once we prove this for our problems, Theorems 4.4.1 and 4.4.3 will follow from the facts that the (n/m/F/f) problem is in NP (see Definition 1.5.5) and the relation $\propto$ (see Definition 1.5.7) is transitive.

The known NP-complete problem X used in our proofs is the following form of the partition decision problem in Karp (1972).

Partition decision problem : Given n non-negative weights w_j, $1 \leq j \leq n$, determine if there exists a subset A of $\{1,2, \ldots, n\}$ such that

$$\sum_{j \in A} w_j = \frac{1}{2} \sum_{i=1}^{n} w_j .$$

Theorem 4.4.1 : The (n/1/./f) problem is NP-complete.

Proof : Given a partition decision problem we construct a $((n+1)/1/./f)$ problem as follows. Let $W = \sum_{j=1}^{n} w_j$. Define the set of jobs by $N = \{ j \mid 0 \leq j \leq n \}$. The processing time p_j of job j is defined by

$$p_o = W/2$$

$$p_j = w_j , \quad j \neq 0 \text{ and } j \in N.$$

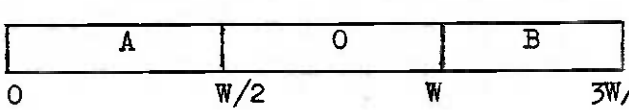
The target start times (a_j) and due dates (b_j) are defined by

$$a_o = W/2 \quad ; \quad b_o = W$$

$$a_j = 0, \quad j \neq 0 \text{ and } j \in N; \quad b_j = 3W/2, \quad j \neq 0 \text{ and } j \in N.$$

The value of D is defined to be zero. The penalty functions g and h are as defined in the $(n/1/./f)$ problem. In this construction, the input length of the problem measured as the sum of the input data is $\frac{3}{2} W (n+2)$, which is clearly bounded by a polynomial in W and n, as is the time necessary to construct a description of this input.

Now it remains to show that the desired solution to partition decision problem exists if and only if there is a schedule $S(\pi)$ of the problem $(n/1/./f)$ such that $f(S(\pi)) \leq D = 0$ (i.e. $f(S(\pi)) = 0$ since we know that $f(S(\pi)) \geq 0$ for any arbitrary schedule $S(\pi)$). If there exists a schedule $S(\pi)$ with $f(S(\pi)) = 0$, then the job '0' must be scheduled as in the following figure.



Now it is clear that $f(S(\pi)) = 0$ if and only if the jobs 1,2, ..., n are partitioned into two sets A and B such that each set has W/2 as the total processing time of jobs in that set. Thus it follows that the answer to this (n/1/./f) problem is "yes" if and only if the answer to partition decision problem is "yes". This proves Theorem 4.4.1. //

Corollary 4.4.2 : The (n/m/F/f) problem for $m \geq 1$ is NP-complete.

Theorem 4.4.3 : The (n/2/F/f) problem with the additional assumption that "$a_j < a_l \Longrightarrow b_j \leq b_l$ for any pair of jobs j and l" is NP-complete.

Proof : For any instance of a partition decision problem with $W = \sum_{j=1}^{n} w_j$, define a ((n+1)/2/F/f) problem as follows :

Let the set of jobs be $N = \{ j \mid 0 \leq j \leq n \}$. The processing times of job j are described by the two tuples $P_j = (p_{j1}, p_{j2})$ where $P_0 = (W/2, W/2)$ and $P_j = (0, w_j)$, $j \neq 0$ and $j \in N$. The target start times and due dates of the jobs are defined by $a_j = 0$, $j \in N$; $b_0 = W$ and $b_j = 3W/2$, $j \neq 0$ and $j \in N$. D is defined as zero and the penalty functions g and h are as defined earlier.

Note that this problem satisfies the additional assumption in a vacuous sense. The proof of Theorem 4.4.3 can be completed by using arguments similar to the arguments in the proof of Theorem 4.4.1. //

Corollary 4.4.4 : The $(n/m/F/f)$ problem for $m \geq 2$ with the additional assumption "$a_j < a_l \Longrightarrow b_j \leq b_l$ for any pair of jobs j and l" is NP-complete.

4.5 Branch-and-Bound Method for the $(n/m/F/f)$ Problem :

Before discussing the branch-and-bound search procedure to solve the $(n/m/F/f)$ problem, we develop the lower bounds required in the search procedure. Recollect that $\Delta(\sigma)$ denotes the objective function value of the non-delay permutation schedule consistent with the permutation σ in the $(n/m/F/a_j \geq 0/T_{max})$ problem. Further in the $(n/m/F/f)$ problem, $S(\sigma)$ is the non-delay permutation schedule consistent with permutation σ and the job release times $r_j = \max(a_j - E^*_\sigma, 0)$, $j \in N$ where E^*_σ is given by (4.2.8).

Proposition 4.5.1 : In a $(n/m/F/f)$ problem, for an arbitrary completion $\sigma\pi$ of σ suppose $\Delta(\sigma\pi) \geq \Delta_H$, a non-negative number. Further let E_H and T_H be non-negative numbers such that $E_H + T_H = \Delta_H$ and $g(E_H) = h(T_H)$. Then we have $f(S(\sigma\pi)) \geq g(E_H)$.

Proof : Proposition 4.5.1 easily follows from Theorem 4.2.5, the monotonically non-decreasing properties of g and h and the relation (4.2.8). //

Given the $(n/m/F/a_j \geq 0/T_{max})$ problem, for any partial permutation $\sigma = (\sigma(1), \ldots, \sigma(q))$ define with the help of machine M_u,

$1 \leq u \leq m$ and jobs in $\bar{\sigma}$, a single machine problem $((n-q)/1/./r_j^u \geq 0/T_{max})$ as follows.

For every job j in $\bar{\sigma} = N - \sigma$ we define :

- strict release time of job j $= r_j^u = \max\left\{Z,\ a_j + \sum_{v=1}^{u-1} p_{jv}\right\}$

  where $Z = \max_{1 \leq k \leq u} \left\{ t(0;\ a;\ \sigma;\ k) + \sum_{v=k}^{u-1} p_{jv} \right\}$,

- due date of job j $= d_j^u = b_j - \sum_{v=u+1}^{m} p_{jv}$,

- processing time of job j $= p_{ju}$.

Let the optimal objective function value of this $((n-q)/1/./r_j^u \geq 0/T_{max})$ problem corresponding to the machine M_u be denoted by

$LB_u\ (r_j^u\ ,\ p_{ju}\ ,\ d_j^u)$, $\quad 1 \leq u \leq m$.

Define,

$$\Delta_\theta = \max\left\{\Delta(\sigma),\ \max_{1 \leq u \leq m} LB_u(r_j^u,\ p_{ju},\ d_j^u)\right\} . \qquad \ldots \quad (4.5.1)$$

Remark 4.5.2 : In the $((n-q)/1/./r_j^u \geq 0/T_{max})$ problem just defined, suppose $r_j^u < r_l^u \Longrightarrow d_j^u \leq d_l^u$ for every pair of jobs j and l in $\bar{\sigma}$. Then there exists a permutation $\pi = (\pi(1), \ldots, \pi(n-q))$ of the jobs in $\bar{\sigma}$ such that $r^u_{\pi(1)} \leq \cdots \leq r^u_{\pi(n-q)}$, $d^u_{\pi(1)} \leq \cdots \leq d^u_{\pi(n-q)}$ and π yields an optimal non-delay permutation schedule of the

$((n-q)/1/./r_j^u \geq 0/T_{max})$ problem. This result follows from Theorem 1 of Sidney (1977). Thus in this case $LB_u(r_j^u, p_{ju}, d_j^u)$ can be worked out exactly in polynomial time. Otherwise this problem is NP-complete.

Remark 4.5.3 : In the $((n-q)/1/./r_j^u \geq 0/T_{max})$ problem, if we take the job release time of job j as $\partial_u = t(0; a; \sigma; u)$ instead of r_j^u, then we get a $((n-q)/1/./T_{max})$ problem with simultaneous release times for all the jobs. We know from Baker (1974) or Conway et al. (1967) that the earliest due date sequence of the jobs in $\bar{\sigma}$ is an optimal permutation of this $((n-q)/1/./T_{max})$ problem. Let $LB_u^1(\partial_u, p_{ju}, d_j^u)$ denote the optimal objective function value of this $((n-q)/1/./T_{max})$ problem.

Define,

$$\Delta_\alpha = \max\left\{\Delta(\sigma), \max_{1 \leq u \leq m} LB_u^1(\partial_u, p_{ju}, d_j^u)\right\}. \quad \ldots \quad (4.5.2)$$

Now using the fact that $r_j^u \geq t(0; a; \sigma; u) = \partial_u$ for every j in $\bar{\sigma}$, we get for $1 \leq u \leq m$,

$$LB_u(r_j^u, p_{ju}, d_j^u) \geq LB_u^1(\partial_u, p_{ju}, d_j^u).$$

Therefore, $\Delta_\theta \geq \Delta_\alpha$.

Remark 4.5.4 : In the $((n-q)/1/./r_j^u \geq 0/T_{max})$ problem suppose preempt - resumption is allowed for the jobs. Then we can find an

optimal solution of this problem by the method outlined in Remark 4.1.7. Call the optimal objective function value of this problem $LB_u^2 (r_j^u , p_{ju} , d_j^u)$. Define,

$$\Delta_\beta = \max\left\{ \Delta (\sigma) , \max_{1 \le u \le m} LB_u^2(r_j^u , p_{ju} , d_j^u) \right\}. \quad \ldots (4.5.3)$$

Now note that for every u such that $1 \le u \le m$ we have,

$$LB_u(r_j^u , p_{ju} , d_j^u) \ge LB_u^2 (r_j^u , p_{ju} , d_j^u).$$

Therefore we get $\Delta_\theta \ge \Delta_\beta$.

It is easy to see that $\Delta_\beta \ge \Delta_\alpha$, since $r_j^u \ge t(0; a; \sigma ; u) = \partial_u$ for every j in $\bar{\sigma}$ and T_{max} is a regular measure of performance for which it is enough to consider permutation schedules alone when all the jobs are released simultaneously.

Lemma 4.5.5 : In a $(n/m/F/a_j \ge 0/T_{max})$ problem, for an arbitrary completion $\sigma\pi$ of a partial permutation σ we have

$$\Delta(\sigma\pi) \ge \Delta_\theta \ge \Delta_\beta \ge \Delta_\alpha .$$

Proof : From the critical path network of $\sigma\pi$ (see Figure 1.3.1) it is easy to see that for $1 \le k \le m$ and $1 \le j \le n-q$ we have

$$t(0; a; \sigma\pi(1), \ldots, \pi(j); k) = t(\partial; a; \pi(1), \ldots, \pi(j); k) \quad \ldots (4.5.4)$$

where $\sigma = (\sigma(1), \ldots, \sigma(q))$; $\pi = (\pi(1), \ldots, \pi(n-q))$ is a permutation

of jobs in $\bar{\sigma}$ and $\partial = (\partial_1, \ldots, \partial_m)$ is such that $\partial_i = t(0; a; \sigma; i)$, $1 \leq i \leq m$. Figure 4.5.1 gives the critical path network of π where ∂ and a provide the machine available and job release times respectively.

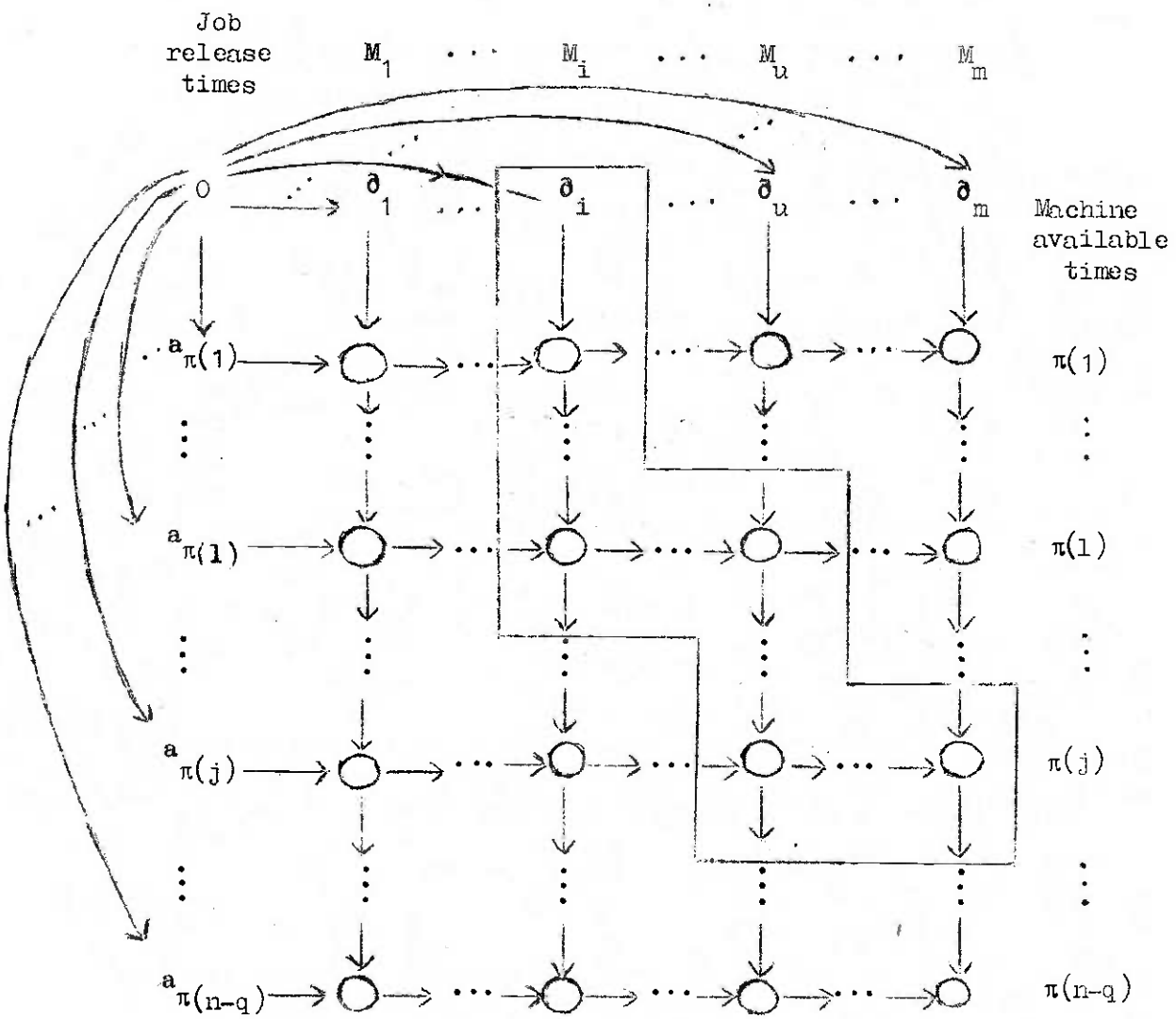


Figure 4.5.1 : Critical path network of π.

From Figure 4.5.1 we see that

$$t(\partial; a; \pi(1), \ldots, \pi(j); m) = \text{the length of a } j\text{-critical path in the figure}$$

$$\geq \text{the length of the circled } j\text{-path in the figure}$$

$$\geq \partial_i + \sum_{k=i}^{u-1} p_{\pi(l),k} + \sum_{v=l}^{j} p_{\pi(v),u} + \sum_{k=u+1}^{m} p_{\pi(j),k} \cdot \quad \ldots \quad (4.5.5)$$

This inequality (4.5.5) is valid for every i, u, l and j such that $1 \leq i \leq u \leq m$ and $1 \leq l \leq j \leq n-q$.

Similarly for every l and j such that $1 \leq l \leq j \leq n-q$ we get,

$$t(\partial; a; \pi(1), \ldots, \pi(j); m) \geq a_{\pi(l)} + \sum_{k=1}^{u-1} p_{\pi(l),k} + \sum_{v=l}^{j} p_{\pi(v),u} + \sum_{k=u+1}^{m} p_{\pi(j),k} \cdot$$

Combining this with (4.5.5) for fixed u, l and j, $1 \leq u \leq m$ and $1 \leq l \leq j \leq n-q$ we have

$$t(\partial; a; \pi(1), \ldots, \pi(j); m) \geq r^{u}_{\pi(l)} + \sum_{v=l}^{j} p_{\pi(v),u} + \sum_{k=u+1}^{m} p_{\pi(j),k} \cdot$$

Using this, (4.5.4) and the definition of $\triangle(\sigma\pi)$, for u, $1 \leq u \leq m$ we have,

$$\Delta(\sigma\pi) \geq \max_{1 \leq j \leq n-q} \left\{ \max(t(\partial\,;\, a;\, \pi(1), \ldots, \pi(j);\, m) - b_{\pi(j)}\,,\, 0) \right\}$$

$$\geq \max_{1 \leq j \leq n-q} \left\{ \max\left[\max_{1 \leq l \leq j} \left\{ r^u_{\pi(l)} + \sum_{v=l}^{j} p_{\pi(v),u} \right\} - d^u_{\pi(j)}\,,\, 0 \right] \right\},$$

$$\geq LB_u\,(r^u_j\,,\, p_{ju}\,,\, d^u_j\,).$$

Combining this with the fact that $\Delta(\sigma\pi) \geq \Delta(\sigma)$ we get

$$\Delta(\sigma\pi) \geq \Delta_\theta\,.$$

Now from Remark 4.5.4 we know that $\Delta_\theta \geq \Delta_\beta \geq \Delta_\alpha$ and hence Lemma 4.5.5 follows. //

In a $(n/m/F/T_{max})$ problem, given any partial permutation $\sigma = (\sigma(1), \ldots, \sigma(q))$, define with the help of machine M_u $(1 \leq u \leq m)$ and the jobs in $\bar{\sigma}$ a $((n-q)/1/./a^u_j \geq 0/T_{max})$ problem as follows. For every job j in $\bar{\sigma}$ we define :

- strict release time of job j $= a^u_j = \max_{1 \leq i \leq u} \left\{ t(\sigma\,;\, i) + \sum_{k=i}^{u-1} p_{jk} \right\}$,
- due date of job j $= d^u_j = b_j - \sum_{k=u+1}^{m} p_{jk}$
- processing time of job j $= p_{ju}$.

Denote the optimal objective function value of this $((n-q)/1/./a^u_j \geq 0/T_{max})$ problem by $LB_u\,(a^u_j\,,\, p_{ju}\,,\, d^u_j\,)$.

Define,

$$\hat{T}_\theta = \max\left\{\hat{T}(\sigma),\ \max_{1\le u\le m} LB_u(a_j^u,\ p_{ju},\ d_j^u)\right\}. \qquad \ldots \quad (4.5.6)$$

Recollect that $\hat{T}(\sigma)$ denotes the objective function value of the non-delay permutation schedule consistent with σ in the $(n/m/F/T_{max})$ problem. Corresponding versions of Remarks 4.5.2, 4.5.3 and 4.5.4 are valid in the present situation as well. Similar to the definitions of Δ_α and Δ_β we define $\hat{T}_\alpha$ and $\hat{T}_\beta$ by

$$\hat{T}_\alpha = \max\left\{\hat{T}(\sigma),\ \max_{1\le u\le m} LB_u^1(t(\sigma;u),\ p_{ju},\ d_j^u)\right\} \ldots (4.5.7)$$

and

$$\hat{T}_\beta = \max\left\{\hat{T}(\sigma),\ \max_{1\le u\le m} LB_u^2(a_j^u,\ p_{ju},\ d_j^u)\right\}. \qquad \ldots \quad (4.5.8)$$

It is easy to verify that $\hat{T}_\theta \ge \hat{T}_\beta \ge \hat{T}_\alpha$.

From Lemma 4.5.5 we can assertain the following corollary.

Corollary 4.5.6 : In a $(n/m/F/T_{max})$ problem, for arbitrary completion $\sigma\pi$ of the partial permutation σ we have

$$\hat{T}(\sigma\pi) \ge \hat{T}_\theta \ge \hat{T}_\beta \ge \hat{T}_\alpha.$$

Theorem 4.5.7 : In a $(n/m/F/f)$ problem, for arbitrary completion $\sigma\pi$ of the partial permutation σ we have

$$f(S(\sigma\pi)) \geq \max\left\{g(E_\Theta),\ h(\hat{T}_\Theta)\right\}$$

$$\geq \max\left\{g(E_\beta),\ h(\hat{T}_\beta)\right\}$$

$$\geq \max\left\{g(E_\alpha),\ h(\hat{T}_\alpha)\right\}$$

where E_Θ, E_β and E_α are obtained from E_H of Proposition 4.5.1 when H takes the values Θ, β and α respectively.

Proof : Theorem 4.5.7 follows easily from Proposition 4.5.1, Lemma 4.5.5, Corollary 4.5.6 and Theorem 4.2.5. //

Remark 4.5.8 : In a (n/m/F/f) problem, for any partial permutation σ, assume that we use Remark 4.5.2 whenever possible and otherwise use Remark 4.5.4. Then we will get a lower bound of $f(S(\sigma\pi))$ for arbitrary completion $\sigma\pi$ of σ such that this lower bound lies between $\max\left\{g(E_\Theta),\ h(\hat{T}_\Theta)\right\}$ and $\max\left\{g(E_\beta),\ h(\hat{T}_\beta)\right\}$.

Remark 4.5.9 : In a (n/m/F/f) problem, given a partial permutation σ, for $1 \leq u \leq m$ we have,

$$LB_u(r_j^u,\ p_{ju},\ d_j^u) \geq LB_u(a_j^u,\ p_{ju},\ d_j^u),$$

$$LB_u^1(\partial_u,\ p_{ju},\ d_j^u) \geq LB_u^1(t(\sigma\ ;\ u),\ p_{ju},\ d_j^u)$$

and $$LB_u^2(r_j^u,\ p_{ju},\ d_j^u) \geq LB_u^2(a_j^u,\ p_{ju},\ d_j^u).$$

Consequently the expressions in the RHS of these inequalities need be computed only when the LHS expressions are strictly positive. Further these inequalities lead to $\Delta_\theta \geq \hat{T}_\theta$, $\Delta_\beta \geq \hat{T}_\beta$ and $\Delta_\alpha \geq \hat{T}_\alpha$. Hence note that $\hat{T}_\theta$, $\hat{T}_\beta$ and $\hat{T}_\alpha$ need be computed only if the respective Δ_θ, Δ_β and Δ_α are strictly positive.

Remark 4.5.10 : In a (n/m/F/f) problem, for a given pair of jobs j and l, and for every pair of machines M_u and M_k such that $1 \leq u < k \leq m$, let

$$\min (p_{ju}, p_{lk}) \leq \min (p_{lu}, p_{jk}).$$

Further assume that $a_j \leq a_l$ and $b_j \leq b_l$.

Then $f(S(\sigma jl\pi)) \leq f(S(\sigma lj\pi))$ where σ and π are arbitrary partial permutations of jobs in $N - \{j,l\}$ such that $\sigma \cap \pi = \emptyset$.

Proof : Define $\delta = (\delta_1, \ldots, \delta_m)$ by $\delta_k = t(0; a; \sigma; k)$, $1 \leq k \leq m$ and use Theorem 4.3.2 to get

$$t(0; a; \sigma jl\gamma ; k) \leq t(0; a; \sigma lj\gamma ; k), \quad 1 \leq k \leq m,$$

where γ is an arbitrary partial permutation of jobs in π. In the same Theorem 4.3.2 if we choose $\delta_k = t(\sigma; k)$, $1 \leq k \leq m$ and $a_i = 0$, $1 \leq i \leq n$ we get

$$t(\sigma jl\gamma ; k) \leq t(\sigma lj\gamma ; k), \quad 1 \leq k \leq m,$$

where again γ is an arbitrary partial permutation of jobs in π.

From these results we can prove that $\Delta(\sigma j l \pi) \leq \Delta(\sigma l j \pi)$ and $\hat{T}(\sigma j l \pi) \leq \hat{T}(\sigma l j \pi)$, using $b_j \leq b_l$. Now using Theorem 4.2.5 and the relation (4.2.8) we get

$$f(S(\sigma j l \pi)) \leq f(S(\sigma l j \pi)). \quad //$$

Remark 4.5.11 : If a pair of jobs l and j satisfy the conditions of Remark 4.5.10 then we have a dominance rule which can be used in the branch-and-bound procedure by eliminating the permutations of the form $\sigma l j \pi$. One can verify the conditions of Remark 4.5.10 at the beginning of the search procedure and eliminate permutations of the form $\sigma l j \pi$ while generating the new nodes.

Using the results of this section one can write down the branch-and-bound procedure. In the following we solve a (5/4/F/f) problem for illustration.

Example 4.5.1 : (5/4/F/f) problem with $g(x) = x$ and $h(x) = 2x$ for $x \geq 0$.

Job j	Processing times				Target times	
	p_{j1}	p_{j2}	p_{j3}	p_{j4}	a_j	b_j
1	2	4	3	7	0	16
2	3	2	8	4	3	21
3	5	3	9	7	6	30
4	1	4	3	2	10	31
5	2	1	4	2	11	37

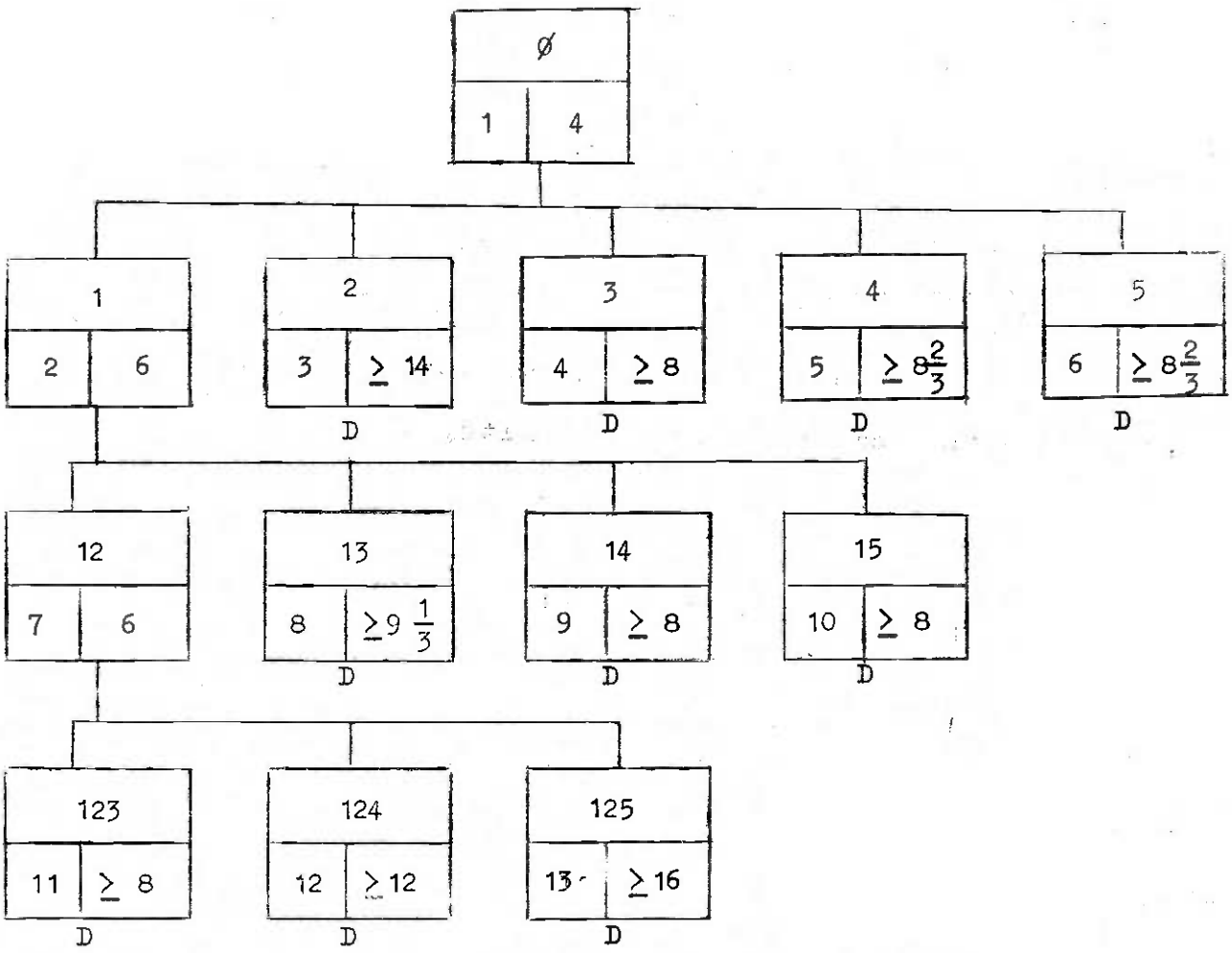
Let π^* denote the permutation corresponding to the best known solution at any stage. At the root node we take $\pi^* = (\pi^*(1), \ldots, \pi^*(n))$ where $b_{\pi^*(1)} \leq \cdots \leq b_{\pi^*(n)}$. In this example, at the root node we find $\pi^* = (1,2,3,4,5)$ with $\Delta(\pi^*) = 4$, $\hat{T}(\pi^*) = 4$, $g(E^*_{\pi^*}) = \frac{8}{3}$ and $h(\hat{T}(\pi^*)) = 8$. Hence $f(S(\pi^*)) = 8$.

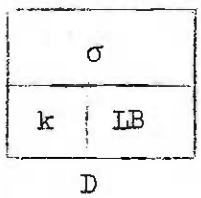
In Figure 4.5.2 we present the tree generated by the branch-and-bound procedure along with relevant information. From Figure 4.5.2, it is clear that the best known solution π^* constructed at the root node, gives an optimal schedule $S(\pi^*)$ with starting time matrix as

$$\begin{pmatrix} 0 & 2 & 6 & 9 \\ 2 & 6 & 9 & 17 \\ 5 & 10 & 17 & 26 \\ 10 & 13 & 26 & 33 \\ 11 & 17 & 29 & 35 \end{pmatrix}$$

and $f(S(\pi^*)) = 8$.

In solving this example dominance rule of Remark 4.5.10 has not helped. The lower bound $LB(\sigma)$ associated with a node representing the partial permutation σ is taken to be $\max\{g(E_\beta), h(\hat{T}_\beta)\}$ given in Theorem 4.5.7. Note that $LB(\sigma)$ is not exactly computed for some of the discarded nodes. The computation of $LB(\sigma)$ is terminated as soon as the optimal solution of a single machine problem corresponding to a machine M_u provides a lower bound $\geq f(S(\pi^*))$.





k : Node number, σ : Partial permutation,
LB : Lower bound, D : The node is discarded since its lower bound is $\geq f(S(\pi^*))$.

At the termination stage $\pi^* = (1,2,3,4,5)$ with $f(S(\pi^*)) = 8$ and thus π^* provides an optimal solution.

Figure 4.5.2 : The tree generated by the branch-and-bound procedure.

Computation of LB(σ) is illustrated for node number 2 which corresponds to the partial permutation $\sigma = (1)$.

	M_k			
	1	2	3	4
$t(0; a; \sigma; k)$	2	6	9	16
$\Delta(\sigma) = 0$				

	M_k			
	1	2	3	4
$t(\sigma; k)$	2	6	9	16
$\hat{T}(\sigma) = 0$				

Note that for a partial permutation σ and u, $1 \leq u \leq m$, we can compute r_j^u, a_j^u and d_j^u using the following relations.

$$r_j^1 = \max\left(t(0; a; \sigma; 1), a_j\right); \quad a_j^1 = t(\sigma; 1); \quad d_j^1 = b_j - \sum_{v=2}^{m} p_{jv};$$

for $2 \leq u \leq m$,

$$r_j^u = \max\left\{r_j^{u-1} + p_{j,(u-1)}, \quad t(0; a; \sigma; u)\right\},$$

$$a_j^u = \max\left\{a_j^{u-1} + p_{j,(u-1)}, \; t(\sigma; u)\right\},$$

and $d_j^u = d_j^{u-1} + p_{jk}$.

For instance, $r_2^1 = \max\left\{t(0; a; \sigma; 1), \; a_2\right\} = \max\{2, 3\} = 3,$

$$r_2^2 = \max\left\{r_2^1 + p_{21}, \; t(0; a; \sigma; 2)\right\}$$

$$= \max\{3 + 3, 6\} = 6.$$

The data and the optimal objective function values of the single machine problems are presented in the following tables and subsequently they are used in computing the required lower bound.

$(4/1/./r_j^1 \geq 0/T_{max})$

j	2	3	4	5
r_j^1	3	6	10	11
p_{j1}	3	5	1	2
d_j^1	7	11	22	30

$LB_1^2 (r_j^1, p_{j1}, d_j^1) = 0$

$(4/1/./r_j^2 \geq 0/T_{max})$

j	2	3	4	5
r_j^2	6	11	11	13
p_{j2}	2	3	4	1
d_j^2	9	14	26	31

$LB_2^2 (r_j^2, p_{j2}, d_j^2) = 0$

$(4/1/./r_j^3 \geq 0/T_{max})$

j	2	3	4	5
r_j^3	9	14	15	14
p_{j3}	8	9	3	4
d_j^3	17	23	29	35

$LB_3^2 (r_j^3, p_{j3}, d_j^3) = 3$

$(4/1./r_j^4 \geq 0/T_{max})$

j	2	3	4	5
r_j^4	17	23	18	18
p_{j4}	4	7	2	2
d_j^4	21	30	31	37

$LB_4^2 (r_j^4, p_{j4}, d_j^4) = 0$

Therefore using (4.5.3) we have $\Delta_\beta = 3$. Now using Proposition 4.5.1 we have $E_\beta = 2$ and $g(E_\beta) = 2$. Note that invoking Remark 4.5.9, we need not solve the $((n-q)/1/./a_j^u \geq 0/T_{max})$ problems for $u = 1,2$ and 4.

$(4/1/./a_j^3 \geq 0/T_{max})$				
j	2	3	4	5
a_j^3	9	10	10	9
p_{j3}	8	9	3	4
d_j^3	17	23	29	35
$LB_3^2 (a_j^3, p_{j3}, d_j^3) = 3$				

Now from the above table using (4.5.8) we get $\hat{T}_\beta = 3$ and $h(\hat{T}_\beta) = 6$. Therefore, $LB(\sigma) = \max\{g(E_\beta), h(\hat{T}_\beta)\} = \max\{2,6\} = 6$.

CHAPTER V

SPECIAL STRUCTURED (n/m/F/f) PROBLEMS

5.0 Introduction :

In this chapter we deal with the flow-shop shceduling problem denoted by (n/m/F/f) under the assumptions A1 to A9 inclusive of A4' and A5' (given in section 1.2 of Chapter I).

Many of the scheduling problems have been proved to be NP-complete (see Bruno et al. (1974), Garey et al. (1975), Garey et al. (1976), Karp (1972), Karp (1975), Ullman (1975) and Ullman (1976)). In view of the complexity of the (n/m/F/f) problems which are NP-complete, significant research has progressed in characterizing special structured (n/m/F/f) problems with polynomial time complex algorithms.

In section 5.1 we present the known special structured flow-shop problems, citing the relevant references. In section 5.2 we discuss definitions and results required for characterizing a new set of special structured flow-shop problems. In section 5.3 we deal with the special structured flow-shop problems having polynomial time complex algorithms.

5.1 Known Special Structured Flow-Shop Problems :

Szwarc (1978) observed that most of the known special structured $(n/m/F/F_{max})$ problems with polynomial time complex algorithms possess the following property : The type of a n-critical path $P_n(\pi)$ in the

critical path network of π remains the same for every permutation $\pi = (\pi(1), \ldots, \pi(n))$ and $P_n(\pi)$ is one of the types given in Figure 5.1.1.

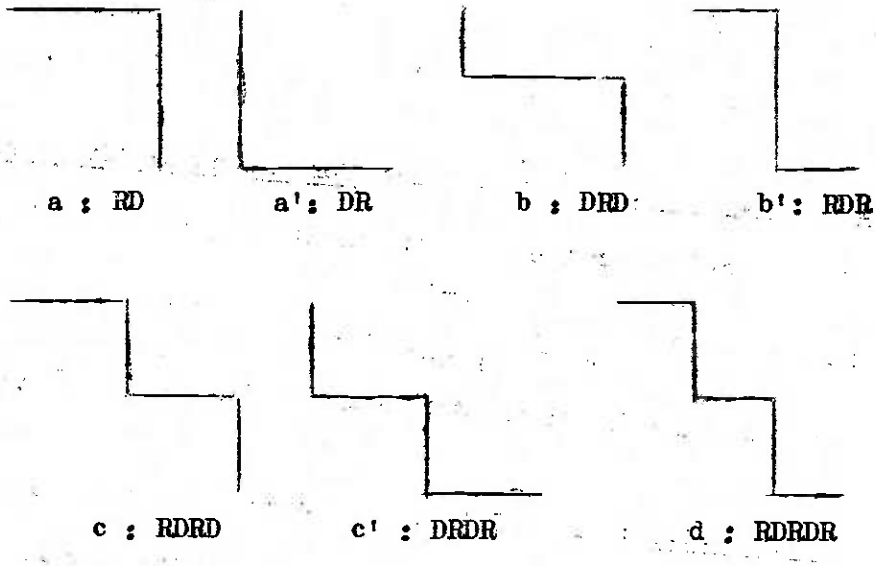


Figure 5.1.1 : Special types of n-critical path $P_n(\pi)$

Remark 5.1.1 : Whenever a n-critical path $P_n(\pi)$ of an arbitrary permutation π is one of the types given in Figure 5.1.1, the $(n/m/F/F_{max})$ problem can be solved by a polynomial time complex algorithm. For details of these algorithms we refer to Szwarc (1978).

We introduce the following two definitions due to Arthanari(1974), for describing the known special structured flow-shop problems.

Definition 5.1.2 : We say that backward cumulative dominance conditions are satisfied for the pair of machines M_u and M_v, $1 \leq u \leq v \leq m-1$, in case $\sum_{k=u}^{v} p_{jk} \leq \sum_{k=u+1}^{v+1} p_{lk}$ holds for every pair of jobs j and l, $j \neq l$ and denote these conditions by BCD (M_u, M_v).

Definition 5.1.3 : We say that forward cumulative dominance conditions are satisfied for the pair of machines M_u and M_v, $1 \leq u \leq v \leq m-1$, in case $\sum_{k=u}^{v} p_{jk} \geq \sum_{k=u+1}^{v+1} p_{lk}$ holds for every pair of jobs j and l, $j \neq l$ and denote these conditions by FCD (M_u, M_v).

Remark 5.1.4 : If BCD (M_u, M_v) holds in a $(n/m/F/f)$ problem, then FCD $(M_{(m-v)'}, M_{(m-u)'})$ holds in the reverse problem $(n/m^R/F/f)$ where $k' = m - k + 1$, $1 \leq k \leq m$. The converse is also true.

Remark 5.1.5 : In a $(n/m/F/f)$ problem, for every permutation $\pi = (\pi(1), \ldots, \pi(n))$, a n-critical path $P_n(\pi)$ is of the type RD in Figure 5.1.1 (a) (DR in Figure 5.1.1 (a')) if and only if BCD (M_u, M_{m-1}), $1 \leq u \leq m-1$ (FCD (M_1, M_v), $1 \leq v \leq m-1$) hold. This remark is a rephrasing of Theorems 1 and 2 by Ramamurthy et al.(1978).

We describe below the known special structured $(n/m/F/F_{max})$ problems with polynomial time complex algorithms.

Case 1 : For some λ, $1 \leq \lambda \leq m$ assume BCD $(M_u, M_{\lambda-1})$, $1 \leq u \leq \lambda-1$ and FCD (M_λ, M_v), $\lambda \leq v \leq m-1$.

This case was independently handled by Arthanari (1974) and Grabowski et al. (1975). Note that for $\lambda = 1$ the given conditions reduces to only FCD (M_1, M_v), $1 \leq v \leq m-1$. Similar interpretation will be given when some conditions are required to be satisfied in aninfeasible range. For subcases of this case see Szwarc (1978) and Burdyuk (1969). It can be easily shown that for an arbitrary permutation π, at least one n-critical path $P_n(\pi)$ is of the type a', b' and a of Figure 5.1.1 for $\lambda=1$, $2 \leq \lambda \leq m-1$ and $\lambda = m$ respectively. Further, for a given λ, at least one n-critical path $P_n(\pi)$ passes through all the cells in column λ.

Case 2 : For some λ, $1 \leq \lambda \leq m-1$ assume FCD (M_1, M_v), $1 \leq v \leq \lambda-1$ and BCD (M_u, M_{m-1}), $\lambda+1 \leq u \leq m-1$.

This case was solved by Grabowski et al. (1975). Arthanari (1974) independently solved this case for $\lambda = 1$ and m-1. For subcases of this case see Johnson (1954), Burdyuk (1969), Arthanari (1974), Gupta (1975a), Nabeshima (1977), Szwarc (1977a) and Szwarc (1978). Under this case, for an arbitrary permutation π, at least one n-critical path $P_n(\pi)$ is of the type b of Figure 5.1.1.

Case 3 : For some λ and λ' such that $1 < \lambda \leq \lambda' \leq m-1$ assume BCD $(M_u, M_{\lambda-1})$, $1 \leq u \leq \lambda-1$; FCD (M_λ, M_v), $\lambda \leq v \leq \lambda'-1$ and BCD (M_u, M_{m-1}), $\lambda'+1 \leq u \leq m-1$.

Ramamurthy et al. (1978) solved this case. Arthanari (1974) handled this case for $\lambda = \lambda' = m-1$. Subcases of this case were tackled by Arthanari et al. (1971), Szwarc (1974) and Gupta (1975a). Under this case, for an arbitrary permutation π at least one n-critical path $P_n(\pi)$ is of the type c (of Figure 5.1.1) with its downward turns at columns λ and m.

Case 3' : For some λ and λ' such that $1 < \lambda \leq \lambda' \leq m-1$ assume FCD (M_1, M_v), $1 \leq v \leq \lambda-2$; BCD $(M_u, M_{\lambda'-1})$, $\lambda \leq u \leq \lambda'-1$ and FCD $(M_{\lambda'}, M_v)$, $\lambda' \leq v \leq m-1$.

This case was solved by Ramamurthy et al. (1978). When $\lambda = \lambda' = 2$, this case was handled by Arthanari (1974). For subcases of this case see Arthanari et al. (1971), Gupta (1975a) and Szwarc (1974a). Here, for any arbitrary permutation π, at least one n-critical path $P_n(\pi)$ is of the type c' of Figure 5.1.1. Further $P_n(\pi)$ makes its downward turns at columns 1 and λ'. Using Remark 5.1.4 one can show that this case reduces to case 3 for the reverse problem $(n/m^R/F/F_{max})$.

Case 4 : For some λ and λ' such that $1 < \lambda \leq \lambda' < m-1$ assume BCD $(M_u, M_{\lambda-1})$, $1 \leq u \leq \lambda-1$; FCD (M_λ, M_v), $\lambda \leq v \leq \lambda' - 1$ and FCD$(M_{\lambda'+1}, M_v)$, $\lambda'+1 \leq v \leq m-1$.

Ramamurthy et al. (1978) solved this case. Subcases of this case with $\lambda = \lambda'$ were solved by Szwarc (1977a), Szwarc (1978) and Nabeshima (1961). Under this case, for an arbitrary permutation π at least one n-critical path $P_n(\pi)$ is of the type d of Figure 5.1.1. Further, $P_n(\pi)$ makes its downward turns at columns λ and $\lambda'+1$.

Case 4' : For some λ and λ' such that $2 < \lambda \leq \lambda' \leq m-1$ assume BCD $(M_u, M_{\lambda-2})$, $1 \leq u \leq \lambda-2$; BCD$(M_u, M_{\lambda'-1})$, $\lambda \leq u \leq \lambda'-1$ and FCD $(M_{\lambda'}, M_v)$, $\lambda' \leq v \leq m-1$.

This case also was solved by Ramamurthy et al. (1978). Under this case, for an arbitrary permutation π at least one n-critical path $P_n(\pi)$ is of the type d of Figure 5.1.1. Here, $P_n(\pi)$ makes its downward turns at columns $\lambda-1$ and λ'. It is easy to show (using Remark 5.1.4) that this case reduces to case 4 for the reverse problem $(n/m^R/F/F_{max})$.

Case 5 : For all the two machine flow-shop problems (M_u, M_v), $1 \leq u < v \leq m$, $(u,v) \neq (1,m)$, there is a common Johnson's permutation.

This case was solved by Achuthan (1977) and is contained in section 3.2 of this thesis. For m=3 this case was handled by Burns et al.(1975) and Szwarc (1977).

Case 6 : Assume (i) Given any pair of jobs j and l, either $p_{lk} \geq p_{jk}$, $1 \leq k \leq m$ or $p_{lk} \leq p_{jk}$, $1 \leq k \leq m$ and

(ii) $\min (p_{j1}, p_{jm}) \geq p_{jk}$ for all $j \in N$ and $1 \leq k \leq m$.

This case was handled by Szwarc (1977a). In fact when m=3, Szwarc (1977a) handled the case just with the condition (ii). Under this case, for an arbitrary permutation π at least one n-critical path $P_n(\pi)$ is of the type b of Figure 5.1.1.

In addition to the cases already discussed, there are special structured $(n/m/F/F_{max})$ problems where sufficient conditions are derived for a particular permutation to be optimal and these sufficient conditions can be verified through polynomial time complex algorithms. For these special structured problems see Szwarc (1974), Szwarc(1977a), Burns et al. (1976), Achuthan (1977) and Dudek et al. (1975).

There are some more special structured $(n/m/F/F_{max})$ problems where the suggested algorithms have computational complexity of order $O(2^n)$. For such special structured problems see Dudek et al. (1976) and Szwarc (1978).

Further known special structured (n/m/F/f) problems other than $(n/m/F/F_{max})$ problems are outlined in the following discussion.

Case I : In a $(n/m/F/f)$ problem assume $BCD(M_u, M_{m-1})$, $1 \leq u \leq m-1$.

This case was solved by Arthanari (1974) for the following objective functions defined for an arbitrary permutation $\sigma = (\sigma(1), \ldots, \sigma(n))$.

(a) Weighted sum of completion times : $\sum_{j=1}^{n} w_{\sigma(j)}\, t(\sigma(1), \ldots, \sigma(j); m)$ ... (5.3.1)

where w_l is a non-negative weight associated with job l, $l \in N$.

(b) Number of tardy jobs : $\left| \{\sigma(j) \mid T_{\sigma(j)} > 0\} \right|$ where $T_{\sigma(j)}$ is tardiness of job $\sigma(j)$ for given due dates d_l associated with job $l \in N$. ... (5.3.2)

(c) Maximum penalty on tardiness : $\max_{1 \leq j \leq n} h_{\sigma(j)}(T_{\sigma(j)})$ where $T_{\sigma(j)}$ is tardiness of job $\sigma(j)$ and $h_{\sigma(j)}$ is a non-decreasing function of $T_{\sigma(j)}$, $1 \leq j \leq n$. ... (5.3.3)

(d) Sum of the absolute deviations of completion times :

$$\sum_{j<l} \left| t(\sigma(1), \ldots, \sigma(j); m) - t(\sigma(1), \ldots, \sigma(l); m) \right| \quad \ldots (5.3.4)$$

Case II : In a $(n/m/F/f)$ problem assume $FCD(M_1, M_v)$, $1 \leq v \leq m-1$.

This case was solved by Arthanari (1974) for the objectives functions given in (a), (b) and (c) of case I.

5.2 Aggregate Cumulative Dominance Conditions :

Definition 5.2.1 : We say that aggregate backward cumulative dominance conditions are satisfied for the triplet (M_u, M_v, M_w) of machines, $1 \leq u \leq v \leq w \leq m-1$ in case

$$\sum_{k=u}^{v} p_{ik} + \sum_{k=v}^{w} p_{jk} \leq \sum_{k=u+1}^{v+1} p_{lk} + \sum_{k=v+1}^{w+1} p_{ik}$$

holds for every triplet (i,j,l) of jobs and denote these conditions by ABCD (M_u, M_v, M_w).

Definition 5.2.2 : We say that aggregate forward cumulative dominance conditions are satisfied for the triplet (M_u, M_v, M_w) of machines, $1 \leq u \leq v \leq w \leq m-1$ in case

$$\sum_{k=u}^{v} p_{ik} + \sum_{k=v}^{w} p_{jk} \geq \sum_{k=u+1}^{v+1} p_{lk} + \sum_{k=v+1}^{w+1} p_{ik}$$

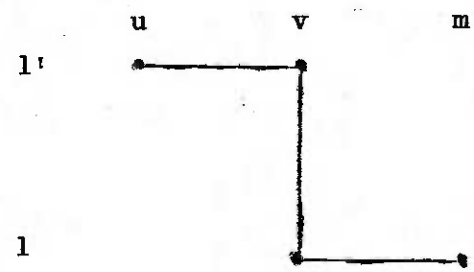
holds for every triplet (i,j,l) of jobs and denote these conditions by AFCD (M_u, M_v, M_w).

Remark 5.2.3 : If ABCD (M_u, M_v, M_w) holds in the $(n/m/F/f)$ problem, then AFCD $(M_{(m-w)'}, M_{(m-v)'}, M_{(m-u)'})$ holds in the reverse problem $(n/m^R/F/f)$ where $k' = m - k + 1$, $1 \leq k \leq m$. The converse is also true.

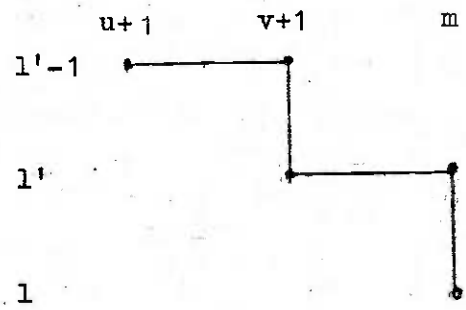
We denote the (n/m/F/f) problem by ($n/m, \delta_k \geq 0/F/f$) when the assumption A4' of section 1.2 is relaxed.

Lemma 5.2.4 : In the ($n/m, \delta_k \geq 0/F/f$) problem, assume ABCD(M_u, M_v, M_{m-1}) for all pairs (u,v) such that $1 \leq u \leq v \leq m-1$. Then in the critical path network of an arbitrary permutation $\sigma = (\sigma(1), \ldots, \sigma(n))$, there exists at least one j-critical path for every $j \geq 3$ such that the j-critical path passes through the cell (2, m).

Proof : Suppose to the contrary. Now let l be the smallest $j \geq 3$ for which the lemma is not valid. Then a l-critical path $P_l(\sigma)$ of σ will consist of a RDR segment as its final portion and otherwise $P_l(\sigma)$ is a DR segment. We discuss the proof when $P_l(\sigma)$ consists of a RDR segment as its final portion and similar arguments go through for the other case as well. Let the final portion of $P_l(\sigma)$ be a RDR segment denoted by γ as represented by the following figure where $u < v \leq m-1$.



Case (i) $l = l'+1$: Consider the following RDRD segment denoted by γ'.



Now,

the length of $\gamma = \sum_{k=u}^{v} p_{\sigma(l'),k} + \sum_{k=v}^{m-1} p_{\sigma(l),k} + p_{\sigma(l),m}$

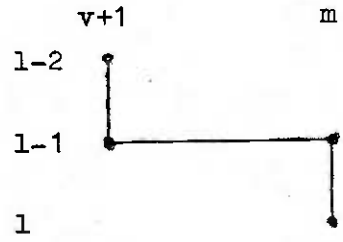
$$\leq \sum_{k=u+1}^{v+1} p_{\sigma(l'-1),k} + \sum_{k=v+1}^{m} p_{\sigma(l'),k} + p_{\sigma(l),m},$$

(by ABCD (M_u, M_v, M_{m-1}))

= the length of γ'.

Replacing γ by γ' in $P_l(\sigma)$ we get another l-critical path $P_l'(\sigma)$ which passes through the cell $(l-1, m)$.

Case (ii) $l > l'+1$: Consider the following DRD segment denoted by γ'.



Again using ABCD (M_v, M_v, M_{m-1}) note that the length of γ' is greater than or equal to the length of the DR segment (denoted by γ'') connecting the end cells (l-1, v) and (l, m) with its right turn at the cell (l, v). Further note that γ'' is a final portion of $P_l(\sigma)$. Now again replacing γ'' by γ' in $P_l(\sigma)$ we get another l-critical path $P_l'(\sigma)$ which passes through the cell (l-1, m).

Now observe that similar arguments go through for the case when $P_l(\sigma)$ is a DR segment. Thus we have produced a l-critical path $P_l'(\sigma)$ passing through the cell (l-1, m). Now using the definition of j-critical path we get a (l-1)-critical path from $P_l'(\sigma)$ by deleting the end cell (l,m) from $P_l'(\sigma)$. This contradicts the choice of l if $l > 3$ and otherwise we have proved the lemma. This completes the proof of Lemma 5.2.4. //

Theorem 5.2.5 : In the $(n/m, \partial_k \geq 0/F/f)$ problem, for some λ, $1 \leq \lambda \leq m-1$ assume ABCD (M_u, M_v, M_λ) for all pairs (u,v) such that $1 \leq u \leq v \leq \lambda$. Then for an arbitrary permutation $\sigma = (\sigma(1), \ldots, \sigma(n))$ we have for $j \geq 3$,

$$t(\partial; \sigma(1), \ldots, \sigma(j); \lambda+1) = t(\partial; \sigma(1), \sigma(2); \lambda+1) + \sum_{s=3}^{j} p_{\sigma(s), \lambda+1}.$$

Proof : Consider the critical path network of σ with reference to the $(n/\lambda+1, \partial_k \geq 0/F/f)$ problem obtained from the $(n/m, \partial_k \geq 0/F/f)$

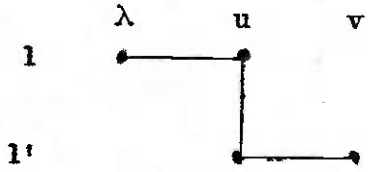
problem by restricting our attention to the first $\lambda+1$ machines. Now invoking Lemma 5.2.4 to the $(n/\lambda+1, \delta_k \geq 0/F/f)$ problem we get Theorem 5.2.5. //

The following remark follows immediately from Theorem 5.2.5.

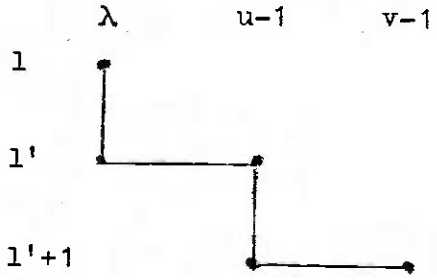
Remark 5.2.6 : In the $(n/m, \delta_k \geq 0/F/f)$ problem, for some λ, $1 \leq \lambda \leq m-1$ assume ABCD (M_u, M_v, M_λ) for all pairs (u,v) such that $1 \leq u \leq v \leq \lambda$. Let $\sigma = (\sigma(1), \sigma(2))$ be a given partial permutation. Let π^* be an optimal solution of the $(|\bar{\sigma}|/m-\lambda, \hat{\delta}_k \geq 0/F/f)$ problem obtained from the $(n/m, \delta_k \geq 0/F/f)$ problem by restricting our attention to jobs in $\bar{\sigma}$ and the machines $M_{\lambda+1}, \ldots, M_m$ in that order where the machine available times are given by $\hat{\delta}_k = t(\delta; \sigma(1), \sigma(2); k)$, $\lambda+1 \leq k \leq m$. Then $f(\sigma\pi^*) \leq f(\sigma\pi)$ for every completion $\sigma\pi$ of σ. Thus the $(n/m, \delta_k \geq 0/F/f)$ problem can be solved by solving the $n(n-1)$ related $(|\bar{\sigma}|/m-\lambda, \hat{\delta}_k \geq 0/F/f)$ problems corresponding to the $n(n-1)$ partial permutations $\sigma = (\sigma(1), \sigma(2))$.

Lemma 5.2.7 : In the $(n/m/F/f)$ problem, for some λ, $1 \leq \lambda \leq m-1$ assume AFCD (M_λ, M_u, M_v) for all pairs (u,v) such that $\lambda \leq u \leq v \leq m-1$. Then in the critical path network of an arbitrary permutation $\sigma = (\sigma(1), \ldots, \sigma(n))$, there exists at least one n-critical path which passes through at least one of the two cells viz. $(n-1, \lambda)$ and (n, λ).

Proof : Suppose to the contrary. Now let l be the largest $j \leq n-2$, such that a n-critical path $P_n(\sigma)$ passes through the cell (j, λ). We shall denote the portion of $P_n(\sigma)$ from the cell (l, λ) onwards by $\bar{P}_n(\sigma)$. Then $\bar{P}_n(\sigma)$ has a RDR segment in its initial portion or $\bar{P}_n(\sigma)$ is a RD segment. Let the initial portion of $\bar{P}_n(\sigma)$ be a RDR segment denoted by γ as in the following figure where $\lambda < u \leq v \leq m$.



Case (i) $l' = l+1$: Consider the following DRDR segment denoted by γ'.



Now,

the length of $\gamma = p_{\sigma(l),\lambda} + \sum_{k=\lambda+1}^{u} p_{\sigma(l),k} + \sum_{k=u}^{v} p_{\sigma(l'),k}$

$\leq p_{\sigma(l),\lambda} + \sum_{k=\lambda}^{u-1} p_{\sigma(l'),k} + \sum_{k=u-1}^{v-1} p_{\sigma(l'+1),k}$,

(by AFCD $(M_\lambda , M_{u-1}, M_{v-1})$),

= the length of γ'.

Replacing γ by γ' in $P_n(\sigma)$ we get another n-critical path $P_n'(\sigma)$ which passes through the cell $(l+1; \lambda)$. The proof of this fact for case (ii) viz. $l' > l+1$ and the remainder of the proof are analogous to the proof of Lemma 5.2.4. This completes the proof of Lemma 5.2.7.//

Theorem 5.2.8 : In the (n/m/F/f) problem, for some λ, $1 \leq \lambda \leq m-1$ assume AFCD (M_λ, M_u, M_v) for all pairs (u,v) such that $\lambda \leq u \leq v \leq m-1$. Then for an arbitrary permutation $\sigma = (\sigma(1), \ldots, \sigma(n))$ we have for $j \geq 3$ and $\lambda+1 \leq k \leq m$,

$$t(\sigma(1), \ldots, \sigma(j); k) = \max \Bigg\{ t(\sigma(1), \ldots, \sigma(j-1); \lambda) + \max_{\lambda+1 \leq q \leq k} \left[\sum_{u=\lambda+1}^{q} p_{\sigma(j-1),u} + \sum_{u=q}^{k} p_{\sigma(j),u} \right],$$

$$t(\sigma(1), \ldots, \sigma(j); \lambda) + \sum_{u=\lambda+1}^{k} p_{\sigma(j),u} \Bigg\}.$$

Proof : This Theorem follows easily from Lemma 5.2.7. //

5.3 New Special Structured Flow-Shop Problems :

In this section we consider new special structured flow-shop problems with polynomial time complex algorithms. These special structured problems are characterised through the BCD, FCD, ABCD and AFCD conditions.

5.3.1 Special Structured $(n/m/F/F_{max})$ Problems :

Case 1 : In the $(n/m/F/F_{max})$ problem, for some λ, $1 \leq \lambda \leq m$ assume ABCD $(M_u, M_v, M_{\lambda-1})$, $1 \leq u \leq v \leq \lambda-1$ and AFCD (M_λ, M_u, M_v), $\lambda \leq u \leq v \leq m-1$.

Using Lemma 5.2.4 and Lemma 5.2.7 it is easy to show that there exists a n-critical path $P_n(\sigma)$ of an arbitrary permutation $\sigma = (\sigma(1), \ldots \sigma(n))$ which passes through the cells (j, λ), $2 \leq j \leq n-1$. Therefore, if the given λ is such that $2 \leq \lambda \leq m-1$ then we have

$$t(\sigma; m) = \max_{1 \leq u \leq \lambda} \left(\sum_{k=1}^{u} p_{\sigma(1),k} + \sum_{k=u}^{\lambda} p_{\sigma(2),k} \right) + \sum_{j=3}^{n-2} p_{\sigma(j),\lambda}$$

$$+ \max_{\lambda \leq v \leq m} \left(\sum_{k=\lambda}^{v} p_{\sigma(n-1),k} + \sum_{k=v}^{m} p_{\sigma(n),k} \right)$$

for an arbitrary permutation $\sigma = (\sigma(1), \ldots, \sigma(n))$. Thus the permutation $(i^*, j^*, \sigma(3), \ldots, \sigma(n-2), l^*, s^*)$ solves the $(n/m/F/F_{max})$ problem optimally if $(\sigma(3), \ldots, \sigma(n-2))$ is a permutation of jobs in $N - \{i^*, j^*, l^*, s^*\}$ and

$$A(i^*,j^*,l^*,s^*) = \min \{ A(i,j,l,s) \mid i,j,l \text{ and } s \text{ are distinct jobs in } N \}$$

where

$$A(i,j,l,s) = \max_{1 \le u \le \lambda} \left(\sum_{k=1}^{u} p_{ik} + \sum_{k=u}^{\lambda} p_{jk} \right) + \max_{\lambda \le v \le m} \left(\sum_{k=\lambda}^{u} p_{lk} + \sum_{k=v}^{m} p_{sk} \right) - \left(p_{i\lambda} + p_{j\lambda} + p_{l\lambda} + p_{s\lambda} \right).$$

Obviously computational complexity of finding $A(i^*, j^*, l^*, s^*)$ is $O(n^4)$. When $\lambda = 1(m)$ note that we have to fix the last (first) two jobs suitably. Thus when $\lambda = 1$ or m the computational complexity of the method reduces to $O(n^2)$.

Remark 5.3.1 : When $m=3$, the above case 1 for $\lambda = 2$ reduces to the new case III with $q = q' = 2$ discussed in page 39 of this thesis.

Case 2 : In the $(n/m/F/F_{max})$ problem, for some λ, $1 \le \lambda \le m$ assume $ABCD(M_u, M_v, M_{\lambda-1})$, $1 \le u \le v \le \lambda - 1$ and $FCD(M_\lambda, M_u)$, $\lambda \le u \le m-1$.

In this case, using Lemma 5.2.4 and Remark 5.1.5 we can prove that the permutation (i^*, j^*, σ, l^*) solves the $(n/m/F/F_{max})$ problem optimally if σ is a permutation of jobs in $N - \{i^*, j^*, l^*\}$ and $A(i^*, j^*, l^*) = \min \{ A(i,j,l) \mid i,j \text{ and } l \text{ are distinct jobs in } N \}$ where

$$A(i,j,l) = \max_{1 \le u \le \lambda} \left(\sum_{k=1}^{u} p_{ik} + \sum_{k=u}^{\lambda} p_{jk} \right) + \sum_{k=\lambda+1}^{m} p_{lk} - p_{i\lambda} - p_{j\lambda}.$$

Case 2' : In the $(n/m/F/F_{max})$ problem, for some λ, $1 \leq \lambda \leq m$ assume BCD $(M_u, M_{\lambda-1})$, $1 \leq u \leq \lambda-1$ and AFCD(M_λ, M_u, M_v), $\lambda \leq u \leq v \leq m-1$.

Using Remarks 5.1.4 and 5.2.3 one can easily see that this case can be solved by solving the reverse problem $(n/m^R/F/F_{max})$ which satisfies the conditions of case 2.

Remark 5.3.2 : Observe that the BCD and FCD conditions are stronger than the ABCD and AFCD conditions. Consequently the special structured $(n/m/F/F_{max})$ problems satisfying BCD or FCD conditions need lesser computational effort than those satisfying ABCD or AFCD conditions.

Case 3 : In the $(n/m/F/F_{max})$ problem, for some λ and λ' such that $1 \leq \lambda < \lambda' \leq m$ assume ABCD$(M_u, M_v, M_{\lambda-1})$, $1 \leq u \leq v \leq \lambda-1$; BCD$(M_u, M_{\lambda'-1})$, $\lambda+1 \leq u \leq \lambda'-1$ and AFCD$(M_{\lambda'}, M_u, M_v)$, $\lambda' \leq u \leq v \leq m-1$.

In this case using Remark 5.1.5, Lemma 5.2.4 and Lemma 5.2.7 observe that for an arbitrary permutation $\sigma = (\sigma(1), \ldots, \sigma(n))$ the n-critical path $P_n(\sigma)$ passes through the cells $(2, \lambda)$ and $(n-1, \lambda')$ and is of type b of Figure 5.1.1 in the columns restricted to the machines M_u, $\lambda \leq u \leq \lambda'$. Thus if the first two and the last two jobs are fixed then the remaining $(n-4)$ jobs can be optimally fixed as the

Johnson's permutation of the two machine flow-shop problem $\left(\sum_{k=\lambda}^{\lambda'-1} M_k , \sum_{\lambda+1}^{\lambda'} M_k\right)$, restricting our attention to the remaining $(n-4)$ jobs when $1 < \lambda < \lambda' < m$. Let π be a Johnson's permutation of the two machine flow-shop problem $\left(\sum_{k=\lambda}^{\lambda'-1} M_k , \sum_{\lambda+1}^{\lambda'} M_k\right)$ and $\pi_{i,j,l,s}$ denote the restriction of π to the set $N - \{i,j,l,s\}$ of jobs. Now $(i^*, j^*, \pi_{i^*,j^*,l^*,s^*}, l^*, s^*)$ solves this case optimally for $1 < \lambda < \lambda' < m$ if $F_{max}(i^*, j^*, \pi_{i^*,j^*,l^*,s^*}, l^*, s^*)$

$$= \min \left\{ F_{max}(i,j, \pi_{i,j,l,s}, l,s) \mid i,j,l \text{ and } s \text{ are distinct jobs in } N \right\}$$

Observe that when $\lambda' = m$ we have to choose the best permutation from among the permutations of the form $(i,j,\pi_{i,j})$ where $i \neq j$ and $\pi_{i,j}$ is the restriction of π to the set $N - \{i,j\}$. Similarly when $\lambda = 1$ we restrict our attention to the permutations of the form $(\pi_{l,s}, l,s)$ where $l \neq s$.

Remark 5.3.3 : When m=3, case 3 of this subsection reduces to :

(i) case I with q=2 (page 28 of this thesis) when $\lambda=2$ and $\lambda' = 3$.

and (ii) case II with q=2 (page 38 of this thesis) when $\lambda = 1$ and $\lambda' = 2$.

Remark 5.3.4 : Suppose that the conditions BCD $(M_u, M_{\lambda'-1})$, $\lambda+1 \leq u \leq \lambda'-1$ in case 3, are replaced by

"for some λ'' such that $\lambda \leq \lambda'' \leq \lambda' - 1$ assume FCD(M_λ, M_v), $\lambda \leq v \leq \lambda'' - 1$ and BCD$(M_u, M_{\lambda'-1})$, $\lambda'' + 1 \leq u \leq \lambda' - 1$ ".

This replacement provides a special structure with reference to the machines M_u, $\lambda+1 \leq u \leq \lambda'-1$ and this structure is similar to the known case 2 in section 5.1. Further this replacement provides a generalisation of case 3 where the results of case 3 are still applicable.

Case 4 : In the $(n/m/F/F_{max})$ problem, for some λ, $1 < \lambda < m-1$, assume ABCD $(M_u, M_v, M_{\lambda-1})$, $1 \leq u \leq v \leq \lambda-1$ and FCD $(M_{\lambda+1}, M_v)$, $\lambda+1 \leq v \leq m-1$.

In this case we can prove that for an arbitrary two sided partial permutation (σ_1, σ_2) with $|\sigma_1| = 2$ and $|\sigma_2| = 1$ the remaining n-3 positions can be optimally fixed as Johnson's permutation of the two machine flow-shop problem $(M_\lambda, M_{\lambda+1})$. Hence we can provide a method with computational complexity $O(n^3)$.

Case 4' : In the $(n/m/F/F_{max})$ problem, for some λ, $1 < \lambda < m-1$, assume BCD (M_u, M_λ), $1 \leq u \leq \lambda-1$ and AFCD$(M_{\lambda+1}, M_u, M_v)$, $\lambda+1 \leq u \leq v \leq m-1$.

Case 4' can besolved by solving the reverse problem $(n/m^R/F/F_{max})$ which satisfies the conditions of case 4.

Remark 5.3.5 : It should be observed from the discussions of this subsection that one can construct many more special structured $(n/m/F/F_{max})$ problems of interest with the help of certain other combinations of the BCD, FCD, ABCD and AFCD conditions.

5.3.2 Special Structured $(n/m/F/f)$ Problems where $f \neq F_{max}$:

In this subsection we deal with special structured flow-shop problems when the objective function f is different from F_{max}. Further we assume that while solving the $(n/m/F/f)$ problem, we restrict our attention to the non-delay permutation schedules alone.

Case I : In a $(n/m/F/f)$ problem, for some λ, $1 \leq \lambda \leq m$, assume BCD$(M_u, M_{\lambda-1})$, $1 \leq u \leq \lambda-1$ and FCD (M_λ, M_v), $\lambda \leq v \leq m-1$.

Under this case, using Remark 5.1.5, for any given permutation $\pi = (\pi(1), \ldots, \pi(n))$, for every j, $1 \leq j \leq n$, there exists at

least one j-critical path $P_j(\pi)$ of π such that $P_j(\pi)$ is of the type b' in Figure 5.1.1. For the given λ, this $P_j(\pi)$ passes through the cells $(1,k)$, $1 \leq k \leq \lambda-1$; (l, λ), $1 \leq l \leq j$ and (j,k), $\lambda+1 \leq k \leq m$. Therefore, under the non-delay permutation schedule consistent with π we have,

$$t(\pi(1), \ldots, \pi(j); m) = \sum_{k=1}^{\lambda-1} p_{\pi(1),k} + \sum_{s=1}^{j} p_{\pi(s),\lambda} + \sum_{k=\lambda+1}^{m} p_{\pi(j),k} \quad \ldots \quad (5.3.5)$$

Given a $(n/m/F/f)$ problem, corresponding to the machine M_λ and a fixed job j in N, we define a single machine problem denoted by $(n-1\ /1/\ .\ /\ f_j^\lambda)$ as follows :

- the set of jobs is given by $N_j = N - \{j\}$.
- the processing time of job l in N_j is $p_{l\lambda}$.
- f_j^λ is the objective function which is to be minimized over the set of all non-delay permutation schedules of the (n-1) jobs in N_j. f_j^λ is defined by $f_j^\lambda(\pi) = f(j\pi)$ for an arbitrary permutation π of the jobs in N_j.

Remark 5.3.6 : If f is a regular measure of performance so is f_j^λ for every j and λ such that j is in N and $1 \leq \lambda \leq m$.

Remark 5.3.7 : Under case I, given a partial permutation $\sigma = (j)$ finding an optimal completion of σ in the $(n/m/F/f)$ problem is equivalent to solving the $(n-1/1/./f_j^{\lambda})$ problem for an optimal solution. Thus solving n such single machine problems, we can get an optimal solution of the $(n/m/F/f)$ problem.

In the following we discuss specific objective functions f for which the $(n/m/F/f)$ problem under case I can be solved using the known polynomial time complex algorithms for the related single machine problems $(n-1/1/./f_j^{\lambda})$, for all j in N.

(A) Weighted sum of completion times $C_w(\sigma)$:

For an arbitrary permutation $\sigma = (\sigma(1), \ldots, \sigma(n))$ the weighted sum of completion times $C_w(\sigma)$ is defined in (5.3.1).

Now for the given λ and a fixed j in N consider the $(n-1/1/./f_j^{\lambda})$ problem. Let $\pi = (\pi(1), \ldots, \pi(n-1))$ be a permutation of jobs in N_j. Using (5.3.5) in the definition of f_j^{λ} we get,

$$f_j^{\lambda}(\pi) = C_w(j\pi) = K_j^{\lambda} + C^*(\pi)$$

where $K_j^{\lambda} = \left(\sum_{k=1}^{\lambda} p_{jk}\right)\left(\sum_{l \in N} w_l\right) + \sum_{l \in N} w_l \sum_{k=\lambda+1}^{m} p_{lk}$ is a

constant independent of π and

$C^*(\pi) = \sum_{s=1}^{n-1} {}^{\prime}\pi(s) \left(\sum_{l=1}^{s} p_{\pi(l),\lambda} \right)$ is the weighted sum of completion times corresponding to the non-delay permutation schedule consistent with π in the single machine problem $(n-1/1/./f_j^{\lambda})$.

Now $f_j^{\lambda}(\pi)$ can be minimized by minimizing $C^*(\pi)$ and note that $C^*(\pi)$ can be minimized applying the rule given by Mc-Naughton (1959). Then using Remark 5.3.7 the $(n/m/F/C_w)$ problem can be solved.

(B) Number of tardy jobs $NT(\sigma)$:

For an arbitrary permutation $\sigma = (\sigma(1), \ldots, \sigma(n))$ the number of tardy jobs $NT(\sigma)$ is defined in (5.3.2). For the given λ and an arbitrary completion $(j\,\pi(1), \ldots, \pi(n-1))$ of the partial permutation (j), using (5.3.5) for $1 \leq l \leq n-1$ we get

$$T_{\pi(l)} = \max \left\{ t(j\pi(1), \ldots, \pi(l); m) - d_{\pi(l)} , 0 \right\}$$

$$= \max \left\{ \sum_{s=1}^{l} p_{\pi(s),\lambda} - d^{j}_{\pi(l)} , 0 \right\}$$

where

$$d^{j}_{\pi(l)} = d_{\pi(l)} - \sum_{k=\lambda+1}^{m} p_{\pi(l),k} - \sum_{k=1}^{\lambda} p_{j,k} , \quad 1 \leq l \leq n-1.$$

... (5.3.6)

For the given λ and a fixed j in N, note that (5.3.6) provides a new set of due dates associated with jobs in N_j. Now, for an arbitrary permutation π of the jobs in N_j we have

$$f_j^{\lambda}(\pi) = \rho(T_j) + NT^*(\pi)$$

where $\rho(x) = 0$ or 1 according as $x \leq 0$ or $x > 0$;

$$T_j = \max\{t(j; m) - d_j, 0\};$$

and $NT^*(\pi)$ is the number of tardy jobs corresponding to the non-delay permutation schedule consistent with π in the single machine problem $(n\text{-}1/1/./f_j^{\lambda})$ with new due-dates given by (5.3.6).

Now $\rho(T_j)$ being independent of π, we can minimize $f_j^{\lambda}(\pi)$ by minimizing $NT^*(\pi)$ which can be done by the rule given by Moore (1968).

(C) Maximum penalty on tardiness $MPT(\sigma)$:

For the given λ and a fixed j in N, define through (5.3.6) new due dates d_l^j for every l in N_j. Now for an arbitrary permutation π of the jobs in N_j, we can show that

$$f_j^{\lambda}(\pi) = \max\{h_j(T_j), MPT^*(\pi)\}$$

where T_j is as given in (B) and $MPT^*(\pi)$ is the maximum penalty on tardiness for the non-delay permutation schedule consistent

with π in the single machine problem $(n-1/1/./f_j^{\lambda})$ with due dates d_1^j. Now this single machine problem can be solved using the method given by Lawler (1973).

The above problems discussed under (A), (B) and (C) were solved by Arthanari (1974) when $\lambda = 1$ or m.

Case II : In a $(n/m/F/f)$ problem, for some λ, $1 < \lambda \leq m$ assume $ABCD(M_u, M_v, M_{\lambda-1})$, $1 \leq u \leq v \leq \lambda-1$ and $FCD(M_\lambda, M_v)$, $\lambda \leq v \leq m-1$.

Under this case, using Remark 5.1.5 and Lemma 5.2.4, for any given permutation $\pi = (\pi(1), \ldots, \pi(n))$, for every j, $1 \leq j \leq n$, there exists at least one j-critical path $P_j(\pi)$ of π such that $P_j(\pi)$ passes through the cells $(1, \lambda)$, $2 \leq 1 \leq j$ and (j, k), $\lambda+1 \leq k \leq m$. Consequently under the non-delay permutation schedule consistent with π, we have

$$t(\pi(1), \ldots, \pi(j);m) = t(\pi(1), \pi(2);\lambda) + \sum_{1=3}^{j} p_{\pi(1),\lambda} + \sum_{k=\lambda+1}^{m} p_{\pi(j),k} \quad \ldots \quad (5.3.7)$$

for every j such that $3 \leq j \leq n$.

Given a (n/m/F/f) problem, corresponding to the machine, M_λ and a partial permutation σ such that $|\sigma| = 2$, we define a single machine problem denoted by $(n\text{-}2/1/./f_\sigma^\lambda)$ as follows :

- the set of jobs is given by $\bar{\sigma} = N - \sigma$.
- the processing time of job 1 in $\bar{\sigma}$ is $p_{1\lambda}$.
- $f_\sigma^\lambda(\pi) = f(\sigma\pi)$ is the objective function defined over the set of all permutations π of the jobs in $\bar{\sigma}$.

Remarks analogous to 5.3.6 and 5.3.7 can be written for the present case as well. Thus for the given λ, by solving the related $n(n-1)$ single machine problems $(n\text{-}2/1/./f_\sigma^\lambda)$ corresponding to the $n(n-1)$ partial permutations σ with $|\sigma| = 2$, we can solve the (n/m/F/f) problem under case II.

Let $\sigma = (\sigma(1), \sigma(2))$ and $\pi = (\pi(1), \ldots, \pi(n-2))$ be arbitrary partial permutations such that $\sigma \cap \pi = \emptyset$. Now for the objective functions discussed in (A), (B) and (C) of case I, the corresponding $f_\sigma^\lambda(\pi)$ can be rewritten under case II using (5.3.7) as follows :

(A) $f_\sigma^\lambda(\pi) = K_\sigma^\lambda + C^*(\pi)$

where

$$K_\sigma^\lambda = w_{\sigma(1)} \sum_{k=1}^{\lambda} p_{\sigma(1),k} + t(\sigma(1),\sigma(2);\lambda) \left(\sum_{j \in N} w_j - w_{\sigma(1)} \right)$$

$$+ \sum_{j \in N} w_j \left(\sum_{k=\lambda+1}^{m} p_{j,k} \right) \text{ is a constant independent of } \pi,$$

and $C^*(\pi) = \sum_{j=1}^{n-2} w_{\pi(j)} \left(\sum_{s=1}^{j} p_{\pi(s),\lambda} \right)$ is the weighted sum of completion times corresponding to the non-delay permutation schedule consistent with π in the $(n\text{-}2/1/./f_\sigma^\lambda)$ problem.

(B) $f_\sigma^\lambda(\pi) = \rho(T_{\sigma(1)}) + \rho(T_{\sigma(2)}) + NT^*(\pi)$

where $\rho(x)$ is as defined in (B) of case I; $T_{\sigma(1)}$ and $T_{\sigma(2)}$ are defined as usual; and $NT^*(\pi)$ is the number of tardy jobs corresponding to the non-delay permutation schedule consistent with π in the $(n\text{-}2/1/./f_\sigma^\lambda)$ problem with due date

$$d_l^\sigma = d_l - \sum_{k=\lambda+1}^{m} p_{l,k} - t(\sigma(1), \sigma(2); \lambda) \quad \ldots (5.3.8)$$

for l in $\bar{\sigma}$.

(C) $f_\sigma^\lambda(\pi) = \max\left\{ h_{\sigma(1)}(T_{\sigma(1)}),\ h_{\sigma(2)}(T_{\sigma(2)}),\ MPT^*(\pi) \right\}$

where $T_{\sigma(1)}$, $T_{\sigma(2)}$ are defined as usual and $MPT^*(\pi)$ is the maximum penalty on tardiness for the non-delay permutation schedule consistent with π in the $(n\text{-}2/1/./f_\sigma^\lambda)$ problem with due dates as given in (5.3.8).

$f_\sigma^\lambda(\pi)$ can be minimized for the above cases (A), (B) and (C) by minimizing the $C^*(\pi)$, $NT^*(\pi)$ and $MPT^*(\pi)$ respectively.

R E F E R E N C E S

Achuthan, N.R. (1977) : "A special case of the ($n/m/F/F_{max}$) problem", Opsearch, vol.14, pp. 71-87.

Achuthan, N.R. (1978) : "Some special cases of the ($n/3/F/F_{max}$) problem", Tech. Report No. SQC-OR 1/78, Indian Statistical Institute, Calcutta, India (submitted to Opsearch).

Achuthan, N.R., Grabowski, J. and Sidney, J.B. (1978a) : "Optimal flow-shop scheduling with earliness and tardiness penalties", Tech. Report No. SQC-OR 3/78, Indian Statistical Institute, Calcutta, India (submitted to Opsearch).

Achuthan, N.R. and Ghosh, D.T. (1979) : "Heuristic rules for the ($n/3/F/F_{max}$) problem", Tech. Report No. SQC-OR 1/79, Indian Statistical Institute, Calcutta, India (submitted to Opsearch).

Aho, A.V., Hopcroft, J.E and Ullman, J.D. (1974) : The design and analysis of computer algorithms, Addison-Wesley, Reading, Mass.

Arthanari, T.S. and Mukhopadhyay, A.C. (1971) : "A note on a paper by W. Szwarc", Nav. Res. Log. Quart., vol.18, pp. 135-138.

Arthanari, T.S. (1974) : On some problems of sequencing and grouping, Ph.D. thesis, Indian Statistical Institute, Calcutta, India.

Ashour, S. (1970) : "An experimental investigation and comparative evaluation of flow-shop sequencing techniques", Opn. Res., vol.18, pp. 541-549.

Baker, K.R. (1974) : Introduction to sequencing and scheduling, John Wiley & Sons, Inc., N.Y.

Baker, K.R. (1975) : "An elimination method for the flow-shop problem", Opn. Res., vol.23, pp.159-162.

Brown, A.P.G. and Lomnicki, Z.A. (1966) : "Some applications of the 'branch-and-bound' algorithm to the machine scheduling problem", Opnl. Res. Quart., vol.17, pp. 173-186.

Bruno, J., Coffman, JR., E.G. and Sethi, R. (1974) : "Scheduling independent tasks to reduce mean finishing time", Comm. ACM, vol.17, pp. 382-387.

Burdyuk, V. Ya. (1969) : "The m-machine problem (m > 2)", Kibernetika, vol.5, pp. 74-76.

Burns, F. and Rooker, J. (1975) : "A special case of the 3 x n flow-shop problem", Nav. Res. Log. Quart., vol.22, pp. 811-817.

Burns, F. and Rooker, J. (1976) : "Johnson's three-machines flow-shop conjecture", Opn. Res., vol.24, pp. 578-580.

Campbell, H.G., Dudek, R.A. and Smith, M.L. (1970) : "A heuristic algorithm for the n job, m machine sequencing problem", Management Sci., vol.16, pp. B630-637.

Coffman, JR., E.G. (ed.) (1976) : Computer and job shop scheduling theory, John Wiley & Sons, N.Y.

Conway, R.W., Maxwell, W.L. and Miller, L.W. (1967) : Theory of scheduling, Addison-Wesley, Reading, Mass.

Cook, S.A. (1971) : "The complexity of theorem-proving procedures", Proceedings of the third annual ACM symposium on theory of computing, pp. 151-158.

Dannenbring, D.G. (1977) : "An evaluation of flow-shop sequencing heuristics", Management Sci., vol.23, pp. 1174-1182.

Dudek, R.A., Panwalkar, S.S. and Smith, M.L. (1975) : "Flow-shop sequencing with ordered processing time matrices", Management Sci., vol.21, pp. 544-549.

Dudek, R.A., Panwalkar, S.S. and Smith, M.L. (1976) : "Flow-shop sequencing problem with ordered processing time matrices : A general case", Nav. Res. Log. Quart., vol.23, pp.481-486.

Garey, M.R. and Johnson, D.S. (1975) : "Complexity results for multi-processor scheduling under resource constraints", SIAM J. comput., vol.4, pp. 397-411.

Garey, M.R., Johnson, D.S. and Sethi, R. (1976) : "The complexity of flow-shop and job-shop scheduling", Math. Opns. Res., vol.1, pp. 117-129.

Garey, M.R. and Johnson, D.S. (1979) : Computers and intractibility : a guide to the theory of NP-completeness, Freeman.

Giglio, R.J. and Wagner, H.M. (1964) : "Approximate solutions to the three-machine scheduling problem", Opns. Res., vol.12, pp.305-324.

Gonzalez, T. and Sahni, S. (1978) : "Flow-shop and job-shop schedules : Complexity and approximation", Opns. Res., vol.26, pp. 36-52.

Grabowski, J. and Syslo, M.M. (1975) : "New solvable cases of m-machine and n-element sequencing problem", Zastosowania Matematyki, Aplicationes mathematicae, vol.14, pp. 599-606.

Gupta, J.N.D. (1975) : "Analysis of a combinatorial approach to flow-shop scheduling problems", Opnl. Res. Quart., vol.26, pp. 431-440.

Gupta, J.N.D. (1975a) : "Optimal schedules for special structure flow-shops", Nav. Res. Log. Quart., vol.22, pp. 255-269.

Gupta, J.N.D. (1976) : "A review of flow-shop scheduling research", U.S. Postal service, Washington, DC, 20260.

Horowitz, E and Sahni, S. (1978) : "Combinatorial problems : Reducibility and approximation", Opns. Res., vol.26, pp. 718-759.

Hutchinson, G.K. and Szwarc, W. (1977) : "Johnson's approximate method for the 3 x n job shop problem", Nav. Res. Log. Quart., vol.24, pp. 153 - 157.

Ignall, E. and Schrage, L. (1965) : "Application of the branch-and-bound technique to some flow-shop scheduling problems", Opns. Res., vol.13, pp. 400-412.

Johnson, S.M. (1954) : "Optimal two and three-stage production schedules with setup times included", Nav. Res. Log. Quart., vol.1, pp.61-68.

Karp, R. (1972) : "Reducibility among combinatorial problems", Complexity of computer computations, pp.85-104, Miller, R.E. and Thatcher, J.W. (eds.), Plenum Press, N.Y.

Karp, R. (1975) : "On the computational complexity of combinatorial problems", Networks, vol.5, pp. 45-68.

Kernighan, B.W. and Plauger, P.J. (1974) : The elements of programming style, McGraw-Hill, N.Y.

Knuth, D.E. (1973) : The art of computer programming, vol. 3, Addison-Wesley, Reading, Mass.

Lageweg, B.J., Lenstra, J.K. and Rinnooy Kan, A.H.G. (1978) : "A general bounding scheme for the permutation flow-shop problem", Opns. Res., vol.26, pp. 53-67.

Lakshmanan, R., Lakshminarayan, S., Papineau, R.L. and Rochette, R.(1978): "Optimal single machine scheduling with earliness and tardiness penalties", Université du Québec à Trois-Rivières.

Lawler, E.L. (1973) : "Optimal sequencing of a single machine subject to precedence constraints", Management Sci., vol.19, pp.544-546.

Lenstra, J.K., Rinnooy Kan, A.H.G. and Van Emde Bas, P.(eds.).(1978) : Interfaces between computer science and operations research, Mathematisch Centrum, Amsterdam.

Lenstra, J.K., Rinnooy Kan, A.H.G. and Brucker, P. (1977) : "Complexity of machine scheduling problems", Ann. Discrete Math. vol.1, pp. 343 - 362.

Lenstra, J.K. (1977a) : Sequencing by enumerative methods, Mathematical Centre Tract 69, Mathematisch Centrum, Amsterdam.

Lomnicki, Z.A. (1965) : "A branch-and-bound algorithm for the exact solution of the three-machine scheduling problem", Opnl. Res. Quart. vol.16, pp. 89- 100.

McMahon, G.B. and Burton, P.G. (1967) : "Flow-shop scheduling with the branch-and-bound method", Opns. Res., vol.15, pp. 473- 481.

McNaughton, R. (1959) : "Scheduling with deadlines and loss functions", Management Sci., vol.6, pp. 1- 12.

Miller, R.E. and Thatcher, J.W. (eds.) (1972) : Complexity of computer computations, Plenum Press, N.Y.

Moore, J.M. (1968) : "An n job, one machine sequencing algorithm for minimizing the number of late jobs", Management Sci., vol.15, pp. 102- 109.

Nabeshima, I. (1961) : "The order of n items processed on m machines", J. Opns. Res. Soc. Japan, vol.3, pp. 170-175 and vol.4, pp. 1-8.

Nabeshima, I.(1977) : "Notes on the analytical results in flow-shop scheduling problem", Parts 1 and 2, Reports of the University of Electro-Communications, vol.27, pp.245-252 and 253-257.

Page, E.S. (1961) : "An approach to scheduling jobs on machines", J.Roy. Stat. Soc., series B, vol.23, pp. 484 - 492.

Palmer, D.S. (1965) : "Sequencing jobs through a multi-stage process in the minimum total time - A quick method of obtaining a near optimum", Opnl. Res. Quart., vol.16, pp. 101 - 107.

Ramamurthy, K.G. and Prasad, V.R. (1978) : "Some classes of flow-shop problems", Discussion paper no.7821, Indian Statistical Institute, Delhi Campus, India.

Rinnooy Kan, A.H.G. (1976) : Machine scheduling problems : Classification, complexity and computations, Nijhoff, The Hague, Netherlands.

Sidney, J.B. (1977) : "Optimal single-machine scheduling with earliness and tardiness penalties, Opns. Res., vol.25, pp. 62-69.

Smith, W.E. (1956) : "Various optimizers for single-stage production", Nav. Res. Log. Quart., vol.3, pp. 59-66.

Szwarc, W. (1968) : "On some sequencing problems", Nav. Res. Log. Quart., vol.15, pp. 127-155.

Szwarc, W. (1971) : "Elimination methods in the m x n sequencing problem", Nav. Res. Log. Quart., vol.18, pp. 295-305.

Szwarc, W. (1974) : "Mathematical aspects of the 3 x n job-shop sequencing problem", Nav. Res. Log. Quart., vol.21, pp. 145-153.

Szwarc, W. (1974a) : "A note on mathematical aspects of the 3 x n job-shop sequencing problem", Nav. Res. Log. Quart., vol.21, pp. 725-726.

Szwarc, W. (1977) : "Optimal two-machine ordering in the 3x n flow-shop problem", Opns. Res., vol.25, pp. 70-77.

Szwarc, W. (1977a) : "Special cases of the flow-shop problem", Nav. Res. log. Quart., vol.24, pp. 483-492.

Szwarc, W. (1978) : "Permutation flow-shop theory revisited", Nav. Res. log. Quart., vol.25, pp. 557-570.

Szwarc, W. (1979) : "The critical path approach in the flow-shop problem", Opsearch, vol.16, pp. 98-102.

Ullman, J.D. (1975) : "NP-complete scheduling problems", J. Comput. Syst. Sci., vol.10, pp. 384-393.

Ullman, J.D. (1976) : "Complexity of sequencing problem", Computer and job-shop scheduling theory, pp. 139-164, Coffman, JR., E.G. (ed.), John Wiley & Sons, N.Y.

Wagner, H.M. and Story, A.E. (1963) : "Computational experience with integer programming", Industrial scheduling, pp. 207-212, Muth, J. and Thompson, G.L. (eds.), Prentice-Hall, Englewood Cliffs, N.J.